Программа-клиент протокола Z39.50

Смирнов В.М.[+], Капустин В.А.[*]

Санкт-Петербургский государственный университет
Санкт-Петербург

[*]*vak@mail.nw.ru* [+]*vms@bmc.usr.pu.ru*

Доступные поисковые клиенты в данный момент представлены, в основном, в виде шлюзов к поисковым системам (выполненных на основе CGI). Соответственно, пользователь поисковой системы работает с достаточно примитивными HTML-формами. В качестве транспортного протокола используется HTTP. Эта схема достаточно проста для реализации и для использования, но в ней есть ряд недостатков:

- Интерфейсные возможности HTML очень низкие.
- Невозможно сохранять для каждого пользователя большие объемы информации (настроек и т.д.).
- Дополнительная поисковая информация (что, где, и как можно искать) либо отсутствует, либо является неактивной и не может быть использована непосредственно.
- Зачастую такие поисковые клиенты (ПК) требуют достаточно больших усилий для сопряжения с поисковой системой (ПС) и не обладают необходимой универсальностью, что вызывает необходимость вносить изменения в поисковый клиент при изменении конфигурации поисковой системы.
- Помимо ставших традиционными клиентских Web-интерфейсов к поисковым системам, в качестве ПК используются также интегрированные клиенты производителей ПС (например ERL поисковый клиент фирмы Crossnet Systems ltd, поисковый plugin Alexa, и др.), но в этом случае использование такого ПО существенно снижает область поисковых операций, предоставляемых пользователю, до уровня возможностей конкретной ПС.

Протокол Z39.50 позволяет избежать этих недостатков, т.к. позволяет создать отдельного клиента поисковой системы, который бы сообщался с самой ПС через этот протокол. Такой клиент должен максимально использовать те “выгодные” стороны протокола, которые позволяют обойти часть проблем, о которых говорилось выше. При разработке клиентской части протокола естественным шагом было использование следующих его достоинств:

Первая Всероссийская научная конференция
ЭЛЕКТРОННЫЕ БИБЛИОТЕКИ:
ПЕРСПЕКТИВНЫЕ МЕТОДЫ И ТЕХНОЛОГИИ,
ЭЛЕКТРОННЫЕ КОЛЛЕКЦИИ
19 - 21 октября 1999 г., Санкт-Петербург

Протокол является строго стандартизированным и описан в синтаксисе ASN – это позволяет не учитывать никаких внешних факторов, касающихся, в частности, серверных реализаций ПС на основе этого протокола.

- Реестр предоставляемых протоколом поисковых и вспомогательных служб является конфигурируемым – это дает возможность использовать одинаковые клиентские части для работы с различными ПС.
- Многие внутренние данные также строго описаны внутри протокола (например, наборы атрибутов)
- Специфика поисковой части протокола (например, наличие памяти и др.) позволяет оптимизировать процесс поиска на базе клиентской части протокола.

Конкретные задачи, поставленные при создании такого клиента можно сформулировать так:

клиент должен

- Поддерживать протокол Z39.50 версии 3.
- Содержать возможности для работы с кириллицей.
- Обладать графическим пользовательским интерфейсом.
- Свободно функционировать в сети без ограничений на опрашиваемые сервера и их изготовителей.
- Предоставлять средства для работы с получаемыми данными.
- Поддерживать зарегистрированные в протоколе форматы записей и предоставлять возможность для использования собственных.
- Предоставлять возможность для работы с зарегистрированными в реестре протокола наборами атрибутов.
- Быть максимально конфигурируемым.

Для реализации такого клиента был выбран язык Java. Этот язык, за счет своей объектной ориентации, упрощенной работы с сетью и платформенной независимости позволяет достаточно просто удовлетворить необходимые требования. Как известно, Java предлагает два пути для написания программы: апплет и независимая программа. В данном случае все будет рассматриваться на примере отдельной программы, т.к. в случае апплета необходимо делать определенные оговорки, возникающие в связи с броузером, в котором выполняется апплет. Нами главный класс реализован в виде апплета, который в случае запуска из броузера выполняется в броузере, а в случае запуска на виртуальной машине просто

добавляется в пустой Frame. В качестве броузера может использоваться любой Java-совместимый броузер. Клиент представляет из себя средство для работы с удаленными ПС по протоколу Z39.50 и включает в себя некоторые дополнительные возможности.

