УДК 681.324

ОРГАНИЗАЦИЯ ПОИСКА В ИНФОРМАЦИОННОМ WEB-ПОРТАЛЕ

А.В. Босов, Р.Б. Чавтараев

Описано решение, реализованное в составе действующего Информационного web-портала РАН (`www.ras.ru`) с целью расширения его функциональности услугами гибкой поисковой системы. Подробно рассмотрена архитектура и программная реализация этого решения.

1. Введение

Поиск информации — одна из первых задач, которую пытается решать пользователь, подключившись к Интернет. Другая заинтересованная сторона предоставляет информацию, стараясь обеспечить её лёгкую доступность. Таким образом, разработчик информационной системы для Интернет должен предложить пользователю удобные инструменты для поиска предоставляемых ресурсов. В современном web-пространстве имеется немало вариантов организации поисковых систем. Но до сих пор для успешного поиска нужной информации пользователю требуется иметь довольно высокую квалификацию, как минимум, опыт, который есть не у каждого пользователя, и поэтому подавляющее большинство запросов остаются с нерелевантными ответами: информационного «мусора» больше, чем полезных ссылок.

Такое положение сформировалось постепенно. Когда пространство web только появилось, было мало ресурсов, все они были статичными, и простейшие поисковые машины, индексируя тексты, успешно решали проблемы пользователей (тоже, кстати, немногочисленных). Далее данных стало больше, и они перестали храниться в статике, а размещались, как правило, в базах данных, работать с которыми поисковые машины уже не могли. Можно считать, что так контент, доступный в Интернет, поделился на «поверхностный» (Surface Web) и «глубинный» (Deep Web). Любопытно, что оценки, характеризующие объёмы этих частей (например, в [1] даётся оценка отличия объёмов

в 400–550 раз), практически однозначно показывают, что искать нужную информацию в «поверхностном» web, индексируемом глобальными поисковыми машинами, бесполезно.

При этом на данный момент большинство ресурсов предоставляют «внутри себя» собственные средства поиска, которые дают заведомо релевантные результаты. Таким образом, можно с успехом создавать централизованные системы, объединяющие всю информацию в рамках некоторой достаточно широкой темы. Известны положительные результаты такого подхода [2, 3], но всё-таки очевидно, что он не универсален как из-за объёмов информации, так и из-за сложности поддержки контента: для более или менее широкой аудитории свести все данные к единой структуре, обеспечить их полноту и актуальность чрезвычайно сложно. Таким образом, для совершенствования поиска в Интернет нужны поисковые службы, обладающие некоторыми интеллектуальными возможностями.

Частично улучшают ситуацию web-порталы, но, в основном, не за счёт особо удобных систем поиска, а за счёт ограничения на индексируемые ресурсы, построения неких тематических каталогов, «вертикального» объединения пользователей и т.п. Эффективность при этом обеспечивается тем, что, найдя нужный портал, пользователь дальше его средствами ищет информацию только в той предметной области, которую действительно отражают ресурсы портала. Но и это решение нельзя признать окончательным. Во-первых, есть области, где искусственно ограничить тематику довольно трудно. Например, в настоящей работе обсуждается портал Российской академии наук, поэтому охарактеризовать тематику можно было бы как «научная информация», но, очевидно, на самом деле такое «ограничение» мало что проясняет. Во-вторых, предъявляются очень жёсткие требования и ограничения к ресурсам, подключаемым к порталам. Обычно предлагается ограниченный перечень адаптеров к ограниченному же набору распространённых систем, расширение же списка — крайне трудоёмкая работа, связанная с программированием именно для того портального решения, которое используется. Конечно, полностью отказаться от разработки нельзя (пока, по крайней мере), но кое-что усовершенствовать в технологии поиска в web-портале можно. Можно, например, облегчить жизнь разработчика (администратора или даже модератора портала), предложив более гибкие средства интеграции ресурсов для целей поиска.

В настоящей работе рассмотрено решение, реализованное в действующем Информационном web-портале РАН (www.ras.ru) [4, 5] с целью усовершенствования механизма поиска информации в научном информационном пространстве РАН. Для этого портала есть и упомянутая широкая область тематики, и внедрённое и эксплуатируемое решение, к которому периодически подключаются новые ресурсы, в том числе и в большей степени для поиска по ним, что делает поставленную задачу весьма актуальной.

2. Варианты организации поиска в web-системах

Для определения места обсуждаемой далее разработки приведём некоторые общие сведения об информационно-поисковых системах. Первый самый известный представитель таких систем — глобальная поисковая машина. С её помощью для множества сайтов, подключённых к соответствующему сервису индексации, могут выполняться полнотекстовые запросы. Функционирование глобальной поисковой машины обеспечивают роботы (crawlers), которые выполняют просмотр ссылок для зарегистрированных сайтов и индексируют текстовое содержание страниц, возможно, с учётом простейших метаданных. Обработку полнотекстовых запросов выполняет поисковый сервер, использующий построенный при сканировании индекс. Алгоритм работы поискового сервера может основываться на разнообразных методах информационного поиска [6–8], в том числе могут учитываться особенности терминологического состава проиндексированных документов, структуры хранилища, тематические каталоги, различные статистические методы и др. В любом случае пользователь будет пытаться общаться с поисковой машиной на естественном языке той предметной области, которую он имеет в виду, но о которой вряд ли что-то «знает» поисковая машина. Хотя семантическому анализу запросов и ресурсов посвящено множество исследований, но реальных решений, доведённых до практического использования в Интернет, пока нет. Кроме того, напомним, что поиск изначально осуществляется только в «поверхностном» web.

