УЧЕНЫЕ ЗАПИСКИ КАЗАНСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Том 149, кн. 2 Физико-математические науки *2007*

УДК 004.658.6+004.82+004.89+004.9

ИНТЕГРАЦИЯ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ НА ОСНОВЕ ОНТОЛОГИЙ

Е.В. Биряльцев, А.М. Гусенков

Аннотация

Рассматриваются подходы к интеграции информационных ресурсов, представленных в виде реляционных баз данных. Исследуются особенности представления информации в реляционных базах данных. Обсуждается способ интеграции гетерогенных реляционных баз данных на основе онтологий, а также необходимые для этого информационные структуры. Предлагается подход к автоматизации построения интеграционных процедур на основе инвариантных к структуре баз данных и предметной области лингвистических эвристик.

Введение

В процессе развития информационных систем крупных корпораций внедряется значительное количество (доходящее до нескольких сотен) разнообразных информационных систем. Современные информационные системы имеют, как правило, собственные базы данных, в которых аккумулируется вводимая и получаемая в процессе эксплуатации информация. В силу информационной и логической связности бизнес-процессов, автоматизируемых различными информационными системами, часть информации, находящаяся в различных базах данных, неизбежно дублируется. В первую очередь это относится к так называемой условно-постоянной информации – словарям, справочникам и другой информации, не меняющейся в оперативном цикле обработки. Другим источником дублирования данных является разрыв бизнес-процессов между различными информационными системами. В этом случае данные, получаемые как выходные в одной из систем, являются входными для другой. Наиболее часто это встречается при агрегации данных – учетные системы генерируют стандартные периодические отчеты, которые являются входными данными для аналитических систем. При отсутствии компьютерных средств переноса информации между информационными системами информация переносится путем ручного ввода во все соответствующие системы. Помимо избыточных трудозатрат, ручной повторный ввод неизбежно приводит к накоплению ошибок, в результате чего данные из различных систем становятся несопоставимыми.

Необходимо отметить, что роль информационных систем, использующих реляционные базы данных в экономике, неизмеримо выше роли систем, оперирующих сплошными текстами, графическими данными и другими типами информации. Это связано прежде всего с наличием развитых средств алгоритмического манипулирования структурированными данными. Для экономики задача интеграции информационных ресурсов, представленных в виде реляционных баз данных, наиболее актуальна, в связи с чем для решения задач интеграции реляционных баз данных были разработаны несколько подходов, наиболее известные из которых следующие.

1. Федеративные базы данных [1] – подход, заключающийся в реализации взаимных связей между каждой из баз данных и всеми другими. Основная проблема

заключается в написании большого количества фрагментов программных кодов, обеспечивающих трансляцию запросов от одной СУБД в терминах другой.

2. Хранилища данных [2] – копии фрагментов информации из нескольких СУБД сохраняются в единой базе данных, как правило, с предварительной обработкой (фильтрацией, соединением, агрегированием). В процессе копирования данные подвергаются преобразованиям с целью согласования их структур с общей схемой хранилища[3, 4].

3. Медиаторы [5] – программные компоненты, обеспечивающие поддержку виртуальных баз данных. Медиатор транслирует каждый запрос пользователя в один или несколько запросов, адресованных различным СУБД. По результатам обработки частных запросов медиатор синтезирует ответ на исходный запрос.

Технологии хранилищ данных и медиаторов предусматривают создание программных оболочек, выполняющих функции извлечения информации из первичных источников (конкретных СУБД). При реализации схемы хранилищ данных обычно используются «встроенные» запросы, которые обращаются к источникам и поставляют информацию для хранилища данных. В системах, основанных на медиаторах, применяются более сложные оболочки, которые способны принимать различные запросы, посылаемые медиатором источнику. Здесь используются параметризованные шаблоны запросов, а также генераторы оболочек на основе высокоуровневых спецификаций [6]. Для конструирования плана запросов (последовательности взаимодействия медиатора с оболочками) используются различные оригинальные стратегии [7, 8].

Все перечисленные подходы предусматривают хранение знаний в процедурном виде, и при изменении структуры источников или состава интегральных запросов требуется программирование с участием человека.

В настоящее время для повышения интеллектуализации операций поиска данных в гомогенных и гетерогенных информационных массивах и переноса данных между гетерогенными информационными массивами используется описание семантики предметных областей в виде онтологии [9]. Данный подход использован, в частности, в концепции семантического Web, предложенной Тимом Бернесом Ли и разрабатываемой консорциумом W3C[1]. До сих пор он применялся, в основном, для описания семантики сплошных или слабоструктурированных текстов.

Реляционные базы данных (РБД) существенно отличаются по формату представления и концепции проектирования от массивов сплошных текстов. В отличие от сплошных текстов, данные в РБД имеют сложную структуру. Доступ на чтение и запись осуществляются с помощью запросов на SQL, а не прямым доступом. Структуры двух семантически эквивалентных баз данных могут быть спроектированы на разных уровнях абстракции, отношения (таблицы) баз данных могут иметь различную глубину нормализации, домены (пользовательские типы данных) и ограничения целостности также могут иметь различающиеся формулировки.

Для эффективной интеграции (агрегации, миграции) информационных ресурсов, накопленных в РБД, необходимо решить задачу автоматической (или максимально автоматизированной) генерации SQL-запросов, осуществляющих выборку и запись произвольных семантически и синтаксически корректных наборов атрибутов предметной области. Наиболее проблемной частью является автоматизированное построение запроса на выборку, который извлекает данные из некоторой РБД в соответствии со структурой, заданной пользователем (человеком или программным агентом). Ниже рассматривается подход к автоматизированному построению таких запросов, основанный на концепциях Semantic Web.

[1]World Wide Web consortium, http://www.w3.org

1. Проблемы интеграции реляционных баз данных

1.1. Структурные проблемы. Структура хранимой в реляционных базах данных информации существенно отличается от сплошного текста. Если для поиска в сплошных текстах в простейшем случае достаточно линейного просмотра текста в поисках ключевых слов, то организовать поиск в реляционной базе данных подобным образом невозможно [10]. Доступ к реляционным базам предполагает знание пользователем структуры базы данных и производится либо через заранее предопределенные формы (QBE), либо посредством языка запросов SQL. Необходимо генерировать синтаксически правильные и семантически осмысленные запросы (как правило, на соответствующей базе данных диалекта SQL) и анализировать текст ответа. В настоящее время подобная генерация осуществляется программистом, что определяется следующими причинами.

• Разбиение данных по таблицам может быть осуществлено очень большим количеством способов, конкретный вариант разбиения конечному пользователю неизвестен, его изучение и составление соответствующего структуре запроса требуют значительного времени и квалификации, которые могут у конечного пользователя отсутствовать;

• Названия таблиц, столбцов и словарных элементов могут отличаться от употребляемых пользователем, даже в пределах одного языка. Перебор пользователем всех возможных формулировок запроса практически неосуществим.

Рассмотрим эти проблемы далее подробно на примере условной компании «Центр», осуществляющей сбор информации от своих подразделений, условно называемых «Восток», «Запад», «Север» и «Юг», в таблицу, содержащую сведения об объемах продаж. Целевая таблица имеет вид, приведенный в табл. 1, где столбцы ГОД, МЕСЯЦ и ПОДРАЗДЕЛЕНИЕ образуют первичный ключ таблицы.

Табл. 1

Баланс

Год	Месяц	Подразделение	Затраты	Поступления

Первая проблема, с которой приходится сталкиваться при переносе информации между этими базами, заключается в возможных различиях в физической структуре таблиц баз данных, т. е. в различном распределении столбцов по таблицам. Такое различие обусловливается гибкостью правил построения схем БД, так называемых правил нормализации. Техника нормализации широко освещена в литературе [11], поэтому отметим только, что при применении правил нормализации одной и той же информации различными проектировщиками мы будем получать различные схемы нормализации в зависимости от квалификации, опыта проектировщиков, преследуемых ими целей и учета особенностей конкретных СУБД. На практике наиболее широко распространен случай, при котором некоторая группа неключевых атрибутов может быть как сохранена в основной таблице, так и вынесена в отдельную таблицу. Например, в нашем модельном случае исходная таблица (табл. 1) может быть разбита на две (табл. 2а и 2б).

