Разработка прикладных систем генерации типовых текстов на ЕЯ на основе представления информации на языке XML

М.В. Болдасов
boldasov@mn.ru

Излагаются предлагаемые автором методы разработки прикладных систем генерации типовых описательных текстов на нескольких ЕЯ из формального описания излагаемой информации в формализме XML. Предложенные методы генерации реализованы в системой разработки и эксплуатации генераторов текстов DEMLinG (Development Environment for Multilingual Generators). Для описания лингвистических знаний, необходимых на этапах генерации, в рамках этой системы были разработаны соответствующие языки, основные особенности которых обсуждаются в статье. Данная статья является результатом практической работы, выполненной автором по созданию системы генерации текстов естественно-языкового представления SQL запросов к БД. Рассмотренные в этой работе методы разработки прикладных систем генерации легли в основу технологии построения легко настраиваемых систем генерации на каждую новую БД, с максимальным использованием ресурсов генерации уже разработанных систем генерации, работающих с другими БД.

1. Введение

Среди проблем работы рядового пользователя с базой данных (БД), пожалуй, наиболее сложным является вопрос, как предоставить ему удобное средство доступа к БД. Использование для этих целей фиксированной иерархии таблиц с простыми средствами сортировки и фильтрации по значениям атрибутов этих таблиц часто не позволяют пользователю эффективно решить поставленную задачу. Использование формальных языков запросов к БД (например, языка SQL) может решить проблему доступа к БД лишь при наличии у пользователя достаточной квалификации в работе языками запросов. Применение систем автоматической обработки текстов (АОТ) позволяет пользователю обращаться к БД на своем родном языке. Для этого к БД присоединяется программный модуль естественно-языкового интерфейса, предварительно интерпретирующий запрос пользователя с естественного языка на язык запросов SQL.
В связи с неоднозначностью ЕЯ, а также с возможным неточным пониманием пользователем того, какая информация хранится в БД, ЕЯ-интерфейсы не всегда адекватно переводят запрос пользователя в язык SQL или на формальном языке оказывается выраженным не то, что имел в виду пользователь. Одно из решений этой проблемы – предоставить пользователю средство контроля правильности «понимания»

его запроса ЕЯ-интерфейсом. Таким средством является генератор ЕЯ представления SQL запроса.

В данной статье обсуждается вопрос построения генератора QGen (Query Generator) для ЕЯ-интерфейсов, разработанных в системе InBASE. Система InBASE была создана в Институте Искусственного Интеллекта (РОС НИИ ИИ). Она предлагает набор инструментальных средств для разработки ЕЯ интерфейсов к базам данных, реализующих «семантически-ориентированный» подход [Нариньяни 1979], согласно которому ЕЯ-интерфейс настраивается отдельно для каждой базы данных. ЕЯ-интерфейс состоит из «семантически-ориентированной грамматики» (едина для всех интерфейсов), словаря типов извлекаемых семантических понятий и модели предметной области (МПО), определяющей используемые понятия связи между ними [Sharoff, Zhigalov 1999]. К настоящему моменту ЕЯ интерфейсы InBASE работают с различными БД, обрабатывая пользовательские запросы на русском, английском и немецком языках.

Идея использования модуля генерации для контроля правильности составления формального запроса к БД не нова. Автору известны такие системы, как CO-OP [McKeown et al. 1983], Remit [Lowden et al. 1986], [Minock 2003]. На входе эти системы принимают представление, полученное в результате синтаксического анализа запроса пользователя (CO-OP [McKeown et al. 1983]) или формулу реляционного исчисления (Relational Calculus, RC) как наиболее общее формальное представление запроса (REMIT [Lowden et al. 1986] и [Minock 2003]). В своем подходе я использую входное представление OQL, построенное системой InBASE в автоматическом режиме. Исходный ЕЯ запрос пользователя в процессе генерации никак не используется. Таким образом, я отказываюсь от использования лингвистической информации, содержащейся в запросе пользователя системы InBASE, и руководствуемся только формальным языконезависимым представлением OQL. Выбор этого представления как входного для генераторов обусловлен двумя факторами. С одной стороны, это представление (в отличие от результирующего представления разбора запроса пользователя SQL) структурировано языком XML и не требует дополнительного разбора или преобразования к специальному формату (такому как, RC). С другой стороны, оно опирается на концепты модели предметной области пользователя. Поэтому оно ближе к человеческому восприятию БД нежели чем результирующее SQL представление, формулируемое согласно понятиям БД..