Из сказанного вовсе не следует, что глобальные поисковые машины не нужны. Несмотря на кажущуюся простоту, их место в Интернет прочно и надолго закреплено, по крайней мере, в качестве начальной точки при поиске ресурсов. Кроме того, для большинства известных глобальных поисковых машин наблюдается и определённый прогресс. Многие сайты, изначально созданные для предоставления услуг по поиску в Интернет, например Yahoo, AltaVista, Yandex, Rambler, постепенно превращаются, а многие уже оформились в виде полноценных «горизонтальных» порталов. В этих порталах объединены функции уже не одной поисковой службы, а нескольких сервисов, которые позволяют более точно определиться с предметной областью, интересующей пользователя, и уточнить запрос (даже если это происходит и неявно для пользователя).

Конечно, этого всё ещё мало. Релевантность в любом случае мала, и, к сожалению, чем более подготовлен пользователь, чем точнее он готов ограничить условия поиска, тем сложнее ему найти инструмент, удовлетворяющий его потребностям.

Другой класс систем, обеспечивающих пользователей услугами поиска, — специализированные предметно-ориентированные системы. Узкая специализация и возможность доступа к «глубинному» web-контенту, охватывающему всю специализацию, в рамках одной системы позволяют обеспечить значительно более высокое качество результатов поиска. При этом и пользовательский интерфейс (как при формировании поискового запроса, так и при выводе результатов поиска) имеет максимально удобный вид. В рамках одной системы, как правило, легко структурировать тематическую информацию, сопроводив документы (объекты) формальными описаниями связанных с ними терминов предметной области, в том числе, с использованием словарей и классификаторов. В итоге появляется возможность сформировать так называемый атрибутный (атрибутно-полнотекстовый, в случае полнотекстовой индексации терминов) запрос, максимально детализирующий потребности пользователя. Проблемой остаются обнаружение этой самой специализированной предметно-ориентированной системы и её взаимодействие с другими системами, ориентированными на ту же самую или тематически близкую предметную область и, возможно, не менее полезными. Если для преодоления первой проблемы можно создавать некие глобальные предметные каталоги (это, конечно, несложно, но вот эффективность такого решения будет с ростом числа систем стремиться к нулю), то вторая проблема требует уже создания некоторой распределённой среды, заставить в которой работать «старые» узкоспециальные системы непросто.

Таким образом, нужны другие технологии. Ключевая идея решения этой проблемы одна, хотя её можно по-разному интерпретировать. Нужны некий представитель, хорошо разбирающий в узкой предметной области, и каталог (рубрикатор, онтология, база знаний и т.п.) «известных» предметов. В этой идее нет ничего революционного, она уже давно и успешно используется в электронных библиотеках, равно как и рассуждения о возможности переноса идеи адаптеров в Интернет [9]. Электронные (цифровые) библиотеки — важнейший класс информационно-поисковых систем. Реализуемые в них технологии основаны на понятии метаданных, позволяющих создавать коллекции распределённых ресурсов, поддерживаемых самостоятельными системами, вошедшими в коллекцию на федеративной основе (по принципу разделения ответственности за информацию между организациями, входящими в систему): коллекции распределённых в Интернет ресурсов обмениваются описательной информацией, формируя репозиторий метаданных, над которым работает индексная служба и служба поиска. Для организации такой распределённой среды применяются хорошо известные и распространённые протоколы, такие как HTTP, Z39.50, LDAP, CORBA.

В качестве стандартного примера метаданных репозиториев электронных библиотек можно привести библиографические атрибуты документов: название, аннотация, рубрики, ключевые слова, авторы и т.д. Экстрагированные метаданные, описывающие документ и его местонахождение, хранятся на поисковом сервере библиотеки и используются пользователем для поиска информации: на основе метаданных строятся индексы (в том числе, и полнотекстовые) и выполняется атрибутный (атрибутно-полнотекстовый) поиск.

Аналогично поиску в специализированных системах за счёт уточнения семантики искомых термов можно задавать условия на атрибуты. Результат такого поиска будет максимально удовлетворять сформулированному пользователем запросу, но при этом всё-таки выполняться запрос будет над некоторым централизованным (быть может, виртуальным) репозиторием метаданных, сами же данные задействовать в поиске, вообще говоря, не предполагается. Кроме того, входящие в библиотеку системы должны следовать некоторой общей схеме репозитория (метарепозитория), по крайней мере, поддерживать собственную подсхему, согласующуюся с метарепозиторием. Таким образом, сложности использования имеют место и здесь.

Из сказанного можно сделать вывод о наиболее перспективном направлении в развитии информационно-поисковых систем. Есть две хорошие идеи: порталы и электронные библиотеки. Объединение и развитие этих концепций непосредственно в интересах поиска должно дать выигрыш в доступности нужных данных для конечного пользователя.

3. Определение используемых технологий

Итак, существует портальное решение `www.ras.ru`, которое можно использовать для реализации выбранной стратегии. Элементы программной инфраструктуры портала, привлекаемой для нужд поиска, рассмотрим далее (собственно всё, что создавалось в портале, на эти цели и ориентировалось, поэтому все решения, реализованные ранее, используются и здесь). Задача портала состоит в интеграции разнородной информации и обеспечении на разных уровнях интероперабельности между системами в целях выполнения информационного поиска в распределённой среде web-портала.

Вначале определим требования к формату данных, используемых для обмена информацией, и к технологии обмена данными.