Мы рассмотрели идеализированную картину, когда базы, участвующие в интеграции, различаются только структурой таблиц. Исходные наборы атрибутов предполагались одинаковыми. Для реальных баз данных это условие не выполняется. Существует еще две проблемы.

Первая проблема связана с возможными отличиями в уровнях абстракции при проектировании баз данных. Так, структуры данных типа, имеющего несколько незначительно отличающихся субтипов, можно представить как отдельными таблицами для каждого из субтипов, так и одной таблицей, в которой будут включены

Табл. 2а

Поступления

Год	Месяц	Подразделение	Поступления

и

Табл. 2б

Затраты

Год	Месяц	Подразделение	Затраты

атрибуты всех субтипов данного типа. В последнем случае в состав таблицы может быть включен столбец-селектор, значение которого определяет субтип конкретной записи в таблице. В практике баз данных известен подход, когда таблица разбивается на множество таблиц, соответствующих значению какого-либо столбца перечисляемого типа (сегментация по ключу, [11]). В качестве примера рассмотрим таблицу, содержащую сведения об объемах продаж условной компании «Центр». Исходная страница имеет вид, где столбцы ГОД, МЕСЯЦ и ПОДРАЗДЕЛЕНИЕ образуют первичный ключ таблицы.

При применении сегментации по ключу мы можем получить следующий набор таблиц:

Табл. 3а

Баланс сегментирован по годам

Баланс 2004

Месяц	Подразделение	Затраты	Поступления

Баланс 2005

Месяц	Подразделение	Затраты	Поступления

Баланс 2006

Месяц	Подразделение	Затраты	Поступления

или

Табл. 3б

Баланс сегментирован по подразделениям

Баланс Запад

Год	Месяц	Затраты	Поступления

Баланс Восток

Год	Месяц	Затраты	Поступления

Баланс Север

Год	Месяц	Затраты	Поступления

Баланс Юг

Год	Месяц	Затраты	Поступления

Как правило, при сегментации по ключу значение ключевого столбца в той или иной форме выносится в название таблицы.

Другой часто используемой возможностью является вынос значения столбца, входящего в состав ключа, в наименование столбца. В рассматриваемом примере это может быть сделано также несколькими способами.

В частности, таблицы дохода и расхода могут быть представлены в виде:

Табл. 4а

Названия подразделений вынесены из ключа в названия столбцов

Год	Месяц	Запад	Восток	Север	Юг

или

Табл. 4б

Названия месяцев вынесены из ключа в названия столбцов

Год	Подр.	Янв.	Февр.	Март	Апр.	Май	Июнь	Июль	Авг.	Сент.	Окт.	Ноя.	Дек.

Очевидно, что возможны также комбинации перечисленных случаев, когда в наименование столбцов или таблиц выносится комбинация ключевых параметров.

Отдельной проблемой является неатомарность атрибутов. Так, в нашем случае мы рассматриваем год и месяц как отдельные атрибуты, однако разработчики базы данных вполне могли счесть их одним атрибутом и представить рассматриваемую информацию следующим образом:

Табл. 5

Неатомарный атрибут «Период»

Период	Подразделение	Затраты	Поступления
Январь 2005			
Февраль 2005			
...			

Здесь в поле «Период» содержится два независимых атрибута, и для решения задачи интеграции необходимо производить лексический анализ поля. Можно привести еще несколько аналогичных примеров – фамилия, имя, отчество могут быть представлены как одним атрибутом, так и двумя или тремя, в название фирмы может быть включено указание на ее форму собственности или оно может быть вынесено в отдельный столбец и т. п. Существуют еще более сложные случаи, например, представление почтового адреса, содержащего массу необязательных или редко встречающихся элементов.

Рассматривая проблему атомарности атрибутов, необходимо упомянуть смежную проблему шкалирования атрибутов. Предположим, что четыре подразделения нашей условной компании расположены в США, Норвегии, Саудовской Аравии и Китае. Тогда, совершенно естественно, доходы и расходы будут исчисляться в различных денежных единицах. Кроме того, весьма вероятно, что будет отличаться и календарное исчисление. Для США, кроме того, более естественен счет по неделям, а не по месяцам, и т. п. В приведенных случаях одномерной количественной шкалы эти проблемы решаются достаточно просто приведением всех шкал к некоторой базовой, включающей в себя все остальные шкалы в качестве частного случая. Более сложные проблемы возникают в случае качественных определений, таких как цвет, форма, вкус и т. п.

Табл. 6

Название подразделения по умолчанию

Год	Месяц	Затраты	Поступления

Некоторая информация может вообще не содержаться в базе, а подразумеваться разработчиками и пользователями. Так, если подразделения компании первоначально вели учет доходов и расходов независимо, то в их базах данных информация о подразделении отсутствовала, так как для каждого подразделения свое собственное наименование является константой во всех записях. Таким образом, при консолидации четырех баз данных подразделений мы могли бы иметь дело с информацией совершенно аналогичной структуры (табл. 6), но относящейся к различным подразделениям.

Существует также проблема, связанная с наименованиями объектов баз данных. Трудно ожидать, что независимые разработчики назовут одинаковые по смыслу объекты (таблицы, столбцы, элементы доменов) совершенно одинаково. В наименованиях объектов могут быть использованы синонимы, сокращения, аббревиатуры, применены различающиеся грамматические конструкции.

Так, в рассматриваемом примере исходная таблица вполне могла бы иметь такой вид:

Табл. 7а

Синонимы в названии элементов БД

Год	Месяц	Отделение	Расходы	Доходы

или

Табл. 7б

Сокращения в названии элементов БД

Год	Месяц	Подразделение	Затраты	Поступление

Эта проблема заслуживает отдельного рассмотрения.

1.2. Лексико-семантические проблемы. Дальнейшее рассмотрение вопроса интеграции будем вести на примере реальных баз данных из области нефтяной промышленности. В частности, изучалась задача переноса данных из базы данных MS SQL в две базы данных Oracle, в совокупности приблизительно эквивалентных по составу информации базе данных в MS SQL. Некоторые параметры их структуры приведены в табл. 8.

Табл. 8

Состав исследуемых баз данных

	MS SQL	Oracle1	Oracle2
Таблиц	644	219	95
Столбцов	11688	2028	524

Типичное описание структуры таблиц рассматриваемых баз данных приведено в табл. 9. Очевидно, что идентификаторы столбцов реальных баз данных неинформативны для задачи идентификации их семантики. Более информативными

являются определения столбцов. В определениях обычно используется лексика литературного языка и профессиональные термины из предметной области, к которой относятся интегрируемые базы данных. Применение описаний для идентификации семантики столбцов представляется единственно возможным путем, но переводит задачу интеграции в область лингвистики и требует ее рассмотрения с лингвистической точки зрения.

Табл. 9

Типичное описание таблицы

Идентификатор столбца	Описание столбца
NC	Номер скважины
GOD	Год
MES	Месяц
PL	Код пласта
SPEX	Способ эксплуатации
DN	Добыча нефти
DW	Добыча воды
DG	Добыча газа
KDEX	Часов работы
PLB	Плотность попутно добытой воды

Даже поверхностный анализ табл. 8 показывает, что подходы к разработке баз данных в MS SQL и Oracle существенно различаются. Количество столбцов и таблиц в базах данных различается в несколько раз. Между тем количество терминов, содержащихся в определениях столбцов и словарных элементов, приведенное в табл. 10, показывает, что количество семантически различных элементов баз данных приблизительно совпадает.