Создаваемое генераторами QGen ЕЯ представление OQL запроса выражается распространенной именной группой. Выбор именно такой формы обусловлен, с одной стороны, ее структурной близостью с представлением OQL (сравни с конструкцией «[modifier] head [complement]» в [Minock 2003]), и, с другой стороны, небольшим количеством конструкций, необходимых для грамматического описания создаваемого высказывания. Кроме того, текст, записанный в такой форме, стилистически хорошо подходит в качестве заголовка к результатам из БД, предъявляемых пользователю по его запросу на ЕЯ:

Input OQL query:
```
SELECT Surname, Name, Position FROM Employee WHERE Position LIKE 'president'
```

Result of generation in English:
```
First and last names of employees with position president.
```

Result of generation in Russian:
```
Имена и фамилии сотрудников в должности директор.
```

Рис. 1. Пример генерации системами QGen высказываний в форме распространенных именных групп из формального OQL запроса к БД, составляемого ЕЯ-интерфейсом системы InBASE.

Система, рассмотренная в статье [Minock 2003], использует для генерации эмпирический подход, реализованный в грамматике фразового лексикона. Главный недостаток этого подхода – слабая адаптивность генератора к новой БД. Я использую более лингвистически мотивированный подход и предлагаем средства его поддержки, обсуждаемые в этой статье.

2. Решения, используемые в системе QGen

На вход системе QGen передается SQL запрос, или, точнее, его SELECT-форма. SELECT-запрос состоит из SELECT-, FROM-, WHERE- и ORDER_BY-частей, и может содержать функторы AVG, COUNT, MAX, MIN, SUM. В WHERE-части запроса могут появляться следующие операции сравнения: “=”, “>”,”<”,”>=”, “<=”, “LIKE”.
Как было отмечено еще в статье [Lowden et al. 1986], выражения на формальных языках бедны концептуальной информацией о предметной области конкретной базы данных. Если генератор производит тексты, полезные для неподготовленного пользователя, такая концептуальная информация добавляется им в результирующий текст в процессе генерации. Значит, система QGen должна использовать в процессе работы над порождаемым текстом концептуальную информацию, уникальную для конкретной БД, то есть зависеть от выбора БД. Поэтому, как и систему InBASE, систему QGen необходимо настраивать на новую БД.
Создаваемый генератор должен быть многоязыковым. Это означает, что он должен уметь производить тексты на различных языках в зависимости от того, какой язык был использован пользователем ЕЯ интерфейса. К разрабатываемому генератору выдвигаются требования легкой адаптивности к смене модели предметной области и языка генерации.
Процесс генерации в системе QGen организуется в виде однонаправленного конвейера: задача построения результирующего текста на ЕЯ разбивается на этапы, на каждом из которых решаются конкретные лингвистические задачи планирования и оформления порождаемого текста. Поэтому для внесения изменений в работу генератора требуется изменить содержание только отдельных этапов генерации – полной переработки генератора не требуется.
Особенности решаемой задачи дополнительно требуют также высокой скорости работы генератора, его небольших размеров и легкости разработки и поддержки этого генератора. Требования небольших размеров генератора и высокой скорости его работы не позволяют воспользоваться готовыми модулями реализации семантического плана высказывания, такими как, например, KPML [Bateman 1996], RealPro [Lavoie et al. 1997] и т.д. Поэтому в данной работе предлагается облегченный вариант этого модуля, осуществляющего языковое оформление высказывания, спланированного в виде синтаксической структуры непосредственных составляющих.