Предъявляемые в рамках портала требования к формату обмена данными следующие:

— *универсальные выразительные возможности* (формат обмена должен позволять представлять любой вид данных, поскольку нельзя учесть интересы всех потенциальных систем);

— *синтаксическая интероперабельность* (формат обмена должен позволять легко создавать или получать синтаксические анализаторы, прикладные интерфейсы, необходимые для манипулирования данными; приложения должны иметь возможность читать данные и получать их представление, с которыми они смогут работать);

— *семантическая интероперабельность* (это наиболее важное и сложное требование к формату обмена состоит в том, что данные должны быть понятными, что связано с установлением соответствия между терминами, используемыми в данных, а это требует анализа содержимого.

Требования к технологии обмена данными следующие:

— технологические решения должны максимально опираться на *открытые стандарты* манипулирования с данными;

— технологические решения должны способствовать распространению и *использованию других технологий и служб*, уже реализованных в портале (таких, как подсистема безопасности, поддержка мультиязычности и др.);

— предъявляемые требования и влияние на *подключаемые системы* (информационные источники) должны быть минимизированы.

Удовлетворять перечисленным требованиям предлагается за счёт использования следующих решений:

— для обмена данными с информационными источниками используются формат XML [11] (для неструктурированных данных), форматы реляционных данных (для структурированных данных); язык XML обеспечивает все предъявленные требования, за исключением семантической составляющей; этот же формат используется для обмена неструктурированными данными между внутренними службами и подсистемами портала;

— применяется технология обмена данными с информационными источниками, основанная на распространённых стандартах доступа к данным (ODBC, OLEDB и т.п.), web-services [12];

— применяется унифицированный формат для обмена структурированными данными между внутренними службами и подсистемами портала, основанный на реализации реляционной модели данных в используемой платформе .Net (класс DataSet);

— применяется формат описания структур данных — XML-Schema [13], добавляющий семантическую составляющую к XML.

Далее, поскольку речь идёт о функционале для распределённой системы и о взаимодействии с неоднородными источниками информации, то, по аналогии с электронными библиотеками, делается вывод о необходимости существования слоя промежуточного программного обеспечения (mediator middleware). Через этот слой службой поиска осуществляется передача поисковых запросов и получение данных федеративных информационных источников. В случае электронных библиотек основными компонентами промежуточного слоя являются посредники/медиаторы (mediator) и адаптеры/оболочки (wrapper). Адаптер используют для организации доступа к источнику информации, поддерживая интерфейсы доступа к данным (для каждого источника — свой адаптер, владеющий информацией об информационном источнике и способах взаимодействия с ним). При получении запроса адаптер обращается к источнику через предоставляемый источником интерфейс, получает от источника данные и конвертирует их в некий общий для системы целевой формат. Посредник осуществляет интеграцию данных различных источников, предоставляемых адаптерами, и может взаимодействовать как с адаптерами, так и с другими посредниками, что позволяет строить сложные сети посредников, взаимодействующих между собой.

Задача поиска в распределённой среде накладывает свою специфику, в том числе, и на взаимодействие с информационными источниками. Во-первых, присутствуют только запросы на получение данных и отсутствуют запросы на модификацию. Во-вторых, спектр запросов на получение данных может быть достаточно широк и включать в себя массу условий. При этом несмотря на то, что каждая система хранения данных оперирует запросами для выборки данных и поддерживает, как правило, некоторый язык запросов, пользователю всё-таки надо предоставить интерфейсную форму с элементами управления для определения значений поиска, условий и т.п. Естественно, что языковое выражение запроса (даже с учётом того, что запрос может быть выражен либо в текстовом представлении, т.е. в виде команды на каком-либо языке, либо в виде программной функции с параметрами) не рассматривается, и допустимый максимум в данном случае — это использование условных операторов AND, OR и других в поисковом поле.

Из этого следует, что необходим механизм, который преобразует данные формы в запрос и передаёт этот запрос на выполнение информационному источнику. Как и в электронных библиотеках, в нашем случае задачу взаимодействия с информационным источником решает адаптер. Единственным существенным отличием портальных адаптеров от традиционных является конкретизация требований к предоставляемым ими интерфейсам за счёт сужения решаемой задачи.

В портале реализован и другой традиционный элемент электронных библиотек — виртуальная (каноническая) схема. Виртуальная схема по сути состоит их метаданных, описывающих все сущности, с которыми оперирует служба поиска наряду с другими подсистемами портала. Она представляет собой набор типов (описаний объектов), их мультиязыковые представления, связи и различные технологические атрибуты. Пользователю предлагается формулировать запросы в терминах этой канонической схемы. Адаптер портала отвечает за реализацию набора типов (подсхемы), которые поддерживает информационный источник.

Задача адаптера — выполнить динамическую генерацию команды и списка параметров со значениями на основе данных, переданных поисковой формой. Далее сформированная команда выполняется адаптером в терминах информационного источника, используя протоколы доступа к данным, реализуемые источником.

Данные, полученные из адаптера, хотя и имеют унифицированный формат, однако нуждаются в приведении к терминам виртуальной схемы. В задачу адаптера, таким образом, входит преобразование полученных данных. Например, для поисковой web-системы результатом поиска должен являться URL объекта поиска. В результирующем наборе, переданном адаптером, может в явном виде не содержаться поля URL, однако его можно сформировать из других полей. Адаптер реализует эту задачу на основе своих метаданных.

4. Место службы поиска в инфраструктуре портала

Информационный web-портал изначально создавался для аккумуляции научных ресурсов и, что самое важное, предоставления эффективного доступа к ним всем категориям заинтересованных пользователей. Технология доступа к научным ресурсам, которые не были представлены в web-пространстве или даже в каком-либо цифровом виде, обеспечена в существующей реализации портала [10] средствами, которые в данной работе не рассматриваются. На данный момент портальная инфраструктура позволяет создавать достаточно богатый контент и манипулировать различной информацией, а также создавать и управлять её представлением. При этом поисковый сервис (как контекстный, так и атрибутный) по внутренним ресурсам реализован в полном объёме.