Табл. 10

Общее количество терминов в исследуемых базах данных

Записи	Количество
Общее количество	48 629
Из них из Oracle	24 035
Из них из MS SQL	24 594

Непосредственное совпадение определений терминов наблюдалось менее чем в одном проценте случаев. Вместе с тем проведенный анализ на уровне используемых лексем показывает, что совпадение лексики гораздо более значительное (табл. 11). Разбор терминов на лексемы и устранение морфологических различий показывает совпадение лексики примерно в 40% случаев.

Табл. 11

Совпадение лексики

Элементы	Количество
Всего слов	163 431
Уникальных корней	6 078
Общих в Oracle и MS SQL	2 518
Только в Oracle	2 423
Только в MS SQL	1 137

Интересна природа оставшихся различий в лексическом составе. Рассмотрение причин различий лексики мы будем вести на примере конкретного сопоставления таблицы базы данных MS SQL и таблицы базы данных Oracle, выполненных экспертным путем. Сопоставление приведено в табл. 12.

Табл. 12

Пример сопоставления таблиц различных баз данных

Столбцы таблицы 1	Лексико-семантическое отношение	Столбцы таблицы 2
Номер скважины	Меронимия	Объект разработки
Код пласта		
Год	Гипонимия	Период эксплуатации
Месяц		
Способ эксплуатации	Синонимия	Метод разработки
Добыча воды	Конверсив, Гипнимия	Тип флюида
Добыча нефти		Отдача флюида
Добыча газа		
Часов работы	Анионимия	Процент простоя скважины
Плотность попутно добытой воды		

Анализ табл. 12 демонстрирует многочисленные лексико-семантические отношения между определениями столбцов таблиц двух различных баз данных. Необходимо отметить, что отношение синонимии, наиболее простое для анализа и представления, встречается достаточно редко. В приведенном примере синонимами, в профессиональном контексте, является пара терминов «Способ эксплуатации» и «Метод разработки».

Другие соответствия демонстрируют более сложные отношения. Чтобы установить соответствие столбцов «Год» и «Месяц» в табл. 1 и столбца «Период эксплуатации» в табл. 2, необходимо учитывать, что «Год» и «Месяц» являются гипонимами лексемы «Период», а также, что в описании столбцов «Год» и «Месяц» подразумевается отнесение их к периоду добычи нефти, а не к другому событию или интервалу, например к вводу скважины в эксплуатацию.

Для соотнесения столбцов «Добыча воды», «Добыча нефти», «Добыча газа» табл. 1 и столбцов «Тип флюида» и «Отдача флюида» необходимо учитывать, что «вода», «нефть» и «газ» являются гипонимами термина «флюид» в данном контексте, а «добыча» и «отдача» являются, в профессиональном контексте, конверсивами, выражающими точку зрения на процесс: «Скважина добывает нефть из залежи», «Залежь отдает нефть скважине».

Соответствие столбцов «Часов работы» и «Процент простоя скважины» основано на антонимии терминов «Работа» и «Простой». Зная время простоя скважины и месяц, к которому он относится, можно установить время работы. Дополнительную сложность придает тот факт, что количественные измерения работы и простоя скважины выражаются различными единицами – часы в абсолютном исчислении и проценты в относительном.

Соответствие между парой столбцов «Номер скважины» и «Код пласта» в табл. 1 и столбцом «Объект разработки» отражает тот факт, что объект разработки в профессиональном контексте - это добыча флюида из конкретного пласта на конкретной скважине. Таким образом, объект разработки включает пласт и скважину как составные части. В этом случае между терминами присутствует отношение меронимии.

Достаточно часто в исследуемых базах данных между определениями сопоставляемых столбцов наблюдаются отношения метонимии типа «Объект – Роль», «Процесс – Результат» и некоторые другие.

Представления реляционных баз данных в формализме онтологий и виды лексико-семантических отношений, существующие в реальных базах данных, рассмотрены также в работах [12, 13].

2. Общая схема автоматизированного доступа

Обычно, как было отмечено во введении, доступ пользователя – человека или программного агента – к реляционным базам данных осуществляется через предопределенные процедурные программные компоненты. При решении задачи интеграции или нерегламентированного доступа к большому количеству баз данных такой подход практически нереализуем. Оптимальным вариантом являлась бы возможность построения (генерации) процедур доступа по мере возникновения реальной необходимости получения информации в формате, заданном пользователем. Пользователь, являющийся специалистом в предметной области, в состоянии сформулировать семантически правильный запрос с точностью до наименований элементов требуемой информации и их распределения по объектам базы данных.

Автоматическое построение процедур доступа подразумевает отсутствие программиста в процессе генерации процедур доступа к данным в соответствии со сформулированными запросами. В идеальном случае взаимодействие пользователя с системой автоматизированного выполнения нерегламентированных запросов может происходить следующим образом.

Пользователь конструирует таблицу, в виде которой он хочет получить ответ. Названия столбцов и условия, накладываемые на их содержимое, пользователь формулирует в произвольном виде с использованием слов естественного языка, аббревиатур, сокращений, числовых значений, дат и т. п. Система, при необходимости, уточняет у пользователя некоторые неоднозначно определенные названия столбцов или условий. В результате запрос пользователя, например, представляется таблицей следующего вида:

Табл. 13

Название столбца	Условие
Месторождение	Ново-Елховское
Номер скважины	
Период	С 01.2000 по 12.2005, помесячно
Объем добычи, т	Сумма
Процент обводненности	Среднее

Система также уточняет некоторые дополнительные сведения, необходимые для корректного построения запроса, в частности единицы измерения числовых величин (если в базе данных есть несколько вариантов), гранулярность временных периодов (с какой точностью задается период – год, квартал, месяц, неделя и т. д.). Пользователь указывает, какая групповая операция должна быть применена к результату – сумма, среднее, максимум и т. д. Пользователь может также указать, что ему нужен не фактический, а плановый (прогнозный, расчетный) показатель, задавая одно из этих ключевых слов в названии столбца.

Полученная таблица транслируется системой в SQL-запрос. Система оценивает время выполнения запроса, о чем информирует пользователя. С одобрения пользователя запрос запускается, результат предоставляется пользователю.

В процессе общения с пользователем система накапливает сведения об используемой конкретным пользователем терминологии. Например, если пользователь задаст имя поля как «Скважина», то система первый раз задаст вопрос, о каком параметре скважины идет речь, запомнит ответ и, в дальнейшем, будет использовать именно этот параметр, когда пользователь пишет «Скважина». Также сохраняются и используются в дальнейшем сведения об обозначениях единиц измерения, к которым привык пользователь, сокращениях, аббревиатурах и т. п.

Разбор запроса пользователя идет в 2 этапа. На первом этапе термины пользователя привязываются к логической модели предметной области. После привязки

всех полей таблицы (в процессе их задания пользователем) к атрибутам логической модели система определяет, как спроецировать запрос на реальную базу данных. Для этого используется обратная привязка полей таблиц базы данных к атрибутам логической модели, которая выполняется при включении базы данных в систему.

Привязка таблиц базы данных к логической модели может быть произведена двумя способами. При первом способе программист дает словесное определение введенному полю, система разбирает это описание и в диалоге определяет, какой атрибут логической модели имеется в виду. Диалог в этом случае эквивалентен диалогу системы с пользователем. При втором способе программист привязывает поле к атрибуту логической модели, производя навигацию по логической модели и сам выбирая нужный атрибут.

Для каждого столбца базы указывается также его модальность, единица измерения и ее точность. При необходимости указываются обстоятельства – места, времени, образа действия. Обстоятельства позволяют различать одинаковые атрибуты, измененные в различных обстоятельствах (например, дебет «до ремонта» и «после ремонта», фамилия «до замужества» и «после замужества» и т. п.).