чен от IP-адреса машины, с которой был получен апплет. В Netscape и Hotjava эта проблема решается путем выделения пользователем для апплета привилегий UniversalConnect и FileAccess. В остальных случаях решение лежит в цифровой сертификации апплета или построении архитектуры с

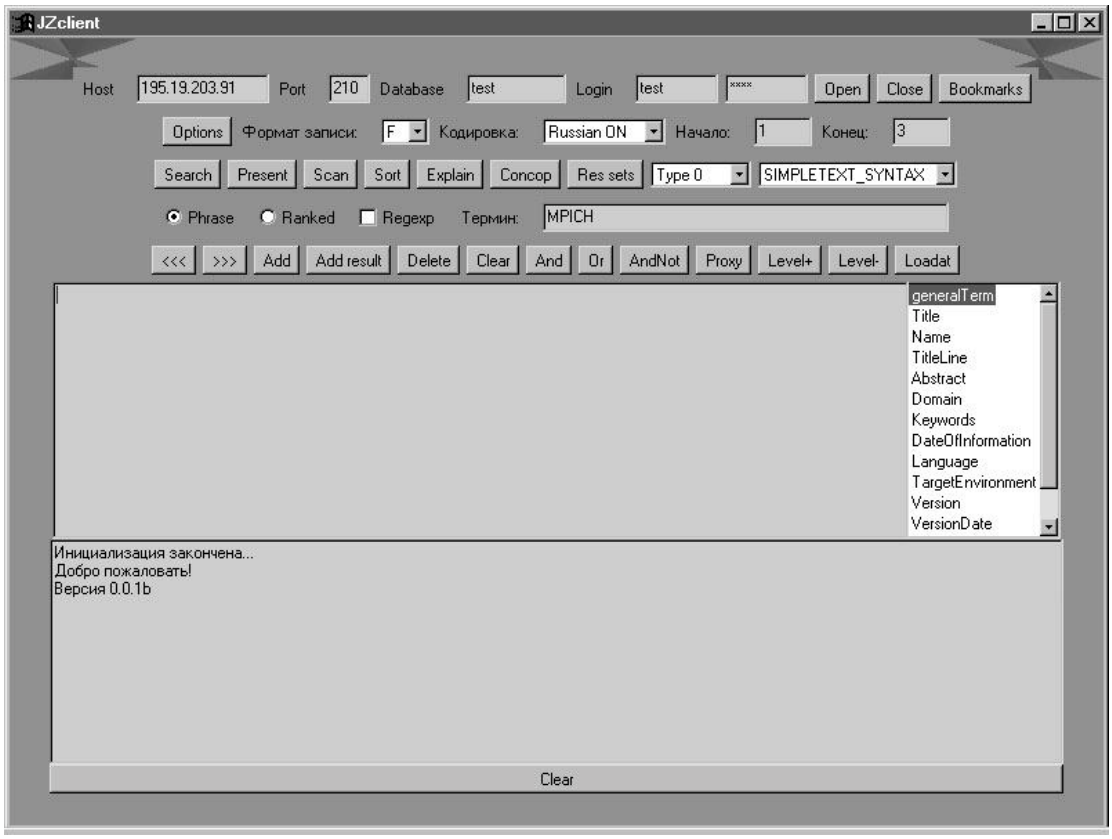

Рис.1. Общий вид программы-клиента протокола Z39.50

Реализацию клиента можно разделить на четыре блока:

- Блок работы с Z39.50 и сетью
- Блок поддержки русского языка
- Visual Query Biulder(VQB) и обработчик RPN-структур.
- Блок работы с данными. Вспомогательные функции.

Все перечисленные блоки будут рассматриваться отдельно, т.к. каждый из них выполняет строго определенную функцию.