Другой пласт представленной на портале информации обеспечивают внешние системы, например созданные ранее web-сайты и базы данных. Для этих данных и требуется совершенствовать поисковые механизмы. Базовой задачей службы поиска портала является включение этих данных в распределённую web-среду портала с целью предоставления единого поискового инструмента, который обеспечивает эффективный информационный поиск с учётом предметной релевантности.

В составе Информационного web-портала служба поиска оформлена как самостоятельная подсистема, реализующая взаимодействие с различными информационными источниками для

выдачи поисковых запросов, получения, преобразования и форматирования совокупных данных с целью предоставления их пользователю через подсистему отображения портала. Естественно, что служба поиска интегрирована с другими важнейшими службами портала: подсистемой безопасности, системой управления контентом, файловым хранилищем.

5. Архитектура и реализация поисковой службы

Служба поиска портала обеспечивает поиск данных по всем подключённым к порталу информационным источникам. Основной подход при выполнении операций поиска заключается в том, что службой поиска предоставляется web-интерфейс для ввода критериев поиска, для формирования поискового запроса и для его передачи системе-владельцу данных, а система-владелец данных должна сама обеспечивать поисковый сервис над своими данными. Служба поиска выступает как получатель массивов найденной информации: полученные данные форматируются, объединяются и, если необходимо, сортируются, после чего выдаются пользователю.

Службой поиска используется виртуальная схема, описывающая структуры данных и расширенная метаданными, предоставленными информационными источниками, содержащими описания способов доступа к информации. Вся совокупность сформированных таким образом метаданных размещается в файлах XML и условно разделяется на три части: *репозиторий типов*, *система реализации типов* и *описание форм*.

Функционально служба поиска реализуется следующими компонентами:

— *шаблоном поисковых форм*; при помощи этого шаблона формируется web-интерфейс для поиска, шаблон поисковых форм представляет собой APSX-страницу, обеспечивающую формирование страницы поиска на основе *репозитория типов* и с использованием *процессора форм*, и позволяет передать заданные пользователем критерии поиска *процессору запросов* для обработки;

— *шаблоном вывода результатов*, представляющим собой APSX-страницу, которая осуществляет форматирование и вывод результатов поиска; этот шаблон получает результирующий набор данных от *процессора запросов*, форматирует и выдаёт результат в виде web-страницы;

— *процессором форм*; этот компонент на основе *описания форм* обеспечивает функциональность для элементов поисковой формы (элементов управления), может использовать *процессор запросов*, например, для построения списков выбора значений или других целей;

— *процессором запросов*; этот компонент осуществляет взаимодействие с адаптерами информационных источников с целью выполнения поискового запроса и формирования общего результата поисковой операции; использует *систему реализации типов* для построения поисковых запросов и их выполнения.

Последовательность выполнения поисковой операции выглядит следующим образом. Шаблон поисковых форм, используя репозиторий типов из виртуальной схемы портала и процессор форм, формирует web-форму для пользователя. Пользователь при помощи элементов управления задаёт критерии поиска. Шаблон поисковых форм при помощи процессора форм строит набор поисковых условий (поисковый запрос) и передаёт его процессору запросов для обработки. Процессор запросов взаимодействует с адаптерами с целью получения данных и консолидирует результаты. Итогом работы процессора запросов является результирующий набор данных, который форматируется и выводится при помощи шаблона вывода результатов.

Далее подробно рассмотрены работа перечисленных компонентов и используемые службой поиска метаописания.

5.1. Репозиторий типов. Для описания данных, по которым могут производиться поисковые операции, используется *репозиторий типов*. Описание типов представляет собой структуру данных, общую для всех информационных источников, которые участвуют в поиске. Репозиторий типов является интерфейсной частью виртуальной схемы портала, которая отображает общую структуру данных, абстрагированную от деталей реализации доступа и преобразования форматов. Он основывается на трёх основных элементах: *тип*, *связь*, *мультиязычное описание*.

Тип представляет собой совокупность атрибутов, которые могут быть представлены в формате XML-Schema. Каждому типу соответствует мультиязычное описание, которое используется для представления пользователю. Так, например, типу, который описывает личность, соответствует описание, содержащее выводимое имя на трёх (и более) языках: «персона» (русский), «person» (английский), «personne» (французский) и т. п. Каждый тип однозначно идентифицируется свойством «name», и, соответственно, существование двух типов с одинаковым свойством «name» невозможно.

Репозиторий типов реализует множественное наследование. Каждый тип может иметь один или несколько базовых типов: наследование означает включение в состав атрибутов данного типа всех атрибутов базовых типов. Если атрибуты базовых типов пересекаются по названиям, результирующий тип будет содержать один унаследованный атрибут с данным именем. Приведём пример описания двух типов с использованием наследования:

```
<Type name="named">
    <Field name="Id" entity="id"/>
    <Field name="Name" entity="name"/>
    <Field name="Description" entity="description"/>
    <Presentation type="list"/>
</Type>
<Type name="file" entity="file">
    <Parent name="named"/>
    <Field name="FSize" entity="size"/>
    <Field name="UploadTime" entity="createtime"/>
    <Field name="MaterialDate" entity="date"/>
    <Field name="FType" entity="filetype"/>
    <Field name="Copyright" entity="copyright"/>
    <Field name="Source" entity="source"/>
</Type>
```

В приведённом фрагменте тип `file` будет содержать атрибуты `Id`, `Name`, `Description` родительского типа и собственные атрибуты `FSize`, `UploadTime`, `MaterialDate`, `FType`, `Copyright` и `Source`.