2.1. Анализ. Лексическая фаза. Первый уровень анализа заданного запроса – лексический. Анализируется, все ли лексемы (слова, аббревиатуры, числа и т. п.), использованные пользователем, известны системе. Если не все лексемы известны, то система начинает уточняющий диалог лексического уровня. На этом уровне в первую очередь отсеиваются опечатки и грамматические ошибки, аналогично функциям проверки правописания в текстовых процессорах. Если неизвестное слово не является ошибкой или опечаткой, то в диалоге с пользователем последовательно проверяются следующие лексические отношения – сокращение (т. е. слово является сокращением известного), аббревиатура, синоним, идиоматическое выражение (как правило – словосочетание, в некотором контексте аналогичное известной лексеме). Последней проверяемой гипотезой является классификация лексемы как имени, обозначения, наименования и т. п. конкретного свойства конкретного экземпляра объекта некоторого типа.

Несмотря на теоретическую возможность установления более сложных лексических зависимостей (антонимия, конверсивы и т. д.) для конечного пользователя, не являющегося лингвистом, давать такую возможность нецелесообразно во избежание усложнения интерфейса конечного пользователя. Если на предыдущей стадии не удается включить лексему в лексический словарь, то пользователя целесообразнее попросить о переформулировке запроса целиком, с исключением неизвестной лексемы.

2.2. Анализ. Семантическая фаза. После идентификации лексем производится разбор семантики запроса, исходя из достаточно общих предположений. Основное используемое предположение следующее: вопрос задается о табличной зависимости между некоторыми свойствами (быть частью, быть подтипом – таким образом унифицируются все отношения) однородных (на некотором уровне абстракции) объектов, процессов или явлений (далее – объекты).

Таким образом, первая задача, которую необходимо решить на семантическом уровне, – идентификация объектов, о которых идет речь.

Из всего потока лексем в запросе выделяются лексемы, которые могут быть идентифицированы как названия объектов. Обычно это существительные, стоящие в родительном падеже и играющие грамматическую роль определения: номер **скважины**, код **пласта**, дата **ремонта**. Здесь скважина, пласт и ремонт – объекты, о которых идет речь. Остальные лексемы интерпретируются как названия свойств объектов или другие служебные слова.

На синтаксическом уровне возможны следующие коллизии:

1) названий объектов в запросе нет;

2) названий объектов несколько;

3) названия объектов не соответствуют их свойствам (у них таких свойств нет);

4) одно из полей именуется названием объекта без указания его свойства.

Коллизии разбираются, исходя из известной связи объектов и свойств. В частности, если не указан объект, проверяется гипотеза об однозначном соответствии группы заданных свойств известным типам объектов (т. е. такие свойства могут быть только у объектов определенного типа, и в его явном указании нет надобности). Во втором случае система пытается разобраться, какие свойства каким названным объектам могут соответствовать. В третьем случае указываемые свойства ищутся у подтипов или частей указанных объектов. В четвертом случае с объектом ассоциируется один или несколько естественных ключей (название, номер, индекс и т. п.). Сгенерированные гипотезы, даже в случае их единственности, предъявляются пользователю, который их утверждает или конкретизирует в случае множественного соответствия.

Дальше необходимо определить модальность запрашиваемых свойств – фактическое это значение параметра или некоторое гипотетическое. Лексемы, указывающие на модальность, могут быть откорректированы и привязаны к пользователям или их группам. В их отсутствие у параметров предполагается первая модальность.

Затем анализируется наличие агрегаторов – сумма, среднее, максимум, минимум и их синонимы. Наличие агрегатора показывает, что требуемый параметр необходимо агрегировать указанным образом.

Последним семантическим вопросом является вопрос об используемых пользователем системы наименований, классификаторов и единиц измерений.

Это, в частности, общетехнические единицы измерения (СИ, СГС, внесистемные), а также категории времени, пространства, административная структура, и переопределяемые качественные классификаторы свойств. Цель разбора – понять, в каких шкалах и единицах задаются заголовки столбцов. В отсутствие явно указанных обозначений размерности используются умолчания.

На этом фаза разбора запроса заканчивается. Все столбцы классифицированы как имена известных свойств известных объектов с известной модальностью, агрегацией и шкалой.

2.3. Синтез. На фазе синтеза используется предварительная разметка столбцов таблиц баз данных атрибутами логической модели (онтологии предметной области). К моменту выполнения запроса БД уже размечена.

Процедура поиска решения производится в две стадии:

1) отбор столбцов БД, соответствующих отмеченным атрибутам;

2) установление возможных связей между столбцами:

   а) принадлежность одной таблице;

   б) связь по мигрированному ключу;

   в) связь по домену.

На этой стадии ищутся соединения, связывающие запрашиваемые атрибуты. Методика отбора связей – по приоритету, согласно подпунктам пункта 2. В результате получается ряд соединений, ранжированных по некоторой метрике, например количество связей указанных трех типов с некоторыми весами. Метрика уточняется на реальной базе, так как это – параметр, зависящий от базы и даже от пользователя.

Составленный запрос с одобрения пользователя выполняется. Результат предоставляется пользователю, при необходимости производится уточняющий диалог:

добавляются/удаляются поля, переопределяются ключевые значения, меняется агрегация.

3. Онтологии задачи интеграции реляционных баз данных

Для эффективной интеграции реляционных баз данных необходима информация следующих типов.

Во-первых, это информация о структуре самой базы данных. Связность и близость значений атрибутов БД определяются связностью кортежей через общие ключи. Таким образом, для выдачи пользователю осмысленной информации поисковая машина должна иметь информацию о возможных соединениях кортежей по ключам. Такую информацию достаточно легко получить непосредственно из схем баз данных, так как современные СУБД хранят информацию о ключах в своих системных данных. Восстановление всех возможных осмысленных комбинаций таблиц базы (документов) также является, как показано ниже, достаточно простой задачей.

Во-вторых, это семантические маркеры для единообразной разметки атрибутов всех баз данных. Для маркирования удобно использовать логическую модель предметной области со связями «класс – подкласс – экземпляр» и «часть – целое». Такой подход известен как Semantic Web[2] и в настоящее время широко апробируется в различных приложениях, в частности, для повышения релевантности поиска в сплошных текстах. РБД фактически уже размечены наименованиями атрибутов и в этом отношении имеют большую степень готовности к использованию совместно с технологиями Semantic Web, так как необходимо установить соответствие между наименованием атрибута и некоторой семантической единицей.

В-третьих, поисковый механизм должен содержать лексико-семантическую информацию о терминологии, применяемой различными группами пользователей для обозначения тех или иных объектов, свойств и связей в предметной области (пользовательские тезаурусы). Наличие такой информации позволит решить проблемы синонимии, сокращений и аббревиатур, применяемых различными группами пользователей.

Следует отметить, что каждая из этих информационных структур может быть представлена множеством экземпляров. Поисковый механизм может иметь доступ ко множеству баз данных, эти базы могут быть связаны с различными предметными областями или быть междисциплинарными, различные группы пользователей могут использовать различные терминологические тезаурусы, вплоть до мультиязычных.

Возникает вопрос о формализме представления столь разнородной и объемной информации. Каждая из этих моделей имеет некоторый устоявшийся формализм представления. Так, схемы баз данных традиционно представляются с использованием SQL DDL, логические модели – ER-диаграммами, лексико-семантическая информация – тезаурусами. Очевидно, что оперирование столь разнородными моделями технически затруднительно. Для наших целей оптимально использовать единый достаточно мощный формализм, ориентированный на автоматическую обработку информации, хранящейся в моделях. В качестве такового можно предложить формализм онтологий [14]. Онтологии включают как частный случай логические и физические модели данных и тезаурусы, так что с теоретической точки зрения мощности формализма онтологий достаточно для представления всех трех типов информационных моделей.

