Технология семантической интеграции баз данных в системной биологии

Д. С. Мигинский
Институт цитологии и генетики СО РАН, Новосибирск, Россия
e-mail: shadow@bionet.nsc.ru

В. В. Лабужский
Институт систем информатики СО РАН, Новосибирск, Россия

М. М. Лаврентьев-мл.
Институт математики им С.Л. Соболева СО РАН, Новосибирск, Россия

А. В. Морозов
Институт цитологии и генетики СО РАН, Новосибирск, Россия

С. А. Соколов
Trafalgar Asset Managers, London, UK

The amount of data on biological and especially molecular-genetic systems is continuously growing. A large number of separate databases, different representation formats, and mistakes in initial data make modeling of the biosystems and analysis of such models essentially complicated. A technology for the semantic data integration is suggested for solving this problem. It is capable to: 1) reveal complementary data elements, index and integrate data by taking into account this information; 2) search for data in all sources integrated in structured way; 3) reveal the gaps, contradictions and mistakes in some cases. A prototype of software system implementing this technology is also described in this paper.

Введение

В настоящее время активно развивается множество информационных ресурсов в области биоинформатики, в первую очередь — базы данных, связанные с описанием отдельных элементов молекулярно-генетических систем и процессов (гены, белки, их функции и структура и т. д.). Развитие этих баз обусловлено необходимостью реконструкции прежде всего целостных моделей живых систем, а не только частных аспектов их функционирования, как это происходило до недавнего времени. Модель простейшей одноклеточной живой системы, описанная в таких терминах, может содержать десятки тысяч элементов и взаимосвязей, что существенно затрудняет их реконструкцию. Развитие баз данных позволяет облегчить решение задачи, однако ряд проблем этому препятствует. Объем информации, который хранится в наиболее развитых базах, существенно превышает возможности человеческого восприятия (в самой авторитетной базе, хранящей информацию о генах, — EntrezGene [1, 2] — содержится более 10 Гбайт полезной

информации в XML-представлении). Информация в такие базы вносится преимущественно вручную, поэтому неизбежно накапливаются неточности, ошибки и пробелы. Проблема усугубляется тем, что предметная область на текущий момент недостаточно строго формализована, вследствие чего каждая база достаточно узко специализирована в соответствии с областью деятельности коллектива, ее развивающего. В результате, при реконструкции более или менее полной модели живой системы эксперт сталкивается с необходимостью работы со многими базами.

В данной статье предлагается подход, позволяющий создать единое информационное пространство, содержащее информацию из различных баз по данной тематике. Такое информационное пространство можно рассматривать как единую "виртуальную" базу. Под виртуальностью здесь понимается то, что пользователь работает с данной программной системой, как с обыкновенной базой, но при этом данные могут храниться и в ней, и во внешних источниках. Предлагаемый подход был реализован в виде прототипа программной системы, реализующей основные функции, необходимые для создания такого информационного пространства.

Предлагаемая программная система — по сути средство автоматизированной поддержки научных исследований. В первую очередь она обеспечивает поддержку решения задач, связанных с реконструкцией и анализом структурно-функциональной организации живых систем.

Программная система реализует следующие основные функции:

1) установление эквивалентности объектов из различных источников с учетом синонимии;

2) анализ противоречий, ошибок, пробелов в информации;

3) определение уровней доверия отдельным информационным источникам;

4) структурированный прозрачный поиск информации во всем информационном пространстве с учетом пунктов 1–3.

Статья построена следующим образом. В разделе "Характеристика предметной области" рассматриваются особенности существующих баз данных в области системной биологии. Раздел "Анализ существующих решений" посвящен обзору подходов к интеграции данных и соответствующих программных средств. Далее следует описание предлагаемой методики интеграции ("Принципы интеграции"). В разделе "Основные архитектурные элементы" предлагается архитектура программной системы, реализующей методику, и рассматриваются ее наиболее важные элементы. И, наконец, освещаются некоторые технические стороны реализации.

1. Характеристика предметной области

Работа направлена в первую очередь на упрощение поиска в больших базах достоверной первичной информации, используемой для реконструкции моделей живых систем на молекулярно-генетическом уровне. Основными элементами таких моделей являются гены, белки, низкомолекулярные соединения, а также более крупные объекты, как рибосомы, ядра клеток, клеточные мембраны и др. Между элементами выделяются различного вида взаимодействия: химические реакции, стадии экспрессии генов, процессы переноса, деградации, а также регуляторные воздействия, влияющие на скорость протекания других взаимодействий. Модель даже простейшей бактериальной клетки может включать в себя десятки тысяч элементов и взаимодействий.

Есть несколько способов получения информации, необходимой для реконструкции таких моделей. Наиболее прямой и эффективный способ — это постановка серии собственных экспериментов. Однако эксперименты весьма дороги и занимают много времени; такой путь зачастую нерентабелен, и далеко не каждый коллектив может себе это позволить. Второй способ — сбор опубликованных экспериментальных данных, полученных сторонними организациями. Этот способ обладает рядом недостатков: данные не всегда достоверны, разрознены, представлены в разных форматах, часто в них присутствуют пробелы. Сбор таких данных весьма трудоемок, особенно если они опубликованы в виде статей. Если предположить, что каждая статья в среднем содержит информацию о четырех элементах и эксперт в день может найти и обработать пять статей, то цена реконструкции модели из десяти тысяч элементов составляет более полутора человеколет.

Тем не менее способ, связанный с поиском уже доступной информации, обладает двумя несомненными преимуществами, которые в большинстве случаев заставляют использовать именно его. В первую очередь этот способ, несмотря на трудоемкость поиска, все же более рентабелен, так как не требует использования дорогого экспериментального оборудования. Кроме того, объем информации такого рода, доступной даже из открытых источников, значительно превышает возможности отдельно взятой лаборатории по ее получению экспериментальным способом. Базы данных по этой тематике существенно облегчают поиск информации (хотя в них также встречаются и недостоверные данные, и пробелы в информации).

На текущий момент известно достаточно много общедоступных баз данных по указанной тематике, более или менее узкоспециализированных. С точки зрения интеграции рассматриваются в первую очередь хорошо структурированные базы, такие как EntrezGene [1, 2], KEGG [3, 4], dbSNP [5, 6] и некоторые другие. Существует также ряд полнотекстовых баз, где в лучшем случае у каждого объекта есть заголовок, далее идет текстовое описание. Для таких баз требуется разработка дополнительных методов экстракции информации в структурированном виде либо ручное аннотирование. В данной статье эти методы не рассматриваются.

В целом базы данных, связанные с молекулярно-генетическими системами и обладающие достаточной степенью структуризации, можно охарактеризовать следующим образом:

— базы имеют достаточно сложную структуру. Каждая сущность в реляционной модели может быть представлена десятком и даже более таблиц. Кроме того, при развитии базы, как правило, постепенно уточняется и расширяется ее схема. Все это делает применение стандартного реляционного подхода затруднительным;

— типичный объем базы — несколько гигабайт в XML-представлении. С учетом сложной структуры каждой из базы и, как следствие, сложных запросов объем можно считать весьма существенным;

— с точки зрения семантики все базы имеют некоторый набор базовых сущностей (хотя синтаксически они могут быть представлены по-разному), но их детализации существенно различаются (от сложно структурированного документа в десятки килобайт в XML-представлении до простой URL-ссылки на другие источники). Кроме того, практически каждая из них определяет свои дополнительные сущности.