Мультиязычные описания можно поставить в соответствие любому типу и любому атрибуту типа. Несколько типов или атрибутов могут быть соотнесены одному описанию. Описание определяется атрибутом `entity` в представлении типа или атрибута. Приведём пример описания для типа `file` и некоторых его атрибутов:

```
<Entity name="file">
    <DisplayName lang="ru">Файл</DisplayName>
    <DisplayName lang="en">File</DisplayName>
</Entity>
<Entity name="filetype">
    <DisplayName lang="ru">Тип файла</DisplayName>
    <DisplayName lang="en">File type</DisplayName>
```

```
</Entity>
<Entity name="copyright">
    <DisplayName lang="ru">Авторское право</DisplayName>
    <DisplayName lang="en">Copyright</DisplayName>
</Entity>
```

Репозиторий типов поддерживает определение связей между типами. Связь определяется как набор типов (два или более), участвующих в соединении. Таким образом, можно задать стандартные виды связей: один к одному, один ко многим, многие ко многим. Приведём пример описания связи для типов `file` и `Category`, данная связь имеет вид многие ко многим:

```
<Link name="file_category">
    <Type name="file" occurs="multiple"/>
    <Type name="category" occurs="multiple"/>
</Link>
```

5.2. Система реализации типов. Система реализации типов предназначена для поддержки взаимодействия с информационными источниками. Как уже отмечалось, в контексте поиска это взаимодействие состоит в формировании поисковых запросов к источникам, получении данных и форматировании результатов и выполняется адаптером источника. В виртуальной схеме портала для каждого источника отводится отдельная секция, которая содержит информацию двух видов. Во-первых, в секции источника содержатся все необходимые данные для создания среды взаимодействия с ним. Например, для взаимодействия с реляционной базой данных в эту секцию включается информация для создания сессии (настройки источника ODBC), для web-сервиса — WSDL-описание или URL, по которому его можно получить. Во-вторых, в секции источника содержится описание отображения части типов и связей из репозитория типов на данный источник (метаданные адаптера). В этом отображении присутствуют только те типы и связи, которые реализуются данным источником (его подсхема). При помощи отображения типов реализуются операции над экземплярами типов (под экземплярами в данном контексте понимается структура данных, соответствующая описанию типа).

Взаимодействие с информационным источником строится следующим образом. Процессор запросов генерирует текстовую команду, которая соответствует операции над экземплярами типа, в контексте поиска это операция выборки данных по условиям. Текстовая команда строится с учётом набора условий (`conditions`), набора возвращаемых полей, языка и т. п. Команда может быть как статической, так и динамической — генерироваться каждый раз при выполнении запроса. Команда описывается на языке адаптера информационного источника. Службой поиска не определяются какие-либо ограничения на формат и содержимое команд, за исключением набора зарезервированных слов. После того, как команда сгенерирована, она передаётся адаптеру, который выполняет команду и возвращает результирующий набор данных.

Система реализации типов позволяет определять статические и динамические команды, а также модули исполнения с использованием языков программирования, поддерживающих CLR [14]. Приведём пример описания реализации для типа `file` в источнике данных:

```
<Type name="file">
  <Script name="Search">
     <![CDATA[
     SELECT %fields% FROM FSFiles
     <%if (Conditions != null && Conditions.Count > 0) {
         if (Conditions["category"] != null)%>
         INNER JOIN FSBelongs
                        ON FSFiles.Id = FSBelongs.FileId
     <%}%>
     WHERE %conditions%
     ]]>
  </Script>
</Type>
```

В данном примере реализации типа `file` с использованием динамической команды применён язык программирования C#, фрагменты программного кода заключены в маркеры <% и %>. Команда формируется из строк текста, которые не являются кодом. Код предназначен для конструирования команды из фрагментов текста, размещённым между строками кода. Фрагмент команды `INNER JOIN FSBelongs ON FSFiles.Id = FSBelongs.FileId` будет включён в результирующий текст только при выполнении условия оператора `if (Conditions["category"]!= null)`. Таким образом, соединение с таблицей `FSBelongs` в команде генерируется только при наличии условия поиска в этой таблице.

В коде могут быть использованы переменные `Conditions`, `Fields`, `Lang` и `User`. Переменная `Conditions` представляет со-

бой коллекцию условий. Каждое условие представляется тройкой «поле–значение–операция» и в контексте поисковых задач выражает критерии отбора данных. В переменной `Fields` содержится коллекция имён терминов, которые должны быть возвращены. Параметр `Lang` определяет язык, который используется в процессе поиска, с его помощью реализуются мультиязычные свойства портала. Параметр `User` используется для интеграции с подсистемой безопасности портала. Он представляет собой объект, реализующий интерфейс `IPrincipal`, через который доступна информация о вызывающем пользователе.

В коде текста формируемой команды можно применить предопределённую переменную `_result` строкового типа, которая содержит сформированную часть команды.

Существует также ещё одно средство для конструирования команд — это шаблоны подстановки `%fields%` и `%conditions%`. При генерации текста команды эти идентификаторы будут заменены на строки, которые формируются из коллекций `fields` и `conditions`, соответственно, с применением шаблона подстановки. Шаблон подстановки представляет собой строку, которая отражает способ преобразования массива элементов в строковый вид с использованием разделителей. Например, если коллекция `fields` содержит элементы `id` и `name`, то, применив шаблон подстановки `{ 0[]:*:, }`, получим строку `id, name`. Шаблоны подстановки могут быть определены для источника данных, для реализации типа и для конкретной команды.