Блок работы с Z39.50 и сетью

В качестве интерфейса к блоку для работы с протоколом и сетью используются три ряда управляющих элементов, позволяющих создавать сетевые соединения и вызывать соответствующие сервисы протокола. Для работы с сетью используются стандартные Java-socket'ы – это позволяет создавать и разрывать соединения при помощи одной стандартной функции connect(). Здесь следует сделать оговорку, касающуюся случая выполнения клиента в броузере: известно, что но умолчанию апплету в броузере не позволено общаться по сети с удаленными машинами, IP-адрес которых отли-

Middleware. В случае отсутсвия какого-либо решения апплет может служить лишь продвинутым Web-интерфейсом для работы с Z-сервером, IP-адрес которого совпадает с IP-адресом сервера, с которого был получен апплет.

В настоящий момент в клиенте реализованы следующие службы протокола:

- Служба **Init** - клиент осуществляет соединение с сервером, причем при инициализации соединения клиентом автоматически запрашивается использование всех допустимых служб. Так же в реализацию этой службы входит блок поддержки контроля доступа.
- Служба **Search** – об этой службе и ее реализации необходимо говорить отдельно (см. пункт VQB).
- Служба **Present** – включает в себя стандартные средства протокола для управления выводом (указание диапазонов вывода, выбор синтаксиса и др.) плюс дополнительные средства для работы с данными (см. пункт Средства для работы с данными)
- Служба **Scan** – реализована в виде отдельного интерфейсного блока, результатом работы которого является список терминов с возможностью добавления выбранных элементов в текст текущего запроса.

- Службы **NamedResSet+DelResSet** – программа автоматически генерирует имена результатов для каждой поисковой операции и позволяет использовать эти результаты в качестве операндов в последующих поисковых операциях, сохраняя подробную историю поиска.
- Служба **Explain** – программа предоставляет пользователю возможность получить информацию об опрашиваемом сервере. Информация выбирается по одной из рубрик (*databaseinfo*, *targetinfo* и др.)

Использование языка Java для разработки данного клиента делает естественным шагом использование для программирования объектно-ориентированного подхода: в качестве несущих типов данных выбраны два класса: Zdata и Zstring, которые выполняют все функции по кодированию/раскодированию сетевых пакетов, разделению структур протокола и т.д. Zstring – это всего лишь набор байтов, с возможным редактированием и записью в сокет сервера. Этот класс-класс самого нижнего уровня, работающий непосредственно с сетью. Сам класс Zstring может порождаться (порождать) из другого класса:Zdata. Zdata – уже более сложный класс, структурированный и содержащий информацию о данных. Он пригоден для обработки процедурами сервиса. Между классами Zstring и Zdata существует взаимно однозначное соответствие. Во время посылки запроса строися нужный Zdata, превращается в соответствующий Zstring, а при приеме ответа наоборот. Такие типы данных предоставляют возможность не программировать каждый сервис протокола отдельно. При такой организации все службы протокола пользуются одинаковыми типами данных, что делает программу прозрачной и позволяет обмениваться данными внутри клиента. Таким образом, все процедуры служб работают по одной схеме: по данным создается Zdata, переводится в Zstring, записывается в socket и наоборот. Отличаются лишь функции обработки данных.

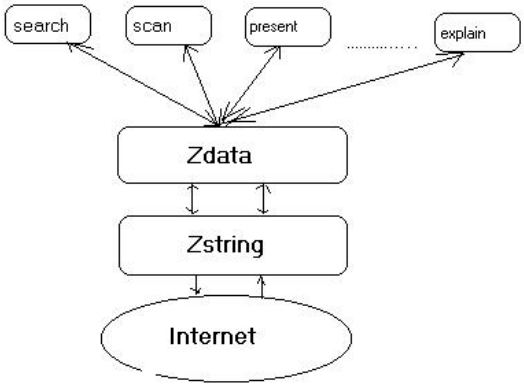

Рис.2. Взаимодействие данных в клиенте

Описание API протокола выделено также в отдельный класс, который содержит все константы, при помощи которых описан протокол.