Для того чтобы упростить поиск первичной информации, требуется разработка программной системы семантической интеграции данных из различных источников, в первую очередь баз данных. Система должна обеспечивать единую точку доступа ко

всем интегрированным данным, т. е. пользовательский интерфейс, предоставляющий возможности поиска и просмотра данных. Система также должна учитывать специфику баз, в частности гетерогенность данных и форматов их представления, присутствие ошибок и пробелов в данных. Разработка такой системы позволит существенно снизить трудоемкость реконструкции сетевых моделей биологических систем, повышая эффективность исследований в целом.

2. Анализ существующих решений

Исходя из потребностей конечного пользователя, т. е. специалиста в области системной биологии, первичной задачей можно считать не интеграцию данных, а получение централизованного доступа к разрозненным информационным ресурсам. В зависимости от способа реализации этого требования, т. е. от того, происходит ли интеграция данных и если происходит, то каким образом, пользователь получает различные возможности по выполнению запросов. В первую очередь от способа интеграции зависят сложность возможных запросов и время их выполнения. Рассмотрим более подробно существующие подходы к решению этой задачи.

1. Наиболее очевидный и простой способ — консолидация различных информационных ресурсов на уровне единого пользовательского интерфейса. При этом рассматриваемые ресурсы не связываются семантически. Преимуществами такого подхода являются высокая масштабируемость и сравнительно умеренная цена реализации подобной системы и подключения дополнительных ресурсов, особенно с применением современных GRID-технологий и поисковых машин, таких как Google или Яndex.

Существенный недостаток такого подхода — невозможность построения структурированных запросов, а только полнотекстовых. Это ограничение является следствием того, что каждый ресурс, вообще говоря, имеет свою семантику, и семантическое сопоставление ресурсов не проводится. Также не сопоставляются форматы представления данных. В результате:

1) пользователь не имеет возможности строить запросы достаточно точно, что повышает трудозатраты на анализ найденной информации;

2) практически невозможно использование таких систем интеграции в связке с программными средствами анализа, так как формат представления результатов запроса определяется тем ресурсом, в котором они найдены, т. е. заранее неизвестен.

Один из наиболее авторитетных ресурсов такого рода в области системной биологии — сайт National Institute for Biotechnology Information (NCBI) [7], позволяющий проводить поиск и навигацию по нескольким десяткам тематических информационных ресурсов — как по отдельности, так и по всем ресурсам вместе.

2. Второй способ подразумевает интеграцию данных на уровне внешних ссылок на другие источники. Он реализован практически во всех популярных биологических базах, в частности в большинстве баз, доступных через сайт NCBI, а также KEGG, UniProt и т. д. Способ заключается в том, что объекты в базе получают дополнительные поля, хранящие гиперссылки (либо информацию, достаточную для их формирования) на объекты в других базах. С учетом того что большинство баз имеют web-интерфейс, пользователь получает возможность навигации между базами.

Преимущество данного подхода — практически нулевые затраты с точки зрения программной реализации. Но при этом такие ссылки, как правило, строятся вручную

при аннотировании базы. Существенным недостатком также является невозможность обращения по таким ссылкам в запросах.

3. Третий способ существенно более трудоемок с точки зрения программной реализации, однако обладает и значительными преимуществами. Суть его в том, что запрос строится к общему виртуальному представлению (термин использован по аналогии с принятым в реляционных базах понятием view — представление) всех рассматриваемых баз, после чего происходит трансляция запроса для каждой из них в отдельности. Общее представление строится как унифицированная схема данных, которая тем или иным образом отображается на схемы всех интегрируемых баз. Схема общего представления отражает модель предметной области (онтологию), в рамках которой существуют рассматриваемые базы. Форматы представления данных и их схем могут быть различны — сопоставление необходимо только на семантическом уровне.

Данный подход позволяет, в отличие от двух предыдущих, строить структурированные запросы (т. е. учитывающие семантику данных) в соответствии с общей схемой. Далее для каждой базы разрабатывается дополнительный модуль (часто называемый драйвером), транслирующий запрос на язык, поддерживаемый конкретной базой.

Примерами систем такого рода являются K2/Kleisli [8], Biomediator [9], TAMBIS [10]. Различаются они в основном подходами к реализации драйверов и описанию общего представления. Например, в K2/Kleisli для этого используется специальный язык, базирующийся на ODL и OQL, в свою очередь входящих в стандарт для объектно-ориентированных СУБД, разработанный ODMG [11]. TAMBIS позволяет конструировать собственное представление любому пользователю с помощью специализированного графического интерфейса.

4. Последний подход в англоязычной литературе известен как data warehousing и применяется в первую очередь в различных бизнес-приложениях. Он во многом похож на предыдущий. В частности, интеграция также осуществляется на основе общей схемы данных, поддерживаются структурированные запросы. Ключевое отличие в том, что происходит предварительная интеграция данных всех баз. В зависимости от задачи, либо интеграция заключается в индексировании части полей из каждой базы, либо осуществляется полная интеграция всех данных в соответствии с общей схемой.

Наиболее распространенной программной системой биологической направленности, основанной на данном подходе, является Sequence Retrieval System (SRS) [12]. Исходно она позиционировалась не как система интеграции, а как поисковая машина для структурированного поиска по данным, представленным в разных форматах. Использование SRS подразумевает создание драйвера для каждого информационного ресурса, обеспечивающего разбор и индексирование необходимых для поиска полей. После этого запросы строятся на основе индекса и выполняются без обращения к исходным данным. Таким образом, если две базы имеют индексы с одинаковой структурой, их можно рассматривать как интегрированные между собой. На текущий момент под SRS разработаны драйверы для нескольких сотен биологических баз, имеется достаточно много инсталляций, в том числе и общедоступных, по всему миру.

Из существенных недостатков данной системы стоит отметить ограниченное использование ссылок при построении запросов, что в свою очередь ограничивает поисковые возможности системы. Это — лимитирующий фактор ее использования, если речь идет о реконструкции сетевых моделей биологических систем и снижении трудоемкости процесса путем частичной автоматизации.

Для решения поставленной в данной работе задачи первые два подхода неприемлемы, так как не поддерживают структурированных запросов. Рассмотрим более подробно оставшиеся.

Подход, связанный с трансляцией запросов “на лету”, по сравнению с предварительным индексированием данных обладает следующими преимуществами:

— все изменения в интегрированных базах сразу становятся доступны для пользователя системы интеграции;

— всегда известно происхождение тех или иных данных (например, для контроля степени доверия к различным источникам данных);

— развертывание такой программной системы не предъявляет специальных требований к аппаратным средствам (не требуются большие вычислительные мощности, дисковое пространство и т. д.).

С другой стороны, предварительное индексирование обладает следующими положительными качествами:

— при выполнении запросов обеспечивает более высокую производительность, не зависящую от качества сетевого канала и загруженности внешних информационных ресурсов (а также их доступности в момент выполнения запроса);

— позволяет проводить масштабный анализ данных с применением методов data mining, сетевого и статистического анализа и т. д.;

— при некоторой доработке программной части такая система может использоваться как высокоуровневая проблемно-ориентированная СУБД, что весьма актуально с учетом существующей динамики наполнения биологических баз и разработки новых;

— далеко не все существующие биологические базы данных поддерживают развернутые языки запросов. Часто доступ ограничивается полнотекстовым поиском или вообще база предоставляется в виде набора файлов без каких-либо программных средств (что, строго говоря, не позволяет ее называть базой данных). Подход, связанный с предварительным индексированием, не требует такой функциональности от интегрируемой базы.