После того, как команда сгенерирована, она должна быть передана адаптеру для выполнения и получения результирующего набора данных. Для этих целей служит секция источника, которая называется `Exec`. В этой секции определяется код для манипулирования данными, а именно код, который обрабатывает команду, созданную на предыдущем этапе, и формирует результирующий набор данных. В качестве параметров передаётся сама команда и описанные выше параметры `Fields`, `Conditions`, `Lang`, `User`.

Для некоторых видов информационных источников реализованы встроенные функции обработки команд. К ним относятся источники данных ODBC, OLEDB, реализации для SQL Server, Oracle, XML, а также сервер интеграции и доступа, входящий в состав существующей реализации портала [10].

Приведём пример определения обработки команды в секции «Exec» для базы данных SQL Server:


```
<Exec>
 _result =
    Utils.ExecSqlSource("workstation id=localhost;
                         packet size=4096;
                         integrated security=SSPI;
                         data source=localhost;
                         persist security info=False;
                         initial catalog=MArchive2",
                         Script, Conditions, Lang, User);
</Exec>
```

Таким образом, предложенная архитектура позволяет реализовывать функции поиска практически к любым источникам данных, даже не поддерживающим какой-либо язык запросов.

5.3. Процессор запросов. Реализацию цепочки выполнения запроса производит *процессор запросов*. Описанная выше функциональность реализуется с помощью динамического построения и применения сборок с использованием технологии генерации кодов CodeDom и позднего связывания Reflection.

Как упоминалось выше, метаданные, используемые службой поиска, содержат фрагменты кода, определяющие функциональность при взаимодействии с информационными источниками. Эти фрагменты сосредоточены в двух местах для каждого источника: в описании реализации типа в системе реализации типов и в процедуре исполнения команды `Exec`. Для взаимодействия с информационным источником процессор запросов предварительно компилирует этот код в исполняемый модуль — сборку (assembly) в терминах .Net, а при выполнении запроса вызывает методы объектов сборки по заданным правилам.

Рассмотрим принципы генерации сборок. Для каждого источника данных генерируется своя сборка, содержащая определения классов в соответствии с типами, реализуемыми источником, и предопределённый класс `Global`. Класс `Global` реализует единственный метод `Execute`, который включает код, содержащийся в секции `Exec` источника. Каждый класс, соответствующий реализуемому типу, содержит методы формирования команд. Каждой команде соответствует свой метод. Метод формируется из фрагментов кода и фрагментов команды, описанных в секции `Script` для реализуемого типа. Возвращаемое значение такого метода — строка, которая является сгенерированной командой.

Приведём пример кода метода, сгенерированного по описанию типа `file`:

```
public class file {
  public virtual string
        Search(PortalSearch.PSFieldsEnumeration Fields,
               PortalSearch.PSConditions Conditions,
               string Lang,
               string User) {
    System.Text.StringBuilder _result =
                  new System.Text.StringBuilder();
    _result.Append("\r\n\t\t\t\t\t\tSELECT %fields%
                     FROM FSFiles\r\n\t\t\t\t\t\t");

    if (Conditions != null && Conditions.Count > 0) {
        if (Conditions["file_category"] != null)
           _result.Append("\r\n\t\t\t\t\t\t\t\tINNER
                JOIN FSBelongs ON FSFiles.Id =
                FSBelongs.FileId\r\n\t\t\t\t\t\tWHERE
                %conditions%\r\n\t\t\t\t\t\t");
    }
    return _result.ToString();
  }
}
```

Существует также возможность использования других сборок, в которых могут быть реализованы функции доступа к данным, формирования команд и др. Для этих целей используется секция источника `Assembly`, которая содержит путь к сборке, её полное имя и другие необходимые данные.

После построения сборок становится возможным обслуживание запросов доступа к данным источников. Общая последовательность поисковой операции представлена на схеме (рис. 1).

Процессор запросов в качестве исходных параметров принимает имя типа, по которому производится поиск, набор условий поиска, условия сортировки, набор возвращаемых полей, язык, на котором должны быть выданы результаты, и учётные данные вызывающего пользователя. Далее определяется список информационных источников, которые реализуют данный тип, с помощью системы реализации типов виртуальной схемы. Для каждого типа выполняется операция построения и исполнения команды, результатом которой является набор данных `DataSet`. Результирующие наборы данных по всем участвующим в запросе информационным источникам объединяются в один и производится сортировка, после чего этот результирующий набор данных выдаётся для форматирования и вывода пользователю.