Блок работы с русским языком

Как известно, в Java для работы с символьной информацией предлагается кодировка Unicode, поэтому все данные с которыми работает клиент находятся именно в этой кодировке. Проблема русского языка распадается на две части: отображение русского текста в том случае, когда Z-сервер корректно поддерживает кириллицу, и работа с сервером, не поддерживающим кириллицу. В первом случае блок русского языка выполняет простую функцию перекодировки данных из кодировки документа в Unicode. Во втором случае предусмотрен вариант представления русского текста при помощи символов, поддерживаемых сервером. К примеру, русский текст

«Библиотека процедур для параллельного программирования»

может быть представлен на сервере в виде текста:

«Bibliosfka pqowfetq elZF paqallflZCnodo pqodqammiqocaniZF»

Для этого в блок поддержки русского языка входят две процедуры: code и decode. Они преобразуют строки в соответствии с выбранной кодировкой. Таким образом, сервер не должен поддерживать кириллицу – на нем данные хранятся в кодированном виде и раскодируются клиентом.

Visual Query Biulder(VQB).

В качестве основного средства для поиска в клиенте реализован «архитектор запросов» (Visual Query Buillder) – средство для упрощенного построения запросов.

Этот механизм позволяет существенно упростить процедуру поиска, спрятав от пользователя лишнюю информацию касающуюся внутреннего представления запроса в протоколе. Этот интерфейс во многом похож на построитель выражений в Microsoft Access, с той лишь разницей, что продуктом его работы является RPN-структура (запрос в обратной польской записи) и работает он с другими объектами.

Для визуализации запроса выбран метод представления этого запроса в виде подобия дерева, содержащегося внутри редактируемой текстовой области. Этот метод более выгоден, нежели представление запроса в виде обычной строки, т.к. способен наглядно представлять запросы достаточно высокой сложности, а также предоставляет возможность для быстрой их корректировки.

Интерфейс к VQB выглядит таким образом (рис 3):

- переключатели **Phrase&Ranked** определяют какой тип поиска будет выполняться для этого термина.
- в поле термин пользователь может ввести фразу или термин и добавить его в рабочее окно при помощи кнопки **Add**. Далее в процессе поиска пользователь может изменять введенный термин прямо в рабочем поле. Если при нажатии кнопки **Add** был выбран атрибут в списке справа - термин будет добавлен для поиска только по выбранному атрибуту.
- кнопка **AddResult** позволяет добавить в запрос в виде самостоятельного термина поименованный результат предыдущего поиска.
- далее идут кнопки добавляющие операторы.
- кнопки **Level+/-** увеличивают/уменьшают приоритет выбранной в рабочем поле области на 1.
- кнопки <<<,>>> реализуют навигацию по сделанным ранее запросам.