В результате был выбран подход, связанный с предварительным индексированием данных. Выбор основан на следующих выводах, полученных из анализа перечисленных преимуществ и недостатков:

1) проблема контроля происхождения данных разрешима при их предварительном индексировании, и в данной статье предлагается ее решение;

2) на текущий момент проблема доступа к самым свежим данным не настолько актуальна в области системной биологии, как систематизация уже накопленных данных;

3) проблемы с производительностью и доступом к информационным ресурсам, не имеющим собственных поисковых механизмов, принципиально неразрешимы в рамках третьего подхода.

3. Принципы интеграции

Модель предметной области, определяющая структуру хранимой информации, — ключевой элемент в любой информационной системе. При попытках интеграции нескольких различных баз данных встает вопрос, в каком виде будет представлена информация, хранимая в общей информационном пространстве. Рассмотрим этот вопрос более подробно.

В первую очередь заметим, что потенциальный набор сущностей, которыми приходится оперировать, достаточно обширен, начиная с различных химических веществ, белков и других субмолекулярных объектов и заканчивая, потенциально, целыми экосистемами, с участием многих видов. Последнее особенно актуально, так как задача состоит не в удовлетворении сиюминутных потребностей специалистов в области системной биологии, а в разработке таких программных средств, которые могли бы развиваться вместе с развитием самой биологии. В частности, вопрос моделирования крупномасштабных систем может стать актуальным буквально в ближайшие годы.

Существует много различных аспектов применения этой информации. Ген может быть, с одной стороны, рассмотрен как атомарная сущность, функцией которой является "производство" белка, с другой стороны, это последовательность ДНК, имеющая сложную внутреннюю структуру, состоящую из кодирующих и не кодирующих районов, функциональных сайтов и т. д. Как правило, каждая база данных рассматривает информацию о гене только с одной точки зрения, и при комплексном анализе возникают проблемы сопоставления данных.

В настоящее время существует достаточно много моделей представления биологических данных. Во-первых, каждая база данных имеет такую модель, но, как уже говорилась, они достаточно узконаправленны. Во-вторых, существуют более или менее полные модели, охватывающие многие аспекты деятельности биологических систем, например Sigmoid [13], Gene Ontology [14]. Последняя, по всей видимости, — наиболее авторитетная база знаний в этой области. Однако все такие модели либо ориентированы на пользователя, и без дополнительной обработки и разработки программных средств не применимы для компьютерного анализа, либо слишком сложны — количество классов или интерфейсов, представляющих биологические сущности, может измеряться сотнями. Представление каждой сущности в виде отдельного класса в программном интерфейсе приводит к жесткой архитектуре, сложности поиска и устранения ошибок, в том числе и концептуальных. И что самое важное: такое представление приводит к сложности изменения модели предметной области, что является вполне закономерным процессом, связанным с развитием самой науки.

Исходя из вышесказанного, можно сформулировать следующие основополагающие требования и принципы, касающиеся разработки системы интеграции баз данных в системной биологии.

— Разрабатываемая система должна быть базой данных, имеющей гибкую схему. Необходимость этого обусловлена, во-первых, изменением схем интегрируемых баз в процессе их развития; во-вторых, интеграцией новых баз; в-третьих, уточнением набора понятий самой предметной области в соответствии с проводимыми исследованиями. Известно, что принципы проектирования реляционных баз основаны на предположении, что предметная область четко определена, следовательно, схема базы данных статична. С учетом того что большинство реляционных и постреляционных СУБД оптимизировано в соответствии с этими принципами (операция ALTER TABLE на работающей заполненной базе может привести к весьма неприятным последствиям), нарушение их приведет как минимум к существенной потере производительности. Противоречие можно разрешить, сформулировав дополнительный принцип: *если предметная область может изменяться, то схема базы данных не должна включать сущности, ее составляющие. Описание предметной области должно храниться как данные в базе.*

— Система должна позволять проводить структурированный поиск по всем интегрированным ресурсам. "Структурированный" означает, что критерии запроса соответ-

ствуют структуре и модели предметной области. Например, белок можно искать по собственному названию, названию организма, гену, результатом экспрессии которого является, или по любой их комбинации.

При интеграции баз реализовать функции структурированного поиска можно двумя способами.

Способ 1. Для каждой новой базы разрабатываются специализированные конверторы запросов, которые приводят запрос в соответствие ее интерфейсу. Если в систему интегрировано N баз, то запрос пользователя будет преобразован в N запросов и передан каждой из них, далее, через некоторое время (в зависимости от производительности баз и качества связи с каждой из них), пользователь получит набор ссылок на найденные объекты из разных баз, аналогичных результатам работы любой системы web-поиска. При этом сопоставление результатов поиска, анализ противоречий и другие задачи ложатся целиком на пользователя.

Способ 2. Более сложный в проектировании и реализации, но и более эффективный следующий подход. При интеграции базы индексируется вся информация, которая в ней содержится, строится индекс, содержащий достаточное количество данных для выполнения запросов. В дальнейшем, по мере развития базы, эта индекс может инкрементально пополняться. Запрос происходит только в рамках индекса, без обращений к внешним базам. Результаты поиска отображаются в виде аннотаций объектов (в том объеме, в котором информация об объектах хранится в индексе) плюс в виде тех же самых ссылок на внешние источники, но уже сгруппированных по объектам. Другими словами, один и тот же ген, найденный в двух разных базах, будет представлен как один объект с двумя внешними источниками. Кроме того, при соответствующей структуре индекса и описания предметной области возможна идентификация противоречивой информации, ошибок, пропусков и т. д.

Достоинства такого подхода — несомненные преимущества для пользователя как в скорости, так и в функциональности, и сравнительная простота интеграции новых баз. Последнее обусловлено тем, что не требуется дополнительных программ для преобразования запросов (которые могут быть весьма сложными), более того, внешняя база данных может даже не предоставлять полноценного поискового интерфейса (что, строго говоря, даже не позволяет называть ее базой данных). Недостатками являются существенно более высокая сложность “ядра” такой программной системы (включающего, в частности, поддержку индексирования) и дополнительные требования к внешним базам — возможность получения полной информации из них. Как показало изучение основных баз в рассматриваемой области, такая возможность доступна для всех них, более того, как правило, данные представляются в удобном для обработки XML-формате. Последнее позволяет свести процесс интеграции новой базы к разработке нескольких XSLT-преобразований.

4. Основные архитектурные элементы

4.1. Метамодель

При разработке программных средств, решающих поставленную задачу, был выбран второй способ как более перспективный в плане функциональности и дальнейшего развития самих программных средств и методов интеграции. Ключевым компонентом в этом случае является предлагаемая авторами база данных MetaBase, хранящая индекс-

ную информацию и определяющая архитектуру и потенциальные возможности всей программной системы.

Для представления данных был выбран объектно-ориентированный (ОО) подход как более гибкий в сравнении с реляционным. Отметим, что это не препятствует использованию традиционных СУБД. Существуют достаточно развитые и эффективные средства объектно-реляционного отображения, как встроенные в СУБД (Oracle, Intersystems Caché [15]), так и внешние (Hibernate).

В соответствии с этим подходом, а также сформулированными ранее принципами, сущности, составляющие схему базы MetaBase, разделяются на следующие уровни (рис. 1).

Информация о предметной области. В терминологии ОО этот уровень хранит информацию о классах, или метаинформацию, определяя основные понятия (ген, оперон, реакция и др.), а также их атрибуты, связи и ограничения.

Индексная информация. В терминах ОО это объекты, т. е. экземпляры классов или понятий, определенных на вышележащем уровне. Являясь экземпляром понятия, объект подчиняется всем ограничениям, наложенным на класс, т. е. имеет вполне определенные атрибуты и связи с объектами других классов.