Процесс генерации организуется в системе QGen в виде однонаправленного конвейера трансформаций входного представления (по-английски этот метод называется sequential graph-rewriting [Bohnet et al. 2001]) и описывается последовательностью этапов. Трансформации, возможные на каждом этапе, определяются в системе набором продукционных правил. В результате применения ресурсов генерации строится структурное представление, соответствующее синтезированному тексту. Процесс генерации завершается сбором результирующей строки текста в результате обхода обрабатываемой структуры данных сверху вниз слева направо с обращением для каждого лексически значимого узла этой структуры к внешнему модулю морфологического оформления с целью получения для узла соответствующих морфологически оформленных словоформ.

Реализуемая схема генерации

Для описания последовательности этапов генерации в системе QGen обратимся к схеме, представленной на рис. 2, которая была составлена мной на основе общей схемы процесса генерации, предложенной в [Bateman & Zock 2001].

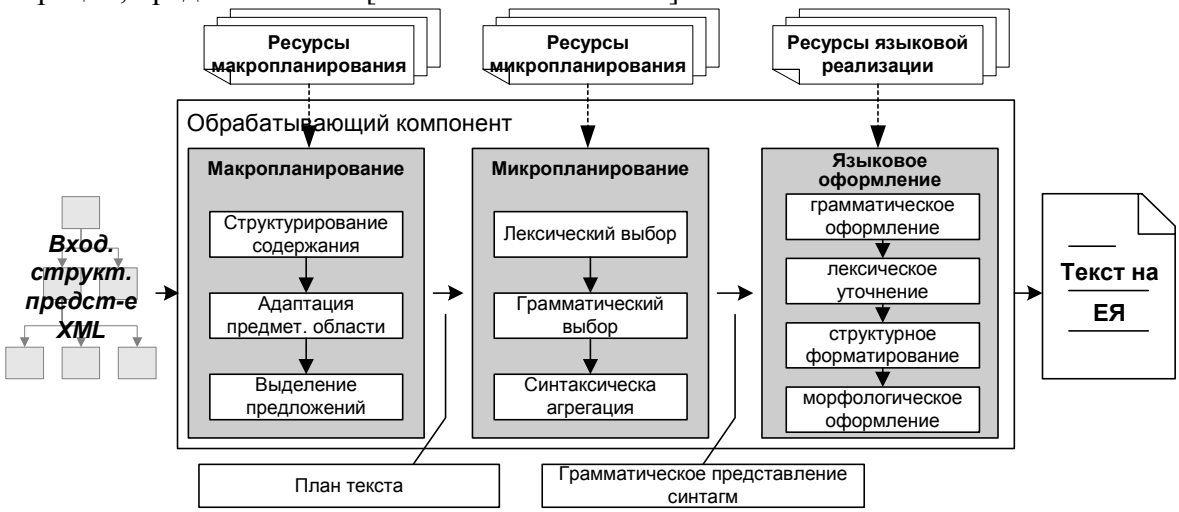

Рис. 6. Модифицированная схема генерации, поддерживаемая создаваемыми инструментальными средствами

На этапах Структурирования содержания, Адаптации к предметной области и Выделения предложений входная структура OQL преобразуется в структуру порождаемого текста. В результате формируется концептуальное представление, представляющее собой план изложения в создаваемом тексте. План текста записывается в терминах структуры непосредственных составляющих, в которой специальным образом помечаются узлы, ограничивающие отдельные предложения.

Для повышения качества генерируемого текста необходимо учитывать различия между представлением информации в модели предметной области (МПО) базы данных и МПО пользователя этой БД. Эта задача решается на этапе Адаптации к предметной области. Некоторые конструкции, типичные для МПО БД, являются неудачными для МПО пользователя. Например, типичный для МПО БД фрагмент представления входного запроса *compare('домашний телефон', '=', '')*, который может быть выражен на ЕЯ как "*с домашним телефоном '*", в МПО пользователя должен быть преобразован в конструкцию *compare(not(''домашний телефон'),,)*, выражаемую на ЕЯ как "*без домашнего телефона*". Два следующих этапа генерации - Лексический и Грамматический выбор, а также этап Синтаксической агрегации, осуществляют переход от концептуального представления к