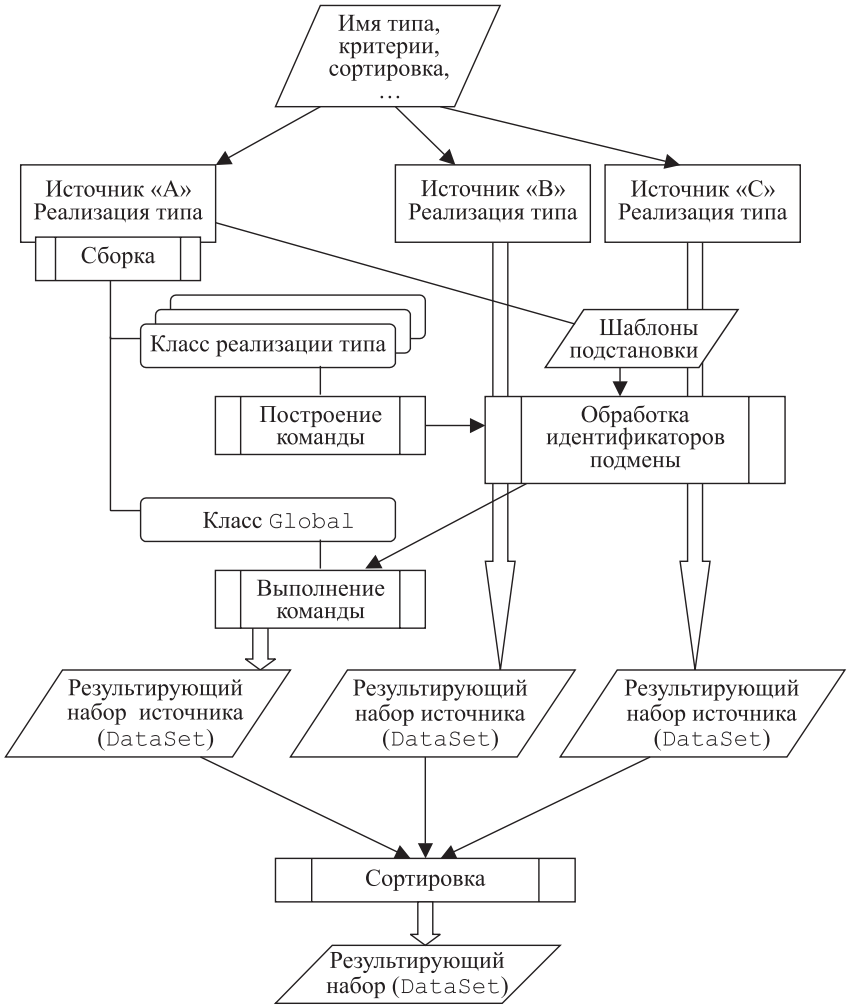

Рис. 1. Последовательность поисковой операции

Рассмотрим подробнее процесс построения и выполнения команды для источника. Как упоминалось выше, каждому типу, реализуемому информационным источником, соответствует сгенерированный класс. Процессор запросов в первую очередь создаёт экземпляр класса, соответствующий типу, и вызывает для построения команды метод, который соответствует операции по-

иска. Этот метод возвращает текст сконструированной команды. Текст команды на данном этапе может содержать идентификаторы подстановки (`%fields%` и `%conditions%`). Используя шаблоны подстановки для коллекций `Fields` и `Conditions`, процессор запросов строит их строковые отображения и осуществляет подмену идентификаторов этими строковыми отображениями. На этом этапе формирование команды завершено.

После формирования команды процессор запросов, используя экземпляр класса `Global` сборки адаптера источника, вызывает метод `Execute`, в который помимо прочих параметров передаётся сгенерированная команда. Метод `Execute` обрабатывает команду и возвращает результат в виде набора данных.

5.4. Шаблон поисковых форм. На основе шаблона поисковых форм строятся интерфейсы для поиска данных различных источников. Шаблон поисковых форм содержит компоненты, единые для всех страниц портала, такие как «шапка» и «подвал», меню и т. п. Содержанием конкретной поисковой формы является набор элементов управления для ввода значений и критериев поиска. Шаблон поисковых форм может реализовывать функциональность web-визарда в случаях, когда по эргономическим соображениям элементы управления нужно разделить на несколько страниц. Также определяется состав и элементы управления для задания критериев поиска (имеются в виду логические операции над всеми введёнными пользователем значениями полей, но не сами значения). Шаблон поисковых форм не определяет состав и типы элементов управления для самих полей.

Каждому полю формы соответствует элемент управления, который позволяет задать поисковое значение для соответствующего атрибута типа, а также условие для проверки введённого значения. Элемент управления может быть любым. Единственным условием применения является то, что задаваемое значение должно преобразовываться в строковый вид. Тривиальным элементом управления для поиска является поле ввода. Для построения элементов управления, их инициализации и получения значений шаблон поисковых форм использует *процессор форм*. При помощи коллекции управляющих объектов строится содержательная часть поискового интерфейса: создаются заголовки, элементы управления и компоненты для проверки значений (валидаторы), после чего готовая поисковая форма выдаётся

пользователю. Данные, введённые пользователем, должны быть обработаны и переданы *процессору запросов*.

После выполнения поискового запроса результат возвращается в виде объекта стандартного типа `DataSet` и выводится *шаблоном вывода результатов* в виде таблицы, которая содержит как некоторые атрибуты найденных ресурсов, так и ссылки на них.

5.5. Процессор форм и описание форм. Задача процессора форм состоит в построении коллекции элементов управления для ввода значений атрибутов, их инициализации, проверке введённых значений и преобразовании их в унифицированный формат. Для отображения элементов управления шаблон поисковых форм получает коллекцию объектов управления полями поиска. Каждый такой объект реализуется процессором форм. Он содержит методы, позволяющие:

— получить заголовок для отображения поля;

— создать элемент управления для поля;

— инициализировать созданный элемент управления, в том числе с использованием запросов к данным;

— получить критерий поиска по полю из значения элемента управления и условия сравнения (>, <, = и т. п.).

Подобно процессору запросов, процессор форм в порядке инициализации компилирует сборку для каждой формы на основе её метаописания. Классы этой сборки реализуют вышеперечисленные функции. Каждая сборка содержит общий класс для формы и по одному классу управления для каждого поля формы. Все классы управления по умолчанию наследуются от встроенного класса `PSControlManager`, у которого определены три виртуальные функции: `CreateControl`, `GetValue` и `DataBinding`. Класс `PSControlManager` реализует при помощи этих функций работу с элементом управления `TextBox`. В процессор форм встроены также ещё несколько классов для работы с другими элементами управления, такими как `ListBox`, `DropDownList` и т. д.