Источники. Каждый элемент данных, хранящийся в базе, может быть подписан тем источником, из которого он взят, например внешней базой данных. Таким же образом представляется самая ценная информация, хранящаяся в базе, — ссылки на исходные данные во внешних базах (как правило — http://).

Структура базы не привязана к конкретной предметной области. Это позволяет использовать ее применительно к разным областям знаний, в том числе не биологическим. Настройка базы данных под предметную область происходит без изменения ее схемы, посредством сохранения формализованной модели рассматриваемой области на одном из уровней базы. Такой подход дает возможность пользователю проводить настройку даже без участия программиста, что невозможно при применении классических подходов к проектированию баз.

Процесс интеграции нового источника (базы данных) с учетом описанной метамодели выглядит следующим образом (рис. 2).

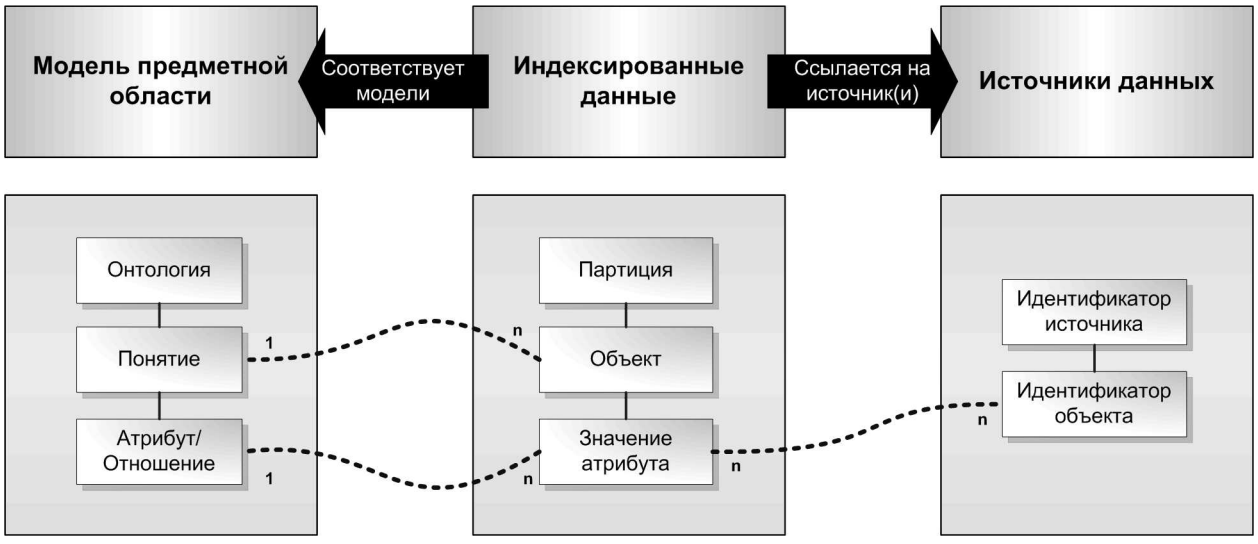

Рис. 1. Основные элементы схемы базы MetaBase

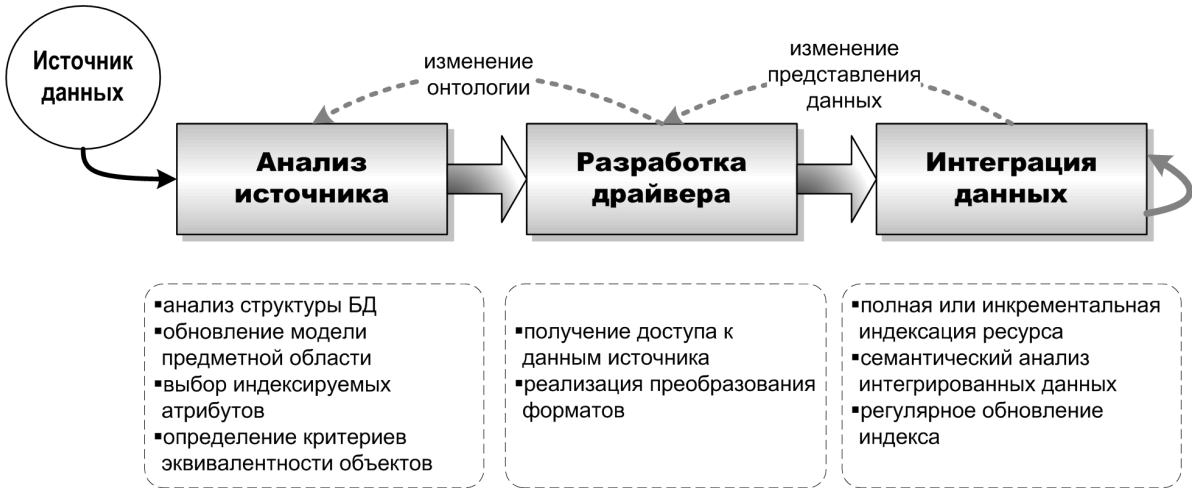

Рис. 2. Процесс интеграции базы данных

1. Анализируется структура интегрируемой базы, в первую очередь с точки зрения семантики. Если ее отличие от модели предметной области, хранящейся в MetaBase, заключается только в другом формате представления (другие имена атрибутов, понятий и т. д.), то ее изменения не требуются. В противном случае в модель предметной области вносятся необходимые изменения (как правило, расширяется набор понятий и/или атрибутов), после согласования со специалистами (в данном случае — в области молекулярно-генетических систем). Отметим, что в модели предметной области не должны быть отражены абсолютно все элементы схемы интегрируемой базы (хотя в случае полностью структурированной базы это возможно). Модель должна содержать только те атрибуты, которые необходимы для поиска. Именно по этой причине MetaBase рассматривается не как база, хранящая полную интегрированную информацию из внешних источников, но как индексная база.

2. Разрабатывается драйвер интегрируемой базы. С учетом выбранного механизма интеграции основная задача драйвера — преобразование данных из формата интегрируемой базы во внутренний формат MetaBase (с учетом того, что на предыдущем шаге было достигнуто семантическое соответствие форматов). Все остальные функции, в частности построение запросов, элементы семантического анализа данных (поиск синонимов, противоречий и пробелов) берет на себя система. В результате разработка драйвера существенно упрощается, и дополнительные усилия, потраченные на разработку ядра (в сравнении с первым способом интеграции), окупаются уже при интеграции трех-четырех внешних баз.

3. Запускается процесс автоматической интеграции (предполагается, что драйвер имеет доступ к полным данным из интегрируемой базы), состоящий из двух основных фаз:

1) преобразования во внутренний формат драйвером;

2) собственно интеграция с данными, уже хранящимися в базе MetaBase. Если база данных уже интегрирована, но информация в ней обновилась, возможна инкрементальная интеграция, учитывающая только изменения и занимающая меньше времени.

В последующих разделах более подробно описаны назначение уровней базы данных, а также процесс интеграции данных и функциональность, предоставляемая пользователю.

4.2. Модель предметной области

В рамках предлагаемого подхода модель предметной области представляется как обычные данные. Эта модель, наряду с информацией о биологических объектах, хранится в базе MetaBase. СУБД, на основе которой разрабатывается данная программная система, определяет ряд сущностей (таблиц, классов), предназначенных для хранения такого рода данных. Рассмотрим более подробно некоторые аспекты и технические решения, связанные с возможностью модификации структуры данных, т. е. модели предметной области.