[2] http://www.w3.org/2001/sw/

Таким образом, вспомогательные информационные ресурсы должны содержать физические модели баз данных, логические модели предметной области и тезаурусы пользовательской терминологии, представленные в формализме онтологий.

3.1. Представление структуры реляционных баз данных в формализме онтологий. Для описания структуры реляционных баз данных существует достаточное количество формализмов: DDL SQL, ER-диаграммы, UML [15] и другие диаграммные техники. Их мощности вполне достаточно для решения задач проектирования баз данных и поддержки доступа к базам данных, рассматриваемых изолированно, каждая в своей собственной системе атрибутов. При решении задач интеграции информации из двух или более реляционных баз данных возникают проблемы [16], связанные с особенностями наименований артефактов баз данных, в первую очередь таблиц и столбцов. Эти особенности требуют рассмотрения наименований артефактов не просто как атомарных имен, а как самостоятельных объектов, обладающих, возможно, собственной структурой и имеющих между собой семантически нагруженные связи. Такие связи невыразимы в традиционных формализмах представления структуры реляционных баз данных, таким образом, мы не можем не только решить задачу интеграции реляционных баз данных, но и сформулировать ее.

Подходящим формализмом для представления сложноструктурированной информации являются онтологии. Известны подходы к описанию структуры конкретных реляционных баз данных с помощью онтологий [16]. В целом, варианты представления структуры реляционных баз данных сводятся к двум основным подходам.

Первый подход предполагает преобразование множества таблиц конкретной базы данных во множество одноименных концептов со слотами, соответствующими столбцам определенной таблицы, и проецирование связей между таблицами (в виде мигрирующих ключей) в отношения между таблицами. В набор концептов онтологии в таком случае также включалось множество доменов (встроенных или пользовательских типов данных). Экземплярами концептов онтологии в данном случае выступают записи таблиц, значения ключей и состав доменов перечисляемого типа.

Второй подход предполагает более высокую степень абстракции концептов. В этом подходе при представлении структуры реляционных БД в формализме онтологий выделяются универсальные (не зависящие от конкретной базы данных) концепты, как ТАБЛИЦА, СТОЛБЕЦ, КЛЮЧ, ДОМЕН, соответствующие основным объектам баз данных, и универсальные отношения между ними:

ТАБЛИЦА *содержит* СТОЛБЕЦ
ТАБЛИЦА *имеет первичный* КЛЮЧ
ТАБЛИЦА *имеет внешний* КЛЮЧ (1)
КЛЮЧ *содержит* СТОЛБЕЦ
СТОЛБЕЦ *имеет тип* ДОМЕН

Объекты (таблицы, столбцы, ключи и домены) конкретной базы данных во втором случае представляются как экземпляры универсальных концептов соответствующего типа.

Оба подхода имеют как преимущества, так и недостатки. В частности, первый подход позволяет сохранить в онтологии полный экстенсионал базы данных, во втором случае это нереализуемо. Вместе с тем, первый подход подразумевает создание для каждой конкретной базы данных уникальной онтологии, что не дает возможности построить универсальные аксиомы.

Отмеченные выше особенности получаемых онтологий определяют необходимость выбора второго подхода при построении онтологий баз данных для задач

их интеграции, так как мы имеем дело с базами данных произвольной и заранее неизвестной структуры. Таким образом, целесообразнее иметь универсальную онтологию и универсальные аксиомы, а конкретные базы данных трактовать как ее реализации. Возможность же представления в онтологии полного экстенсионала базы данных не находит применения. Достаточно иметь возможность хранить в онтологии условно-постоянную информацию базы данных (словари и справочники), для чего достаточно расширить понятие домена.

Задача интеграции реляционных баз данных может выражаться в совместном использовании информации из двух или более баз данных (далее, для удобства, мы будем говорить о двух базах данных) для построения общего интегрального документа (совместного представления) или в задаче переноса некоторого объема информации из базы-источника в базу-приемник информации. В обоих случаях необходимо приведение информации к некоторой общей для двух баз данных форме.

Любая информация в реляционной базе данных имеет вид набора таблиц, возможно связанных между собой общими ключами. Для дальнейших рассуждений, оставляя открытым вопрос о существенности этих связей, можно сказать, что задача интеграции сводится к последовательному нахождению способов извлечения из базы данных ее атрибутов в виде задаваемых пользователем таблиц. Для задачи построения совместного представления это будет собственно целевая таблица, для задачи переноса информации – это таблицы целевой базы.

В более формализованном виде поставленная задача имеет следующий вид.

Дана онтология O вида (1) и произвольное множество {C} столбцов из O.

Требуется определить, возможно ли построение {C}, и если возможно, то каким образом (задача 2).

Если в одной из таблиц РБД требуемое множество столбцов содержится как подмножество, то задача решается тривиально. Однако в реальных случаях искомое множество столбцов {C} будет содержаться, вероятнее всего, в нескольких различных таблицах. Для восстановления искомого множества из реально существующих таблиц требуется соединение этих таблиц между собой. Правила манипулирования реляционными данными разрешают соединения таблиц только при наличии общего ключа, причем в одной из таблиц этот ключ играет роль первичного, а в другой – вторичного ключа. Для формализации данных правил мы вводим в онтологию (1) две функции интерпретации (ФИ) следующего вида:

ФИ1:

Если ТАБЛИЦА1 имеет первичный КЛЮЧ1 и ТАБЛИЦА2 имеет внешний КЛЮЧ1, то существует ТАБЛИЦА3, содержащая столбцы, принадлежащие ТАБЛИЦА1 и ТАБЛИЦА2.

ФИ2:

Если ТАБЛИЦА1 содержит СТОЛБЕЦ1, то существует ТАБЛИЦА2, содержащая все остальные столбцы ТАБЛИЦА1, кроме СТОЛБЕЦ1.

Первая функция интерпретации соответствует операции соединения по ключу, вторая – операции проекции реляционного отношения, необходимой для сокращения получаемого при соединении таблиц множества столбцов до искомого.

Задача 2 сводится к нахождению такой последовательности применения ФИ1 и ФИ2, которая даст в результате искомое множество {C}, т. е. принадлежит известному классу задач о блуждании по ориентированному графу, где вершинами являются экземпляры концепта ТАБЛИЦА, а дугами – наличие общего ключа, ориентированного посредством отношений «*имеет первичный*» и «*имеет вторичный*». Подходы к решению данного класса задач хорошо известны [17] и представляют лишь вычислительную сложность при большой размерности задачи.

Приведенная задача рассматривалась в предположении, что набор атрибутов обеих баз данных, участвующих в процессе интеграции, одинаков или имеет пересечение по искомой информации. В любом случае для применения предложенных подходов одинаковые по семантике атрибуты должны иметь одинаковое наименование.

3.2. Онтология предметной области. Несмотря на активное создание в последнее время онтологий разнообразных предметных областей, они, как правило, носят экспериментальный характер. Онтологии, которые являлись бы для какой-либо предметной области стандартом де-юре или де-факто, в настоящее время отсутствуют. В качестве прототипа онтологии предметной области можно использовать логические модели, созданные в некоторых отраслях и имеющие статус отраслевого стандарта. Одной из них является модель данных Epicentre версии 3.0 нефтетехнической корпорации Petrotechnical Open Software Corporation (POSC)[3]. Она представлена в виде ER-диаграмм [11], а также в виде набора текстовых файлов на языке EXPRESS (ISO 10303, part 11). Это представление ориентировано на генерацию структур баз данных по логической модели, а также на визуальное восприятие IT-специалистами. Априори возможность представления названной модели в виде онтологий была неочевидной.