Для задания собственной функциональности взаимодействия с элементом управления посредством методов `CreateControl`, `GetValue` и `DataBinding` можно переопределить методы базового класса, поместив код метода в соответствующую секцию метаописания поля формы. По умолчанию используемым классом управления является класс `PSControlManager`, но можно исполь-

зовать и другой встроенный базовый класс, указав его псевдоним в теге описания поля атрибутом `controlpattern`.

Приведём пример описания формы, который показывает, как простое описание поля формы, так и описание с переопределением методов:

```
<Form name="ArchiveSearch" type="act" >
  <DisplayName lang="ru">Поиск по архивам</DisplayName>
  <DisplayName lang="ru">Archive search</DisplayName>

  <Field name="Name"/>
  <Field name="Inventory">
    <CreateControl
      controltype="System.Web.UI.WebControls.DropDownList,
      System.Web,
      Version=2.0.0.0,
      Culture=neutral,
      PublicKeyToken=b03f5f7f11d50a3a"/>
    <GetValue valuefieldprop="Text">
      _result = new PSCondition(FieldInfo.Name, "=",
            (DropDownList)ControlObj.SelectedItem.Value);
    </GetValue>
    <DataBinding sourcetype="link" sourcename="Inventory"
                 textcolumn="Name" datacolumn="Id"/>
  </Field>
  <Source name="MArchive"/>
  <Source name="VArchive"/>
</Form>
```

В данном примере поле с именем `Name` описано в простом виде. В этом случае для реализации класса управления будет использован предопределённый класс `PSControlManager` и для него будет сгенерирован элемент управления `TextBox`. Поле `Inventory` описано таким образом, что будет сгенерирован класс-наследник, который оперирует с элементом управления `DropDownList` и переопределяет метод `GetValue` базового класса. Также в приведённом описании формы видны источники, по которым будет производиться поиск. Если источники не заданы, поиск производится по тем источникам, где присутствует реализация класса `Act`.

После инициализации и построения сборок процессор форм используется шаблоном поисковых форм для создания интерфейса, позволяющего задавать поисковые критерии и инициализировать элементы управления.

5.6. Шаблон вывода результатов. Шаблон вывода результатов предназначен для формирования web-представления результирующего набора данных. После выполнения запроса процессором форм результирующий набор данных представляется объектом `DataSet`. Для форматирования данных, содержащихся в наборе, шаблон поисковых форм использует метаописания поисковой формы, содержащиеся в секции `output`. На их основе строится результирующая таблица, в которой каждое поле форматируется в соответствии с заданными правилами, а вся таблица разбивается на страницы.

5.7. Средства администрирования. Вся необходимая информация, используемая службой поиска, физически размещается в файлах XML. Несмотря на то, что эти файлы могут быть сформированы и отредактированы вручную при помощи любого текстового редактора, средства администрирования службы поиска существенно облегчают управление службой, позволяя пользователю не вдаваться в детали реализации виртуальной схемы.

Средства администрирования представляют набор пользовательских интерфейсов, которые позволяют:

— управлять репозиторием типов;

— выполнять подключение информационных источников и управлять настройками доступа к ним (настраивать адаптеры);

— создавать и редактировать формы атрибутного поиска;

— создавать и проверять фрагменты кода для нужд службы поиска на разных языках платформы .Net.

6. Заключение

В настоящей статье рассмотрены основные принципы и архитектура одного из важнейших компонентов Информационного web-портала — программной системы, обеспечивающей официальное представление Российской академии наук в сети Интернет. Реализация описанной службы поиска в составе действующего академического портала обеспечила расширение его функциональности в части наиболее значимых потребностей пользовательской аудитории и позволила расширить спектр ресурсов, имеющих перспективу интеграции в Единое научное информационное пространство РАН.

Список литературы

1. Deep Web — Bright Planet's white paper. — http://www.completeplanet.com/tutorials/deepweb/.
2. European Research Gateways Online. — http://www.cordis.lu/ergo.
3. *Laitinen S.*, *Sutela P.*, *Tirronen K.* Development of Current Research Information Systems in Finland // Proc. of CRIS-2000, 22–26 May, Acicastello, 2000.
4. *Соколов И.А.*, *Босов А.В.*, *Бездушный А.Н.* О Информационном web-портале Российской академии наук // Системы и средства информатики. Вып. 13. — М.: Наука, 2003. — С. 119–138.
5. Россия. Веб-портал РАН обеспечивает эффективный доступ научного сообщества к актуальной информации // Информационный бюллетень Microsoft. 2004. Вып. 26. С. 67–68.
6. *Некрестьянов И.С.*, *Пантелеева Н.* Системы текстового поиска для Веб // Программирование. № 4. 2002. С. 33–57.
7. *Гаскаров Д.В.* Интеллектуальные информационные системы. — М.: Высшая школа, 2003.
8. *Гринберг И.*, *Гарбер Л.* Разработка новых технологий информационного поиска // Открытые системы. 1999. № 9–10.
9. *Жигалов В.* Как нам обустроить поиск в Сети? // Открытые системы. 2000. № 2. http://www.osp.ru/os/2000/12/053.htm.
10. *Босов А.В.*, *Иванов А.В.*, *Полухин А.Н.*, *Чавтараев Р.Б.* Управление сайтом информационного web-портала // Системы и средства информатики. Вып. 15. — М.: Наука, 2005. — С. 233–259.
11. XML Core Working Group Public Page. — http://www.w3.org/XML/Core.
12. Web Services Activity. — http://www.w3.org/2002/ws/#drafts.
13. XML Schema. — http://www.w3.org/XML/Schema#dev.
14. *Рихтер Дж.* Программирование на платформе Microsoft .NET Framework. Мастер класс. 3-е изд. / Пер. с англ. — М.: Русская редакция, 2005; — СПб.: Питер, 2005.