Как было показано ранее, описание предметной области может быть сведено к двум ключевым сущностям в базе: понятие и атрибут. Атрибуты могут иметь разные типы, в частности ссылочные, кроме того, могут вводиться дополнительные ограничения. Набор их типов и ограничений повторяет классическую объектно-ориентированную метамодель, которую можно найти в любом руководстве по языку UML, поэтому мы подробно на этом не останавливаемся. Для удобства использования системы поддерживается возможность представления этой информации в виде файла в формате XML Schema.

Если данных в базе нет, то любая модификация предметной области никаких проблем не вызывает. Если же база данных содержит информацию, то в общем случае модификация предметной области способна привести к повреждению данных, что абсолютно неприемлемо. Выделим несколько видов элементарных модификаций.

— Создание нового понятия или атрибута. Не приводит к повреждению информации.

— Удаление понятия или атрибута. Наиболее сложный случай, когда все значения атрибута или все объекты данного типа (понятия) должны быть удалены. Это в свою очередь может повлечь необратимые последствия, если, например, понятие было удалено ошибочно.

— Переименование понятия или атрибута или любые другие модификации, также приводящие к возникновению сложных ситуаций. Для упрощения архитектуры и минимизации потенциальных источников ошибок все подобные операции сводятся к удалению старого и созданию нового элемента (атрибута, понятия). Например, переименование состоит из следующих этапов:

1) создается новое понятие или атрибут, в который копируются все данные, возможно, с некоторым преобразованием;

2) удаляется атрибут со старым названием по всем правилам удаления.

Для разрешения проблемы в случае удаления следует руководствоваться принципом “не навредить”. Удаленный атрибут на самом деле не удаляется, но помечается как не рекомендуемый к использованию. Не удаляется и вся информация, с ним связанная. При разработке внешних программных компонентов, использующих систему, поля, помеченные таким образом, использовать не рекомендуется. Пользователи и разработчики уже существующих компонентов будут иметь доступ к старому полю и возможность приспособиться к новому представлению информации. Решение об окончательном удалении поля будет приниматься в случае полной уверенности, что старым представлением никто не пользуется. С удалением понятия решение проблемы аналогично. Отметим также, что ситуация, в которой необходимо удалить элемент модели предметной области, возникает достаточно редко, при условии, что эта модель тщательно разрабатывалась и верифицировалась, с учетом мнений различных специалистов. Как правило, наиболее типичная ситуация — расширение этой модели.

4.3. Индексная информация

Под индексной информацией понимаются объекты, являющиеся экземплярами понятий, описанных в модели предметной области. Такие объекты хранят часть информации об элементах биологических систем в структурированном виде. Назначение этой информации — в первую очередь осуществление поиска, а не полноценное описание элемента. Именно поэтому мы и используем термин "индексная информация". В то же время хранение полного описания элемента не противоречит принципам организации системы (при условии, что информация может быть полностью структурирована). Другими словами, при необходимости система может быть также использована как полноценная проблемно-ориентированная высокоуровневая СУБД.

Если рассматривать систему как СУБД, а модель предметной области — как схему базы данных, то можно увидеть много общего с обычными реляционными базами. Однако существует ряд принципиальных отличий. Наиболее существенным из них является то, что каждый атрибут объекта может иметь несколько значений. В реляционной же модели постулируется, что каждый кортеж (строка — аналог объекта) отношения (таблицы) состоит из фиксированного числа элементов. Многозначность значений атрибутов, с которой приходится сталкиваться при работе с биологическими объектами, возникает из проблемы поддержки синонимии. Например, каждый биологический вид помимо канонического латинского названия может иметь еще ряд используемых (homo sapiens — human, mus musculus — mouse). Такая же ситуация наблюдается с наименованиями генов, белков и т. д. Многозначность проявляется и в некоторых других случаях. Например, химические реакции (направленные) характеризуются набором реагентов и продуктов, причем эти параметры, и только они, позволяют однозначно идентифицировать реакцию. В связи с их множественностью построение реляционной модели и сведение ее к третьей нормальной форме весьма затруднительно, так как по сути нет естественного первичного ключа, однозначно идентифицирующего такой объект (хотя и существует способ естественной идентификации).

Вопрос синонимии особенно важен при интеграции данных, когда не просто сливается несколько баз в единое пространство с единым форматом представления, но также устанавливаются факты эквивалентности объектов из различных баз. Возвращаясь к биологическим видам, заметим, что их названия однозначно идентифицируют сами виды, но при этом они являются многозначными. Можно, конечно, использовать каноническое латинское название как первичный ключ, но никто не гарантирует, что во вновь интегрируемой базе вид будет идентифицироваться именно им, а не, например, английским названием. В результате критерий эквивалентности, построенный на базе такого ключа, не будет выполнен и не будет установлена эквивалентность. Более правильно применительно к биологическим видам критерий эквивалентности объектов из разных баз формулировать так: *если в базе A каждый объект типа Species (вид) характеризуется множеством синонимичных имен и база B обладает таким же свойством, то два объекта из баз A и B эквивалентны, если два соответствующих множества синонимов имен имеют непустое пересечение.*

Подводя итоги, скажем, что модель данных, представляющих индексную информацию в базе MetaBase, напоминает реляционную модель, но есть два существенных отличия. Первое отличие: схема задается динамически, а не статически, как в традиционных реляционных СУБД (этот аспект был рассмотрен в подразделе "Модель предметной области"). Второе существенно отличие: каждый атрибут может иметь множество значений. Это накладывает отпечаток и на язык запросов, заставляя внести некоторые

модификации в традиционный SQL. Более подробно эти отличия рассмотрены далее в разделе “Реализация”.

4.4. Интеграция данных

Для создания информационного пространства, обеспечивающего сравнительно полное покрытие рассматриваемой области, необходимо интегрировать разноплановые базы, количество которых измеряется двумя-тремя десятками. Ряд аспектов, связанных с такими базами, обязательно надо учитывать.

— Количество потенциальных баз достаточно велико, чтобы рассмотреть вопрос об универсальных механизмах интеграции и максимальном упрощении этого процесса.

— Семантика баз и форматы представления данных различаются. Некоторое семантическое пересечение тем не менее есть практически у всех баз, в противном случае интеграция была бы бессмысленна.

— Информация в базах не обладает такими свойствами, как полнота, абсолютная достоверность и непротиворечивость. Это обусловлено пополнением таких баз вручную. Требуется разработка методов диагностики ошибок и противоречий, а также заполнения пробелов.

Сам процесс интеграции данных разбивается на две стадии. На первой стадии происходит простая трансформация из формата интегрируемой базы данных во внутренний формат MetaBase. Для осуществления этого процесса требуется уточнение модели предметной области, а также разработка специализированного драйвера, который, как уже было сказано, в большинстве случаев представляет собой XSLT-трансформацию. Сама проблема преобразования форматов является чисто технической.