В модели данных Epicentre определено более 1000 реально существующих технических и бизнес-объектов, связанных с разведкой и добычей нефти. В терминологии POSC-моделирования данных эти объекты названы сущностями (entities). В модели определены характеристики, которые могут содержать сущности, названные атрибутами сущностей (attributes). Наиболее важными являются атрибуты, определяющие взаимосвязи между сущностями.

Один из основных принципов модели Epicentre основан на различии между объектами, свойствами или характеристиками объектов (properties) и видами деятельности (activities), которые используют объекты и определяют их свойства. Отличительная особенность свойств объекта – это возможность иметь многократные версии или описания, а также наличие однозначной связи свойства со своим собственным определением (описанием) или историей обработки. Модель Epicentre рассматривает принцип моделирования, при котором конкретному экземпляру сущности – объекту – обеспечивается его существование. Поэтому каждый экземпляр представляет отдельный объект и не может быть версией существующего объекта. Версии характеристик объекта отделены от объекта. В Epicentre эти характеристики связаны с наличием атрибутов или свойств сущностей. Также в модели Epicentre спецификация сущностей была расширена для того, чтобы включить сущности, имеющие стандартный набор экземпляров. Подобные сущности называются справочными (reference_entities) и отличаются от других тем, что некоторый стандартный набор значений был определен POSC. Их присутствие требуется для совместимости со спецификациями POSC. Все справочные сущности имеют дополнительные характеристики, позволяющие задать источник информации, содержащейся в экземпляре, и связанную с ним библиографию.

Другая фундаментальная часть архитектуры Epicentre – это понимание того, что многие сущности характеризуются пространственным представлением. Чтобы облегчить общее использование пространственных данных для всей модели, POSC определил набор общих пространственных объектов и пространственных связей. Для каждого геометрического объекта деятельности в разных разделах модели может быть задано его местоположение через отношения с одним или несколькими общими пространственными объектами.

[3]www.posc.org

Логическая модель данных, предлагаемая POSC, представляет собой набор определений сущностей и связей между ними в виде выражений на языке Express и диаграмм «сущность-связь». Каждая сущность, представленная в модели, определяется такими параметрами, как набор атрибутов, локальные правила, цепочки супертипов. Атрибут – перечень объектов, составляющих данную сущность. Атрибут имеет несколько параметров, таких как имя атрибута, список опций (ключевой, обязательный, внешний и др.), типы связей атрибута. С сущностями связаны также Правила сущности – расширенные ограничения целостности данных, влияющие на возможные значения атрибутов и связей.

Сущности в модели делятся на разные уровни абстракции представляемых объектов. Описание предметной области начинается с очень высокого уровня абстракции, на котором нефтегазовая специфика практически отсутствует. Высокоуровневая модель сущностей строится с понятий E_and_P_data, под которым понимается любой информационный объект, и Data_collection, под которым понимается произвольный набор объектов. При дальнейшей детализации возникают более специфические объекты, описывающие более узкие понятия, относящиеся к следующим подгруппам:

– геолого-геофизические данные, включающие в себя структуры данных, получаемых в результате проведения различных геофизических исследований и изысканий;

– геологические данные, включающие в себя структуры данных, необходимых для описания строения земной коры и земной поверхности;

– технические данные, включающие в себя структуры данных по конструкции скважин и нефтепроводов, бурового оборудования, применяемых методах и материалах, данные по производимым операциям и параметрам получаемых продуктов.

Физическая модель данных в POSC представляет собой набор конкретных таблиц с конкретными столбцами. Для ее получения объектная модель Epicenter приводится к реляционной модели путем операции, называемой в стандарте POSC «проекцией». Правила проекции определяют варианты физической структуры базы данных. Можно варьировать названия отношений и атрибутов БД, а также до определенной степени глубину нормализации, ряд других характеристик. Таким образом, в стандарте POSC предполагается получение физической модели хранения из логической модели. Доступ к данным осуществляется через запросы на специализированном языке PSQL, оперирующем понятиями логической модели данных, что позволяет интегрировать различные базы данных, полученные различными проекциями. Вместе с тем механизмы проекции, предлагаемые POSC, не позволяют построить обратную проекцию произвольной физической структуры РБД на логическую модель, что сильно ограничивает применимость стандарта POSC для миграции данных произвольного формата.

3.3. Лингвистическая онтология. Для создания лингвистической онтологии природно-технических объектов выбран подход, основанный на построении тезаурусов WordNet [18, 19]. Словарь предметной области был построен путем объединения словоформ из описаний сущностей и атрибутов модели Epicentre и описаний атрибутов таблиц и доменов таблиц-справочников реляционных баз данных реальной нефтедобывающей корпорации. Лексико-семантические характеристики баз данных, использованных при построении словаря, описаны в [11, 20]. В настоящее время словарь содержит около 6000 словоформ. Для каждого слова определяется входной синонимический ряд (синсет). На лексико-семантических вариантах слов и на синсетах определены следующие отношения: гипонимия, часть – целое, несовместимость, антонимия, конверсивность, омонимия.

Описаниям синсетов построенного таким образом тезауруса присущи особенности, вытекающие из специфики описания атрибутов реальных баз данных: наличие коротких фраз, сокращений, технических аббревиатур, орфографических ошибок [20]. Однако в отличие от анализа сплошных текстов для баз данных имеется возможность уточнения распознавания входных слов путем просмотра содержимого доменов атрибутов таблиц и сопоставления их с онтологией предметной области.

4. Реализация

В качестве единого средства представления онтологий был выбран язык OWL, разработанный рабочей группой Semantic Web Activity и рекомендованный консорциумом W3C, а именно диалект языка OWL-DL (Description Logic)[4]. Выбор языка OWL-DL обусловлен чисто практическими аспектами, в частности, его поддержкой в существующих сегодня системах описания знаний и системах логического программирования, а также перспективами его распространения в будущем как международного стандарта.

Для апробации изложенного выше подхода в 2006–2007 гг. была разработана экспериментальная среда OakOwlProject 1.0. Разработка преследовала цель программно отработать представление и манипуляции со всей совокупностью онтологий.

В качестве прототипа для реализации некоторых интерфейсных и функциональных возможностей среды OakOwlProject был выбран известный OpenSource проект Protégé[5]. Этот проект был существенно функционально расширен для решения задачи автоматизированного построения и интеграции разнородных онтологий. Реализация среды OakOwlProject выполнена на языке Java.

OakOwlProject поддерживает операции со следующими структурами данных:

– онтологии предметной области;

– онтологии баз данных;

– онтологии языка предметной области,

а также модуль продукций – набор инструментов для интеграции концептов перечисленных выше онтологий.

Внешний вид интерфейса OakOwlProject в части его работы с онтологиями предметной области показан на рис. 1.

4.1. Онтология предметной области. Для построения онтологии предметной области была использована логическая модель Epicentre и разработана схема конвертации этой модели в язык описания онтологий OWL, рекомендованный консорциумом W3C.

При построении OWL-онтологии были использованы следующие основные подходы:

• любой сущности Epicentre соответствует простой именованный класс OWL-онтологии с сохранением в именах классов приставок, позволяющих идентифицировать сущности-свойства и сущности-справочники; все эти классы располагаются в корне таксономического дерева онтологии;

• степени связности сущностей (один-к-одному, один-ко-многим, многие-ко-многим) в OWL соответствует определение простых свойств-атрибутов, если связанная сущность не является типом данных, и свойств-значений в противном случае; указание степени связи между классами реализовано с помощью понятия кардинальности в OWL.