грамматическому. В генераторе QGen такой переход описывается двумя взаимозависимыми процессами: выбором грамматической структуры и лексическим выбором.
На этапе Лексического выбора генератор работает с концептами МПО, сопоставляя им лексические единицы, наиболее точно отражающие эти концепты в окружающем их контексте. Вставленные лексические единицы дополнительно классифицируются семантическими, грамматическими и морфологическими свойствами, что влияет как на последующие лексические сопоставления, так и на дальнейшее построение грамматической структуры высказывания. На этапе Грамматического выбора лексико-семантисизированная концептуальная структура преобразуется в грамматическую. Преобразование базируется на проведенных на предыдущем этапе лексическом выборе и семантической классификации; преобразование выполняется снизу вверх: от лексикализованных вершин концептуального представления к вершинам, органичивающим в этом представлении отдельные предложения.
На этапе агрегации углубляется уже первоначально заданная в структуре OQL агрегация типа «общий член». Результирующая генерация соответствуют генерации типа «общая структура» согласно классификации, проведенной в работе [Reiter & Dale 2000]. Например, грамматическая структура, описывающая ЕЯ предложение "*сотрудники, имеющие двух детей или имеющие трех детей* " агрегируется к структуре, описываемой ЕЯ предложением "*сотрудники, имеющие двух или трех детей*", а структура, выражаемая на ЕЯ как "*рабочие телефоны и домашние телефоны сотрудников*", агрегируется к "*рабочие и домашние телефоны сотрудников*".
Процесс генерации завершается языковым оформлением построенного грамматического представления, осуществляемого на этапах грамматического оформления, лексического оформления, и структурного форматирования. Языковое оформление включает в себя решение вопросов управления и согласования в грамматических группах, уточнения лексических единиц создаваемого высказывания, их морфологической реализации, а также расстановки знаков пунктуации.

Языки описания действий, проводимых на этапах генерации

Для задания преобразований, проводимых на выделенных этапах генерации, автором статьи были разработаны языки их описания. Анализ разработанной схемы генерации показал, что в системе выделяются три основные вида деятельности: структурные преобразования данных, выбор слов в предложениях создаваемого текста и согласование и упорядочение слов в предложении. Каждому виду деятельности был сопоставлен своей язык его описания: язык планирования, язык лексического выбора и язык грамматического оформления. Действия по преобразованию обрабатываемых данных на любом из этих языков задаются продукционными правилами, которые применяются к фрагменту обрабатываемых данных, ограниченному узлом применения правила, при выполнении дополнительных условий, заданных в этих правилах (дополнительных ограничений на их применение). Языки различаются деталями своего синтаксиса и набором возможных действий.
Язык словаря описывает продукционные правила, сопоставляющие лексически значимому объекту модели предметной области генератора, слово или словосочетание на ЕЯ. Проводимые на этом этапе преобразования имеют две природы: регистр (такие преобразования задаются правилами, подбирающими слова, например, для операций сравнения) и МПО (описывается правилами, подбирающие слова, например, для таблиц баз данных и их атрибутов).

Выбор слова влияет на создаваемую грамматическую конструкцию. Поэтому, сопоставляемому слову могут быть добавлены ограничения на грамматическую реализацию, ограничения на участие сопоставляемого слова в грамматических конструкциях. Эти ограничения записываются в виде свойств, понимаемых грамматикой – ресурсами Грамматического выбора и Грамматического оформления. Количество этих свойств фиксировано. Оно может быть расширено при расширении грамматического ресурса языка.

Для реализации возможности использования синонимии при построении текста, левая часть правила может быть расширена указанием дополнительных ограничений на контекст использования данного словарного правила. В этом случае одному и тому же объекту МПО могут соответствовать разные слова и словосочетания, употребимые в разных контекстах.

Приведем в качестве примера словарное правило, описывающее сопоставление слова «ребенок» атрибуту с индексом «12» таблицы, описывающей сотрудников некоторого предприятия.

```
[12] (^node(where)) ~> ребенок (prep:S, SF:quantitative_rel)
```

Статья применима, если атрибут указан в WHERE-части SQL запроса. Ограничения на грамматическую реализацию предписывают использовать существительное «ребенок» внутри предложной группы с предлогом «*с*», и указывают на возможность присутствия количественной характеристики этого слова.