Более интересна вторая стадия, на которой происходит семантическая интеграция данных. К началу этой стадии имеется ряд новых объектов, помещенных в общее информационное пространство. Это означает, в частности, то, что мы уже можем производить их поиск стандартными средствами. Но пока еще не решена задача инкрементального пополнения уже накопленных знаний и анализа ошибок. Эту же задачу можно сформулировать с точки зрения конечного пользователя в виде следующего варианта использования системы: *результаты запроса, представляющие информацию об определенном биологическом объекте, должны быть представлены пользователю в виде одной информационной сущности; при этом не важно, из какого количества источников собрана эта информация.* Этот вариант использования можно реализовать двумя способами — либо объединять объекты непосредственно во время выполнения пользовательского запроса, либо заранее, при интеграции. Был выбран последний вариант, так как, во-первых, он дает возможность диагностики ошибок, во-вторых, уменьшает время выполнения запросов. В этом случае необходимо:

1) при интеграции базы задать критерий эквивалентности двух объектов заданного типа;

2) для каждого нового объекта построить запрос на поиск эквивалентов;

3) если результат запроса непустой, то к найденному объекту добавить информацию из текущего. Отметим, что возможен случай, когда найдено более одного объекта. Это означает, что вновь интегрируемый объект заполняет пробел в знаниях и устанавливает эквивалентность двух или более объектов, уже существующих ранее в базе, которые не рассматривались как эквивалентные. В результате, все они будут объединены в единый объект. На практике такие случаи происходят нечасто и могут быть следствием ошибки

в интегрируемой базе, поэтому принятие решения об интеграции может быть оставлено на усмотрение специалиста (в зависимости от настроек системы).

Последние два пункты (2-й и 3-й) система выполняет автоматически.

Само объединение объектов происходит путем объединения значений всех их атрибутов. При этом можно диагностировать некоторые ошибки на основе критериев, заданных при описании предметной области. Например, определенные атрибуты должны иметь хотя бы одно значение. Если критерий не выполнен, объект тем не менее будет сохранен в базе, но помечен как неполный с выдачей соответствующей диагностики оператору. Обратной ситуацией является атрибут с единственным допустимым значением, при этом два объекта, признанные эквивалентными, содержат разные значения этого атрибута. Возможны и другие ситуации, большинство из которых система сама не способна разрешить, но способна их диагностировать и оставлять их разрешение на усмотрение специалиста.

Отметим также, что большинство операций в соответствии с принципом “не навредить” обратимо. Они обратимы либо сами по себе, либо за счет дополнительных средств аудита, которые реализованы в системе. Одно из них — источники — рассматриваются в следующем подразделе.

4.5. Источники

Вследствие того, что предлагаемая база данных MetaBase хранит информацию, взятую из различных внешних информационных ресурсов, целесообразно хранить также данные об ее источниках. В разработанной системе это реализовано в виде семейства операций, позволяющих для каждого объекта (и даже значения атрибута) определить его происхождение. Помимо внешних баз данных источниками информации могут являться: описание эксперимента, статья, гипотеза специалиста.

Другое применение механизма источников — интеграция пользовательских данных, введенных вручную. Пользователь (эксперт) может вносить информацию в свою копию базы, доступную только ему. При необходимости переноса этой информации в общее информационное пространство запускается стандартный процесс интеграции, как правило, с низким уровнем доверия. При этом специалист, отвечающий за актуальность информации в общей базе, принимает решение о “допуске” этих объектов в базу.

5. Реализация

5.1. Основные технические решения

Архитектура разработанной системы представлена на рис. 3. Рассмотрим ее основные компоненты.

База MetaBase предназначена для хранения всех данных в системе, включая информацию о модели предметной области, индексную информацию и источники (основные элементы ее схемы приведены на рис. 1). База построена на основе объектно-ориентированной СУБД Intersystems Caché 5 [15]. Выбор обусловлен тем, что применение реляционных СУБД в чистом виде существенно усложняет разработку из-за большого количества элементов схемы, необходимой для функционирования MetaBase. Базирующиеся же на XML СУБД пока еще не обладают должным уровнем производительности при работе с большими объемами данных (единицы и десятки гигабайт).

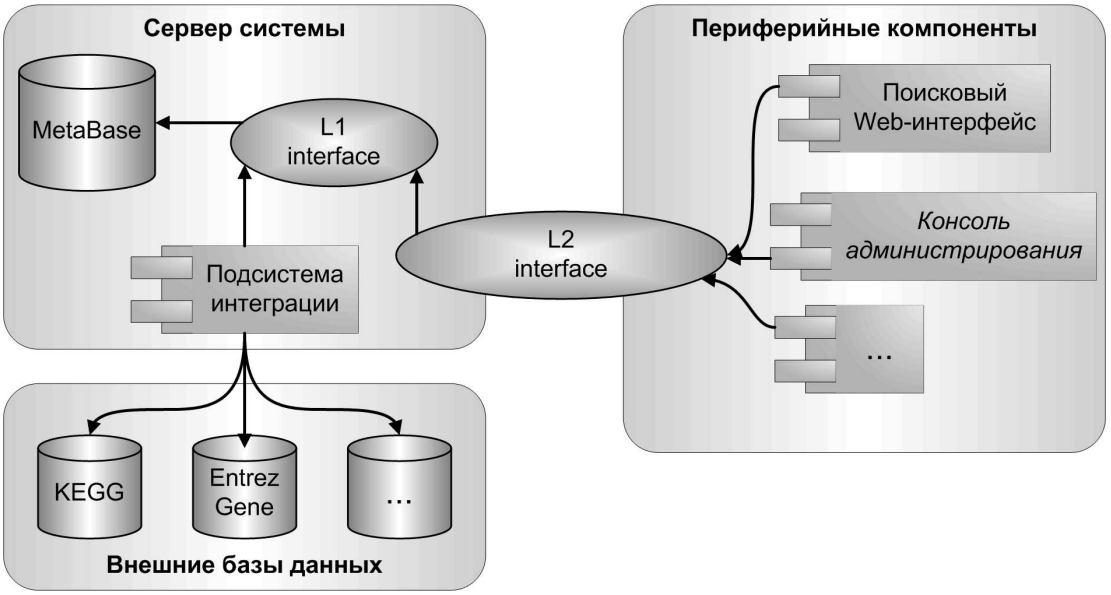

Рис. 3. Архитектура системы интеграции данных

Ввиду того, что выбор оптимальной СУБД для реализации MetaBase неочевиден даже на поздних стадиях разработки, система спроектирована так, чтобы подсистему хранения данных можно было легко заменить. В частности, рассматривается альтернативная реализация на основе Hibernate и какой-либо реляционной базы, например, PostgerSQL или Oracle.

L1 — программная инфраструктура, инкапсулирующая работу с базой данных MetaBase. Она включает в себя набор интерфейсов, не зависимых от базовой СУБД, и реализации этих интерфейсов. Интерфейсы предоставляют базовые функции создания, удаления, модификации и поиска объектов в базе MetaBase и обеспечивают ее целостность. При переходе на другую базовую СУБД интерфейсы сохраняются, меняется только их реализация. Это обеспечивает независимость всех остальных подсистем от используемой СУБД.

L2 — программная инфраструктура, обеспечивающая удаленный доступ к базе MetaBase. Ее интерфейсы практически полностью повторяют интерфейсы L1, но сама библиотека может быть встроена в приложение, работающее как удаленный клиент к базе. Библиотека реализована на основе технологии EJB3 с применением контейнера JBoss 4 и делегирует все свои функции к L1. Разница в функциональности между L1 и L2 заключается в основном в том, что последний активно использует стратегию отложенной загрузки объектов, в то время как в L1 этого не требуется, так как подсистема должна работать непосредственно на том же сервере, что и база данных MetaBase.

Каждый из интерфейсов L1 и L2 предоставляет два альтернативных варианта доступа к базе. Основным является ОО-интерфейс, базирующийся на Java, обеспечивающий все базовые функции, необходимые для работы с базой. Однако в некоторых случаях этот интерфейс не очень удобен. Дополнительно разработан интерфейс, базирующийся на обмене XML-документами. В частности, он позволяет легко импортировать и экспортировать данные для взаимодействия с внешними системами, предназначенными, например, для анализа данных.