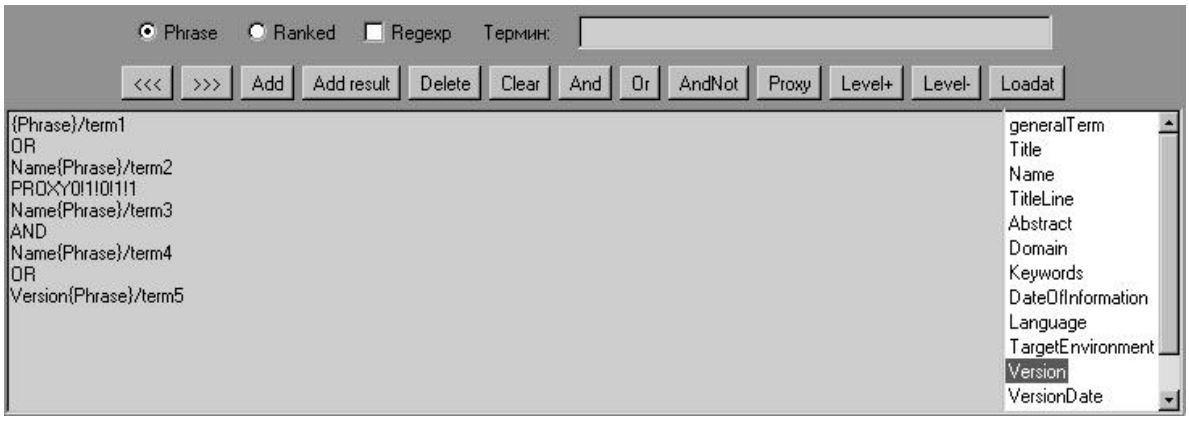

Рис.3. Визуальный построитель запросов

Таким образом, пользователь должен заполнить рабочую область и передать ее содержимое процедуре обработки нажатием кнопки **Search**. Затем содержимое рабочей области преобразуется в RPN-строку (в протоколе Z39.50 поисковые запросы представляются при помощи RPN):

```
RNP string ::=
        Argument|Argument+Argument+Operator
Argument ::= Operand|RPN string
operand ::= AttributeList+Term|ResultSetid
        |Restriction
Restriction ::= ResultSetid+AttributeList
operator ::= AND|OR|AND-NOT|Prox
Prox ::=Distance+Relation+Unit
        +Ordered flag+Exclusion flag
Distance ::= integer
Relation ::= «<»|»>»|»=»|»>=»|»<=»
Unit ::= Character|Word|Sentence|Paragraph
        |Section|Chapter|Other Defined
Ordered flag ::= boolean
Exclusion flag ::= boolean
```

Заполненная пользователем рабочая область должна быть преобразована в такую же структуру. Этот процесс выполняется в два этапа.

Первый этап заключается в преобразовании в строку типа:

```
...(((name1/term1)OR(name2/term2))AND
((name3/term3)ANDNOT(name4/term4))
```

На этом этапе выполняется проверка правильности составленного запроса, кодировка русских слов (если эта опция включена), подстановка номеров атрибутов и другая подготовительная работа. Стоит отметить отдельно процесс обработки трех и более терминов (связанных операторами) в одной группе. В этом случае клиент сам расставит скобки в соответствии с приоритетами операторов в группе. Также возможна расстановка скобок последовательно (если соответствующая опция включена).

Отдельной частью интерфейса VQB является список атрибутов – из этого списка пользователь должен выбрать необходимый атрибут, вместе с которым вводимые им термины будут добавляться в запрос. Клиент предусматривает возможность загрузки и сохранения наборов атрибутов для поиска.

Блок работы с данными.

Блок работы с данными реализован в виде отдельного окна.

В этот блок входят процедуры обработки пролученных с сервера записей:

- процедура раскодирования – превращает данные из битового массива в символьный.
- процедура обработки – раскодирует русскоязычный текст (если опция включена), выделяет имена полей, определяет формат записи.
- процедура форматирования – форматирует текст документа.
- вспомогательные процедуры – для сохранения записей, загрузки, навигации по вектору записей и т.д.

Раскодирование данных происходит в соответствии с заложенными в протокол стандартами: из записи результата извлекается информация о содержимом записи (octet stream, text и др.) и далее обрабатывается в соответствии с типом данных.

Процедура обработки прежде всего определяет синтаксис полученной записи (GRS1, MARC, …) и пытается выделить поля атрибутов. Далее происходит (если необходимо) раскодирование кириллического текста.

После обработки программа приводит полученные данные к внутреннему формату представления: хэш-таблице, где выделенные атрибуты служат ключами для содержимого полей. Для управления выводом записей в клиенте предусмотрен внутренний язык форматирования – это средство, позволяющее пользователю, составив программу на этом языке, самостоятельно управлять выводом. Используются следующие команды:

- *N.N - сегмент+смещение (*-весь текст поля)
- \c=N - цвет
- \CR - с новой строки
- \X=N - вставить N пробелов
- \T=text - вставить текст
- \V=text,format - проверка на значение, если true, то format
- \D - стереть пустую строку
- &AtributeName=format – вывести соответствующее AttributeName поле из хэш-таблицы с введенным форматом.