Язык планирования описывает продукционные правила, задающие действия структурного преобразования фрагмента обрабатываемых данных, ограниченных узлом применения правила. К таким действиям относятся добавление, изменение и удаление элементов обрабатываемой структуры данных, изменение свойств этих элементов, а также объединение элементов структуры данных в грамматические группы. Обрабатываемые узлы выбираются с помощью механизма сопоставления с образцом. Результат сопоставления назначается идентификатору, уникальному в пределах данного правила. В левую часть правила могут быть добавлены дополнительные ограничения на его применение.

Приведем в качестве примера правило, планирующее грамматическую группу Posessive, состоящую из двух членов: *N* (*Noun*) и *dep* (*Dependent*), с которыми сопоставлены соответственно узлы *selectItem* и *fromItem* запроса *query*.

```
[query](selectItem.defined() & fromItem.defined()) ~>
group (#this Posessive) members (> selectItem N) (fromItem dep);
```

Главным членом назначается узел *selectItem*. Правило применимо, если в обрабатываемых данных заданы узлы *selectItem* и *fromItem.* Создаваемая группа описывает такие словосочетания, как, например, *Зарплата сотрудника* или *Стаж сотрудника*.

Язык грамматического оформления описывает продукционные правила, задающие языковое оформление грамматических групп. Правило описывает члены группы. Сопоставление с узлом в структуре осуществляется по имени члена группы. Левая часть строки описания члена группы задает его позицию в группе, правая часть описывает его согласование с другими членами группы, а также с внешним контекстом группы.

Приведем в качестве примера правило, описывающее грамматическую группу Attributive2.

```
[attributive_2]
{
  dep(n) = (gender:n, number: n, case:n),
```

```
    >n = (number:^, case:^)
}
```

Этим правилом описывается реализация одного из видов определительного отношения. Главный член представлен существительным, он занимает второе место в синтагме и наследует от общей составляющей групповые признаки числа и падежа. Зависимый член согласуется с ним по роду, числу и падежу. Описанное правило работает как для русского языка, так и для английского (сравни: ru: «замужняя сотрудница», en: «married employee»; ru: «максимальная зарплата», en: «maximal salary»).

Среда разработки и эксплуатации генераторов

Интерпретаторы и отладчики предлагаемых языков были реализованы в единой среде разработки и эксплуатации генераторов текстов на ЕЯ. Эта среда была названа DEMLinG (Development Environment for MultiLingual Generators). Среда DEMLinG была реализована на языке MS Visual J++. Архитектурно система DEMLinG разделяется на вычислительный компонент, осуществляющий ЕЯ генерацию, и один или несколько клиентов. Клиенты могут быть другими приложениями, использующими виртуальную машину системы DEMLinG, или интерфейсами, такими как среды разработки ресурсов генераторов, веб-интерфейсы работы генераторов с конечным пользователем, интерфейсы, интегрирующие генератор в другое программное приложение и т.д.
Вычислительный компонент системы DEMlinG включает в себя интерпретаторы и отладчики языков планирования, лексического выбора и грамматического оформления. Дополнительно, вычислительный компонент поддерживает язык описания сценария генерации, позволяющий описывать последовательность задач, решаемых в процессе генерации. Виртуальная машина предусматривает также возможное подключение к ней через COM-интерфейс модулей генерации, реализованных в других системах (например, модуля языкового оформления, реализованного в системе KPML [Bateman 1996], или модуля морфологического оформления, реализованного фирмой Dialing (URL: www.aot.ru)).