Подсистема интеграции данных помимо своей основной функции, следующей из названия, обеспечивает также семантический анализ данных, включающий поиск синонимичных объектов, конфликтов и пропусков. Для преобразования данных из внешних баз во внутренний формат подсистема задействует драйверы внешних баз. Все остальные ее функции унифицированы. Подсистема взаимодействует непосредственно с L1, так как интеграция данных является наиболее ресурсоемкой операцией в системе. С этой точки зрения реализация через L2 существенно уменьшила бы производительность. С другой стороны, непосредственный доступ к базе, увеличив производительность, сделал бы невозможным переход на другую базовую СУБД без модификации этой подсистемы.

Пользовательский поисковый интерфейс служит примером внешнего клиента, который может быть реализован на основе разработанной программной системы. Это базирующийся на web-технологиях интерфейс, позволяющий пользователю сконструировать запрос, выполнить его и просмотреть результаты. На рис. 4 представлен процесс построения запроса (этап 1: выбор типа данных; этап 2: формирование критериев запроса) с помощью предлагаемого интерфейса. На первом этапе выбирается тип (понятие предметной области), экземпляры которого необходимо искать. На втором этапе формируются критерии запроса с учетом атрибутов данного типа. В данном случае выбран тип “ген”, в результате критерии запроса можно строить на основе его атрибутов, а также на основе атрибутов тех понятий, на которые он содержит ссылки. В частности, в приведенном примере запрос строится на основе имени организма, которому ген принадлежит, при этом организм (Species) является отдельным понятием в модели предметной области.

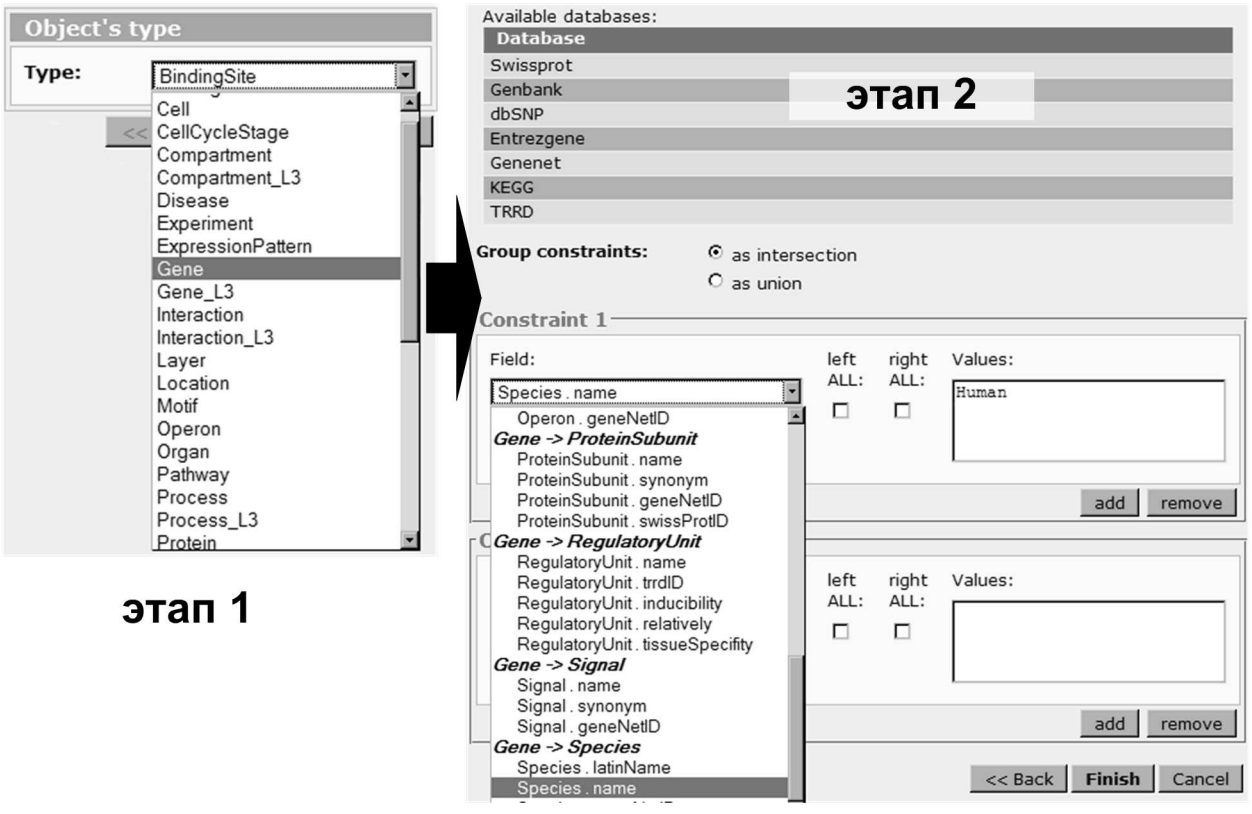

Рис. 4. Построение запроса с помощью поискового web-интерфейса

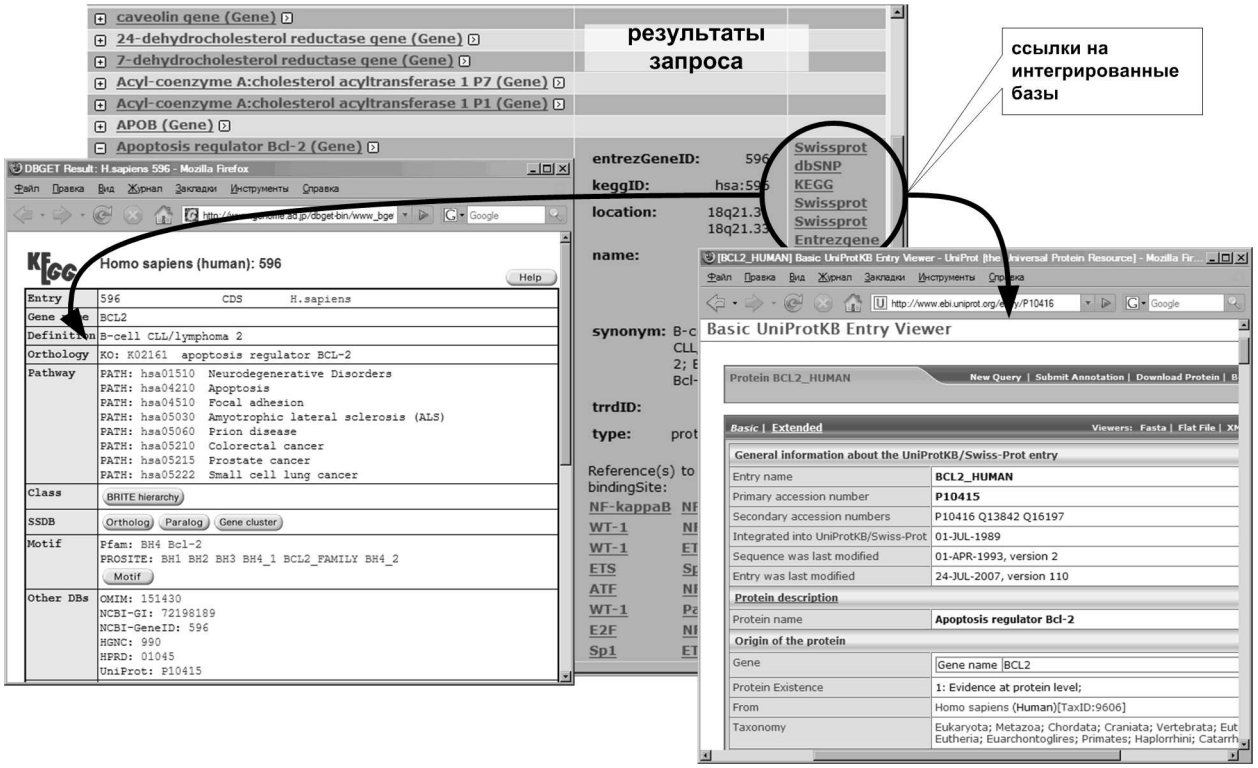

Рис. 5. Просмотр результатов запроса и исходной информации из внешних источников

На рис. 5 показаны результаты выполнения запроса. Интерфейс позволяет осуществлять навигацию по всем ссылкам внутри базы посредством этой формы. Также, что очень важно, здесь присутствуют ссылки на исходные данные из внешних баз, из которых был построен данный объект. В этом примере объект был создан на основе данных из четырех баз. Исходная информация из двух баз представлена на рис. 5.