[4]http://www.w3.org

[5]http://protege.stanford.edu/

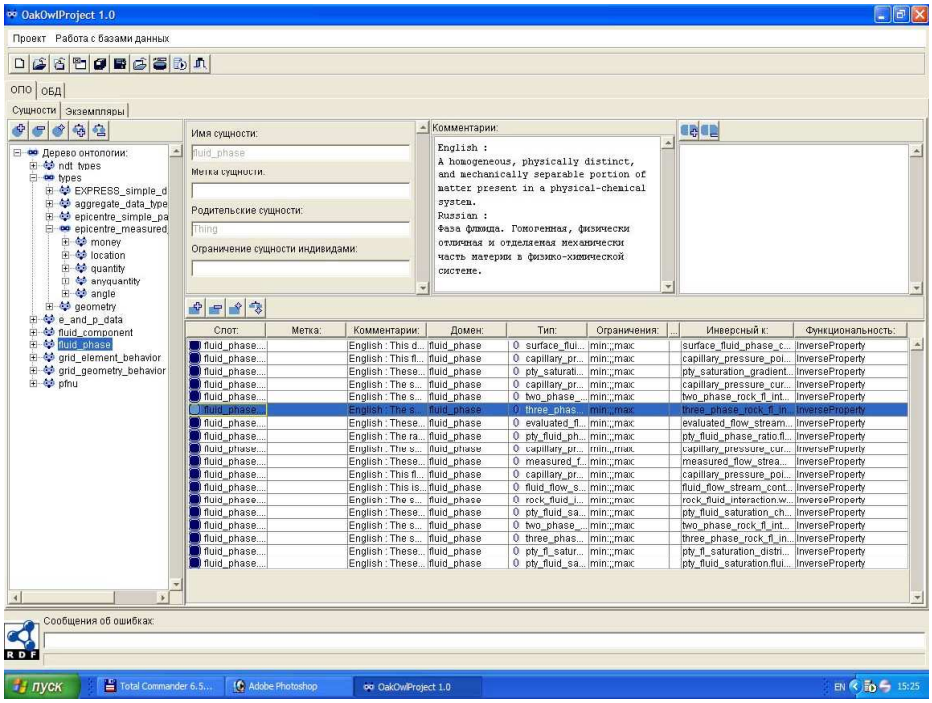

Рис. 1. Общий вид интерфейса пользователя системы OakOwlProject

В OWL отсутствуют структурные элементы, которые в полном объеме описывают определение уникальности Epicentre. Поэтому в определение каждого класса на языке OWL добавлено новое предопределенное свойство, в котором перечислены все атрибуты, образующие уникальный ключ. Аналогичным же образом решена проблема сохранения условий ограничений.

Для каждой категории данных Epicentre в OWL-онтологии построены отдельные классы, в свойствах которых использовались встроенные типы данных языка OWL. Построена формальная LR(1) [21] грамматика модели Epicentre, на основе которой реализовано семантическое преобразование модели Epicentre версии 3.0, описанной на языке EXPRESS, в онтологию на языке OWL. Выполнена русификация описания сущностей и атрибутов модели Epicentre, а также соответствующих им классов и свойств на OWL. Более подробно схема построения OWL-онтологии на основе модели Epicentre изложена в работах [22, 23]. Онтология реализована на диалекте языка OWL DL, соответствующем правилам дескриптивной логики, что в дальнейшем позволит использовать системы логического вывода на экземплярах понятий.

В рамках экспериментальных работ были конвертированы в OWL модели данных Epicentre версий 2.2. и 3.1. Каждая их полученных онтологий имеет объем порядка 200 тыс. строк.

4.2. Онтологии баз данных. Онтологии баз данных в OakOwlProject строятся автоматизированным образом, путем импорта системных данных из целевой схемы баз данных. Реализованная версия поддерживает извлечение системных данных из MS SQL и Oracle и их дальнейшее преобразование в экземпляры онтологий. Отработка манипуляций с базами данных велась на приведенных в п. 1.2. базах данных. Некоторые их характеристики приведены в табл. 14.

Табл. 14

Характеристики тестовых баз данных

	MS SQL	Oracle1	Oracle2
Таблица	644	219	95
Представление	301	30	12
Синоним		1093	3
Столбец	11688	2028	524
Индекс		729	263
Ключ	18	0	0
Первичный Ключ		171	88
Внешний Ключ		392	84

Тестовые базы данных анализировались на полноту информации, необходимой для дальнейших операций по интеграции данных. Поскольку тестовые базы данных являлись реальными экземплярами промышленно эксплуатируемых систем, по анализу полноты можно было сделать выводы об объемах необходимых подготовительных работ для реального применения предлагаемых подходов. Анализ показал, что таблицы и столбцы баз данных, являющиеся исходным материалом для сопоставления столбцов и построения интеграционных процедур, имеют недостаточно полные комментарии.

Информация о возможных соединениях таблиц, необходимая для автоматизации построения интеграционных процедур, извлекается из информации о внешних ключах. Полностью эта информация заносится в базу данных только при автоматизированном проектировании баз данных, при создании таблиц SQL-скриптами или мастерами оболочки базы данных, эта информация разработчиками, как правило, не вносится.

В рамках проекта недостающая информация (комментарии и ключи) была внесена вручную. При реальном применении предложенных подходов для интеграции баз данных необходимо предъявлять достаточно жесткие требования по качеству проектирования и документирования баз данных.

4.3. Модуль сопоставления. Модуль сопоставления предназначается для отработки эвристик установления соответствия таблиц базы данных и концептов онтологии предметной области. Сопоставление основывается на анализе определений концептов и слотов, с одной стороны, и комментариев к таблицам и столбцам, с другой стороны.

Сопоставление выполняется в два этапа.

Лексический этап. На этом этапе устанавливается понимание употребленной пользователем лексики. Все лексические единицы, употребленные пользователем, проверяются на наличие их в тезуарусе. При отсутствии лексической единицы в тезаурусе проверяются гипотезы об опечатке, грамматической ошибке или употреблении нестандартного сокращения. Для проверок этих гипотез используется метрика Левенсшейна или метод парных совпадений. Последний метод используется при проверке возможности употребления пользователем многословного термина с нефиксированным порядком слов. На основе вычисленных лексикографических расстояний употребленной пользователем лексемы и элементов тезауруса пользователю предъявляется несколько наиболее правдоподобных сочетаний. Пользователь может выбрать один из предложенных вариантов или зафиксировать употребленную лексему как новую.

Семантический этап. На этом этапе устанавливаются возможные соответствия таблицы базы данных и концепта логической модели. Базовый подход предусматривает выделение в текстах комментариев к таблице и ее столбцам существитель-

ных, стоящих в родительном падеже, которые проверяются по описаниям концептов как возможное имя концепта. В текстах комментариев столбцов выделяются существительные в именительном падеже, которые проверяются по определениям слотов как возможные имена слотов. Проверяется наличие у выделенного множества концептов выделенных множеств слотов. Концепты ранжируются по количеству найденных соответствий слотов и предъявляются пользователю для окончательного выбора. Исследовались некоторые модификации данного метода, в частности, рассматривался вариант, когда в таблице отсутствует имя известного концепта. В данном случае выделенное множество имен слотов проверяется на всех концептах. При поиске соответствий имен используются взаимозаменяемые элементы синсетов из тезауруса.

Проведенные эксперименты с реальными базами данных показали, что для эффективного применения предлагаемого подхода интеграции баз данных существующие описания баз данных требуют некоторой достаточно несложной формализации. Основным требованием является употребление развернутых определений свойств при определении столбцов таблиц, с обязательным упоминанием объекта, которому эти свойства принадлежат.

Заключение

Интеграция информационных ресурсов, представленных в виде реляционных баз данных, является в настоящее время весьма актуальной задачей, в связи с чем для ее решения предложено множество подходов, на основе которых разработаны промышленные интеграционные платформы, в том числе такие известные, как IBM WebSphere Integration Server[6] и SAP NetViewer [7]. Эти и другие интеграционные платформы успешно применяются в реальной практике. Вместе с тем большинство из известных подходов базируется на прямом процедурном представлении знаний о структуре интегрируемой информации, т. е. интеграционные процедуры в виде SQL-скриптов, java-классов или иных языковых компонент явно оперируют именами артефактов интегрируемых баз данных. Недостаток такого представления знаний, требуемых для интеграционных операций, заключается в повышенной трудоемкости сопровождения интеграционных процедур, связанной с необходимостью ручной модификации кода интеграционных процедур при модификации интегрируемой информации.