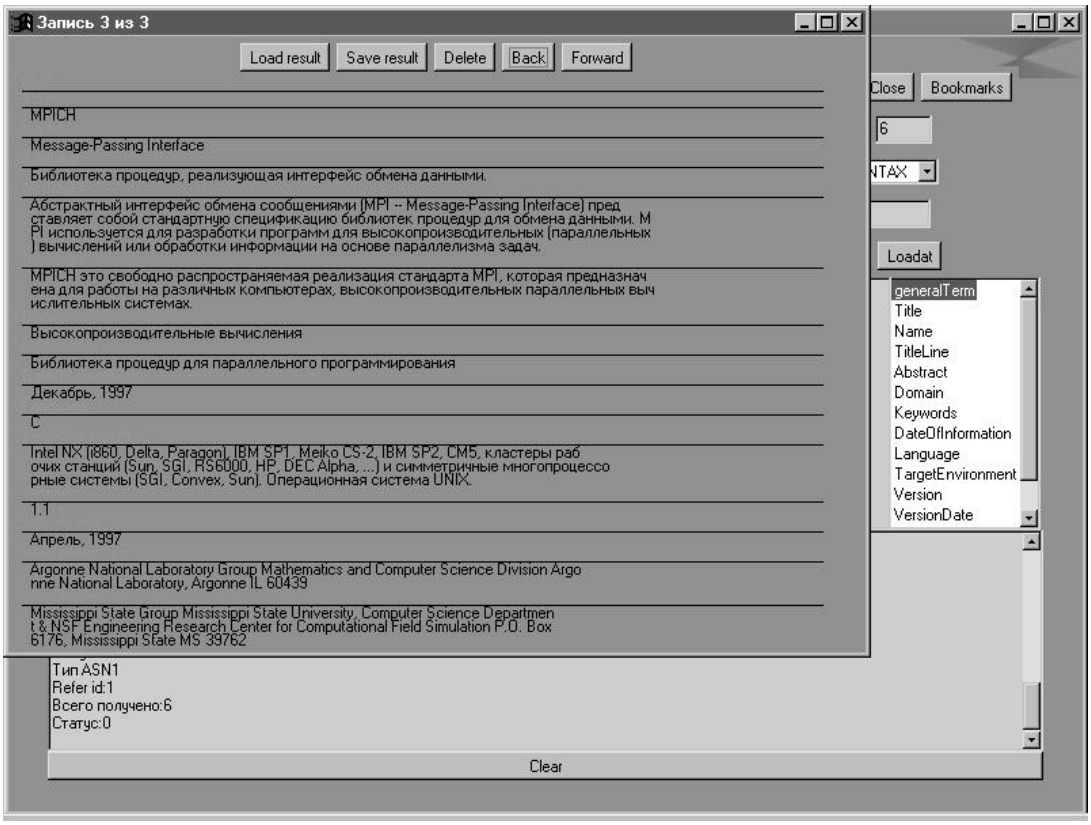

Рис.4. Форматирование результатов поиска

Таким образом, пользователь составляет программу вывода. Например, пользователь хочет выводить лишь два поля: Title и Keywords. Тогда формат вывода может выглядеть так:

&Title=\X=10 *1.10 \c=1 \CR *10.20 \CR \T="End."
& Keywords=* \D \c=2 \T=”End”

Заключение

Описанное выше прикладное средство поиска обеспечивает работу с протоколом Z39.50, но не использует возможности протокола на полную мощность. Среди таких возможностей можно назвать возможность конкурентного выполнения запросов (когда пользователю предоставляется возможность выполнять несколько запросов параллельно в рамках одной Z-сессии или разных).

Также клиент мог бы содержать возможности для использования нетривиальной для поисковых систем вообще службы **Resource control**, которая позволяет информировать пользователя о состоянии ресурсов на сервере, где он производит поиск, и в некоторой степени позволяет управлять ими. Так же в клиенте пока не реализован механизм связи между результатами **Explain**-запросов и настройками самого клиента (это обстоятельство обусловлено тем фактом, что служба **Explain** реализуется разработчиками серверов поразному) Учет этих и других замечаний мог бы превратить данного клиента в действительно универсальное (благодаря Java-реализации) средство для поиска в рамках стандарта Z39.50.

Работа поддержана РФФИ, грант 98-07-91197.