Система реализована на платформе Java, что обеспечивает переносимость на различные платформы и операционные системы.

5.2. Драйверы баз данных

Предлагаемая авторами программная система предоставляет множество различных точек расширения и интерфейсов, позволяющих развивать ее функциональность в различных аспектах. Наиболее существенными внешними компонентами с точки зрения эксплуатации системы являются драйверы баз данных. Разработка дополнительных пользовательских или программных интерфейсов является вопросом удобства использования, так как в системе уже есть их реализации по умолчанию. Однако при подключении новой базы данных драйвера по умолчанию не существует и существовать не может. Именно поэтому одним из требований при разработке системы стало сведéние сложности разработки нового драйвера к минимуму.

Напомним, что все функции драйвера сведены к преобразованию одного формата данных в другой. Хотя первичным способом доступа к базе MetaBase является программный интерфейс на языке Java, для удобства разработки драйверов, как уже упоминалось, поддерживается также интерфейс, основанный на обмене XML. Для представления модели предметной области в этом случае используется формат XML Schema,

благодаря чему появляется возможность использовать уже имеющиеся внешние программные средства, работающие с XML. В частности, для проверки соответствия данных модели предметной области можно использовать любой пакет разборки XML, поддерживающий XML Schema, например, Xerces. Это позволяет производить первичную проверку корректности данных формату без вовлечения, а следовательно, и без загрузки подсистемы интеграции.

При изучении наиболее значимых баз данных в области молекулярно-генетических систем было отмечено, что все они имеют возможность выдачи информации в виде XML как опцию. Таким образом, драйверу необходимо перевести один XML-документ в другой. Наиболее простой способ это сделать — создать XSLT-преобразование, которое и будет драйвером. Если же внешняя база представлена в каком-либо другом формате, то требуется разработка специализированной программы, что с применением регулярных выражений и других специализированных средств синтаксического анализа также весьма несложно.

Заключение

Результатом работы является разработка технологии семантической интеграции баз данных в области системной биологии, в первую очередь применительно к молекулярно-генетическим системам. На основе этой технологии реализован прототип программной системы, обеспечивающий следующие функциональные возможности:

— интеграция данных из внешних баз данных и других структурированных источников информации;

— семантический анализ интегрируемых данных: идентификация и совмещение эквивалентных объектов из различных источников, выявление противоречий и пробелов в данных;

— прозрачный поиск во всех интегрированных базах через единую точку доступа (пользовательский интерфейс) без реального обращения к ним. Это существенно увеличивает производительность системы на операциях поиска, при этом время выполнения запросов не зависит от качества канала связи с внешней базой;

— аудит интегрируемых данных, позволяющий отслеживать их происхождение, а также устанавливать уровни доверия к различных базам. На основе уровней доверия возможно применение различных стратегий интеграции данных;

— базовые функции разделения доступа (аутентификация, авторизация), необходимые при многопользовательском режиме доступа.

Интегрированы следующие базы данных (разработаны соответствующие драйверы): UniProt [16], GeneNet [17], TRRD [18], GenBank/EMBL [19], EntrezGene [1, 2], KEGG [3, 4], dbSNP [5, 6].

Разработанная система является распределенной, что обеспечивает высокий уровень масштабируемости. Ее структура позволяет производить расширение функциональности: подключать дополнительные базы данных, а также альтернативные пользовательские интерфейсы и другие программные модули прикладного уровня.

В дальнейшем планируется:

— разработка варианта системы на основе других базовых СУБД;

— разработка полноценного интерпретатора языка запросов;

— оптимизация производительности системы.

Список литературы

[1] Maglott D., Ostell J., Pruitt K.D., Tatusova T. Entrez Gene: gene-centered information at NCBI // Nucleic Acids Research, 2005. Vol. 33 (Database Issue). P. D54–D58. [http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=539985]

[2] EntrezGene home page [http://www.ncbi.nlm.nih.gov/sites/entrez?db=gene]

[3] KEGG: Kyoto Encyclopedia of Genes and Genomes [http://www.genome.ad.jp/kegg]

[4] Kanehisa M., Goto S., Kawashima S., Nakaya A. The KEGG databases at GenomeNet // Nucleic Acids Research. 2002. Vol. 30, N 1. P. 42–46. [http://nar.oxfordjournals.org/cgi/content/full/30/1/42]

[5] dbSNP home page [http://www.ncbi.nlm.nih.gov/projects/SNP/]

[6] Sherry S.T., Ward M.H., Kholodov M. et al. dbSNP: the NCBI database of genetic variation // Nucleic Acids Research. 2001. Vol. 29, N 1. P. 308–311. [http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=Retrieve&db=PubMed&dopt=Abstract&list_uids=11125122]

[7] NCBI home page [http://www.ncbi.nlm.nih.gov/]

[8] Davidson S.B., Crabtree J., Brunk B.P., et al. K2/Kleisli and GUS: Experiments in integrated access to genomic data sources // IBM Systems J. 2001. Vol. 40, N 2. P. 512–531.

[9] Donelson L., Tarczy-Hornoch P., Mork P., et al. The BioMediator System as a Tool for Integrating Biologic Databases on the Web // Proc. of the Workshop on Information Integration on the Web. 2004. P. 779–783.

[10] Stevens R., Goble C., Paton N.W. et al. Complex Query Formulation Over Diverse Information Sources in TAMBIS // Proc. of Workshop on Computation of Biochemical Pathways and Genetic Networks (August 1999), European Media Lab (EML). P. 83–88.

[11] Object Data Management Group [http://www.odbms.org/odmg.html]

[12] BioWisdom | SRS [http://www.biowisdom.com/navigation/srs/srs]

[13] SIGMOID home page [http://www.sigmoid.org/]

[14] Gene Ontology Home [http://www.geneontology.org/]

[15] Инновационная высокопроизводительная система управления базами данных [http://www.intersystems.ru/cache/index.html]

[16] UniProt home page [http://www.ebi.uniprot.org/index.shtml]

[17] Ananko E.A., Podkolodny N.L., Stepanenko I.L. et al. GeneNet in 2005 // Nucleic Acids Research. 2005. Vol. 33 (Database Issue). P. D425–D427.

[18] Kolchanov N., Ignatieva E., Podkolodnaya O. et al. Transcription Regulatory Regions Database (TRRD): a Source of Experimentally Confirmed Data on Transcription Regulatory Regions of Eukaryotic Genes // Bioinformatics of Genome Regulation and Structure II. Springer Science+Business Media, Inc. 2006. P. 43–53.

[19] GenBank Overview [http://www.ncbi.nlm.nih.gov/Genbank/]

Поступила в редакцию 3 декабря июля 2007 г.,
в переработанном виде — 23 апреля 2008 г.