Предлагаемый подход основан на разделении интеграционных знаний на независимую от предметной области и структуры интегрируемых баз данных немодифицируемую процедурную компоненту и модифицируемую информационную компоненту, зависящую от предметной области и структуры интегрируемых баз данных. Процедурная компонента базируется на продукциях, инвариантных относительно структуры конкретной базы данных и состава предметной области, и после первоначальной настройки не требует модификации при изменении структур баз данных.

Известные интеграционные платформы, в частности IBM WebShphere Integration Server, предполагают участие разработчиков баз данных в написании интеграционных процедур. В реальных условиях это сильно усложняет и удорожает интеграционные проекты. Предлагаемый подход этого не требует при соблюдении разработчиками баз данных некоторых формальных правил проектирования и описания структуры базы данных.

[6] http://www-306.ibm.com/software/ru/websphere/

[7] http://www.pressebox.de/pressemeldungen/netviewer-gmbh/boxid-53076.html

Можно предполагать, что данный подход позволит существенно сократить сроки и трудоемкость интеграции реляционных баз данных на корпоративном уровне, а в перспективе возможно и в системах широкого доступа.

Данная работа выполнена при поддержке РФФИ (проект № 06-07-89219-а).

Summary

E.V. Birialtcev, A.M. Gusenkov. Based on ontology approach to integration of relation databases.

This paper is devoted to a problem of integration of relational databases. We consider main structural and lexico-semantic distinctions in representation and the description of the information of relational data which complicate integration. Distinctions in names of objects of databases are most a challenge at integration of relational data. Representation of structure of a database, subject domain and language of the description in the form of ontology can solve this problem. We offer such representation of ontology which divides the procedural knowledge independent of structure of integrated databases and descriptive knowledge, depending from their structure.

Литература

1. *Sheth A.P., Larson J.A.* Federated databases for managing distributed, heterogeneous, and autonomous databases // Computing Surveys. – 1990. – V. 22, No 3. – P. 183–236.
2. *Lomet D., Widom J. (eds).* IEEE Data Engineering Bulletin. – 1995. – V. 18, No 2.
3. *Gupta A., Mumick I.S.* Materialized Views: Techniques, Implementations, and Applications. – Cambridge: MIT Press, 1999. – 616 p.
4. *Chaudhuri S., Dayal U.* An overview of data warehousing and OLAP technology // SIGMOD Record. – 1997. – V. 26, No 1. – P. 65–74.
5. *Wiederhold G.* Mediators in the architecture of future information systems // IEEE Computer. – 1992. – V. 25, No 1. – P. 38–49.
6. *Garcia-Molina H., Papaconstantinou Y., Quass D., Rajaraman A., Sagiv Y., Vassalos V., Ullman J.D., Widom J.* The TSIMMIS approach to mediation: data models and languages // Intelligent Information Systems. – 1997. – V. 8, No 2. – P. 117–132.
7. *Papakonstantinou Y., Gupta A., Haas L.* Capabilities-base query rewriting in mediator systems: IBM Almaden Technical Report. [Электронный ресурс]. – 1995. – Режим доступа: http://dbpubs.stanford.edu/pub/1995-2, свободный.
8. *Yerneni R., Li C., Garcia-Molina H., Ullman J.D.* Computing capabilities of mediators // Proc. of ACM SIGMOD. – N. Y.: ACM Press New York, 1999. – P. 443–454.
9. *Гаврилова Т.А., Хорошевский В.Ф.* Базы знаний интеллектуальных систем.– СПб.: Питер, 2001. – 384 с.
10. *Бирюльцев Е.В., Гусенков А.М., Елизаров А.М.* О доступе к электронным коллекциям в виде реляционных баз данных на основе онтологий // Тр. девятой Всерос. науч. конф. RCDL'2007. – Переславль-Залесский: Ин-т программных систем РАН, 2007. – Т. 1. – С. 211–216.
11. *Дейт К.Д.* Введение в системы баз данных.– М.: Изд. дом «Вильямс», 2001. – 72 с.
12. *Бирюльцев Е.В., Гусенков А.М., Косинов Я.Г.* Представление структуры реляционных баз данных в формализме онтологий // Тр. Казан. школы по компьютерной и когнитивной лингвистике TEL-2006. – Казань: Отечество, 2007. – С. 32–37.

13. *Бирюльцев Е.В., Гусенков А.М.* Онтологии реляционных баз данных. Лингвистический аспект // Тр. междунар. конф. Диалог'2007, «Бекасово», 30 мая – 3 июня 2007 г. – М.: Изд. центр РГГУ, 2007. – С. 50–53.

14. Онтологии и тезаурусы: Учебн.-метод. пособие / Под ред. Б.В. Добров, В.В. Иванов, Н.В. Лукашевич, В.Д. Соловьев. – Казань: Казан. гос. ун-т, 2006. – 198 с.

15. *Буч Г., Рамбо Д., Джекобсон А.* Язык UML. Руководство пользователя / Пер. с англ. – М.: ДМК, 2000. – 432 с.

16. *Жучков А.В. и др.* Новые технологии для понятийных сетей, создаваемых в рамках МНТП «Вакцины нового поколения и диагностические системы будущего» [Электронный ресурс] // Электронные библиотеки. – 2003. – Т. 6, Вып. 6. – Режим доступа: http://www.elbib.ru/index.phtml?page=elbib/rus/journal/2003/part6/ZATGS, свободный.

17. *Андерсон Дж.* Дискретная математика и комбинаторика / Пер. с англ. – М.: Изд. дом «Вильямс», 2003. – 960 с.

18. *Fellbaum C. (ed.)* WordNet: An Electronic Lexical Database. – Cambridge: MIT Press, 1998. – 423 p.

19. *Vossen P.* Building a multilingual database with wordnets for several European languages [Электронный режим]. – Режим доступа: http://www.illc.uva.nl/EuroWordNet/, свободный.

20. *Бирюльцев Е.В., Гусенков А.М., Галимов М.Р.* Особенности лексико-семантической структуры наименований артефактов реляционных баз данных // Тр. казанской школы по компьютерной и когнитивной лингвистике TEL-2005. – Казань: Казан. гос. ун-т, 2006. – С. 4–12.

21. *Льюис Ф., Розенкранц Д., Стирнз Р.* Теоретические основы проектирования компиляторов. – М.: Мир, 1979. – 654 с.

22. *Бирюльцев Е.В., Гусенков А.М., Хайруллина А.И.* Представление модели данных Epicenter POSC на языке онтологий OWL // Тр. Казан. школы по компьютерной и когнитивной лингвистике TEL-2006. – Казань: Отечество, 2007. – С. 38–49.

23. *Бирюльцев Е.В., Гусенков А.М.* Построение онтологии предметной области на основе логической модели баз данных // Тр. Всерос. конф. с международным участием «Знания-Онтологии-Теории» (ЗОНТ-07). – Новосибирск: Ин-т математики им. С.Л. Соболева СО РАН, 2007. – Т. 1. – С. 176–183.

Поступила в редакцию
25.07.07

Бирюльцев Евгений Васильевич – кандидат технических наук, заведующий лабораторией технологии баз данных НИИ математики и механики им. Н.Г. Чеботарева Казанского государственного университета.

E-mail: *ltbd@ksu.ru*

Гусенков Александр Михайлович – старший научный сотрудник лаборатории технологии баз данных НИИ математики и механики им. Н.Г. Чеботарева Казанского государственного университета.

E-mail: *Alexandr.gusenkov@ksu.ru*