3. Заключение

В статье предлагается новый подход генерации на ЕЯ, расчитанный на создание небольших генераторов, настроенных на определенный тип текстов и определенный тип входных данных. Описанный подход применен к задаче генерации описаний на естественном языке OQL запросов к БД, порождаемых ЕЯ-интерфесами системы InBASE. Схема генерации, реализованная в генераторах QGen, имеет две основных особенности по сравнению с общей схемой Бейтмана – Зока [Bateman & Zock 2001]:

- В процессе генерации не создается промежуточного семантичесокого представления. Входное представление преобразуется сразу к грамматическому. В отличие от больших систем языкового оформления (таких как KPML [Bateman 1996]), предложенный мной подход при выборе грамматических и лексических средств выражения в создаваемом тексте использует в основном когнитивные, а не семантические свойства
- Ресурсы этапа языкового оформления в предложенной схеме генерации работают с грамматическими структурами, представленными в виде иерархии грамматических зависимостей. Это представление создается на этапе грамматического выбора продукционными правилами, определяющими каскадные шаблоны для выбранного регистра входного концептуального представления и типа создаваемого текста.
- Поскольку формальное представление для генератора не несет в себе лингвистической инфомрации, большое внимание в предложенной схеме уделяется лексическому выбору. На этом этапе в обрабатываемую структуру текста вносится дополнительная семантическая информация,

классифицирующая значимые объекты, используемая при принятии решений по организации грамматической структуры. Такой подход позволяет разработчику генератора иметь большинство настроек процесса генерации в одном месте – в ресурсе Лексического выбора. Это позволяет сконцентрировать все работы по настройке нового генератора на настройке словаря, что значительно упрощает процесс его создания.

Опыт работы над созданием системы QGen показал, что предлагаемый мной подход близок с точки зрения его многоязыковости к подходу, использующему для генерации промежуточное семантическое представление. Разработанные к настоящему моменту генераторы, работающие с двумя базами данных (кадров предприятия и магазина автомобилей) и с двумя языками (русским и английским), используют по большей части общие ресурсы генерации. Для новой базы данных и нового языка заново приходится создавать только ресурсы лексического выбора и адаптации предметной области. Общий объем реализации ресурсов в системе QGen генераторов составил 508 правил. Из них 303 правила словарных правила, 19 правил грамматического оформления и 186 – правил планирования. Опыт разработки генераторов системы QGen показал, что настройка генератора на новую БД занимает несколько дней работы одного или двух человек – лиц с навыками программирования и базовыми знаниями в лингвистике.

Область дальнейших исследований в этом направлении видится автором как накопление опыта применения системы DEMLinG в задаче InBASE и расширение области применения системы DEMLinG на решение других задач генерации.

Литература

1. Нариньяни А.С. 1979. Лингвистические процессоры ЗАПСИБ. *Препринт N 199.* Вычислительный центр СО АН СССР. Новосибирск.
2. Sharoff S., Zhigalov V.: Register-domain separation as a methodology for development of NL Interfaces to Databases // Human-Computer Interaction - INTERACT'99. Angela Sasse and Chris Johnson (eds.) Published by IOS Press, 1999
3. McKeown, Kathleen. 1983, Paraphrasing Questions Using Given and New information American Journal of Computational Linguistics 9(1). 1983: 1--10
4. Barry G. T. Lowden, Anne N. De Roeck: The REMIT System for Paraphrasing Relational Query Expressions into Natural Language. VLDB 1986: 365-371
5. Minock, M. "A phrasal generator for describing relational database queries", Proc. of the 9th EACL worshop on natural language generation, Budapest, Hungary, Apr 2003.
6. Bateman, J. KPML Development Environment. Technical Report, IPSI, GMD, Darmstadt, Germany. 1996.
7. Lavoie, B. and Rambow, O. A Fast and Portable Realizer for Text Generation Systems. In Proceedings of the Fifth Conference on Applied Natural Language Processing, 1997: pages 265-268.
8. Bernd Bohnet and Leo Wanner. On Using a Parallel Graph Rewriting Grammar Formalism in Generation. Proceedings of the 8th European Natural Language Generation Workshop at the Annual Meeting of the Association for Computational Linguistics, Toulouse, 2001.
9. John Bateman and Michael Zock. Natural Language Generation, R. Mitkov (Ed.) Oxford University Press, Oxford 2001.
10. Reiter, E. and R. Dale 2000. Building Natural Language Generation Systems. Cambridge University Press.
11. Kasper, R. A flexible interface for linking applications to Penman's sentence generator. In Proceedings of the DARPA Speech and Natural Language Workshop, 1989: pages 153-158.