ПОДХОД К ПОЛУАВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ АДАПТЕРОВ В ПОСРЕДНИКЕ НЕОДНОРОДНЫХ КОЛЛЕКЦИЙ ЭЛЕКТРОННЫХ БИБЛИОТЕК

Осипов М.А Брюхов Д.О. Калиниченко Л.А. Леонтьев И.В.
osipov@newtech.ru brd@synth.ipi.ac.ru leonidk@synth.ipi.ac.ru ileon@synth.ipi.ac.ru
Институт Проблем Информатики Российской Академии Наук
117333, Москва, ул. Вавилова 44-2

Аннотация

В работе[1] обсуждается подход к полуавтоматической генерации адаптеров для подключения неоднородных коллекций электронных библиотек к их посредникам. Рассматривается общая архитектура адаптеров, функции их компонентов, сценарий работы, а также основные действия, необходимые для их генерации.

1. Введение

Основной идеей предлагаемой инфраструктуры интегрированного доступа к неоднородным информационными источникам является введение промежуточного слоя между электронными коллекциями данных и потребителями информации. Основными компонентами промежуточного слоя являются информационные посредники, существующие независимо от электронных коллекций. Посредники обеспечивают унифицированный интерфейс запросов к многочисленным источникам данных и освобождают пользователя от необходимости находить подходящую коллекцию, осуществлять в ней поиск требуемой информации и вручную объединять информацию, полученную из различных коллекций. В рамках рассматриваемого проекта в качестве канонической модели данных в посреднике выбрана модель данных языка СИНТЕЗ [3]. В качестве языка запросов используется язык SOQL, который является расширением языка запросов OQL для объектной модели данных с изменениями и дополнениями, обусловленными спецификой модели данных СИНТЕЗ.

Важным элементом архитектуры посредника являются адаптеры. Они обеспечивают преобразование запросов посредника, выраженных относительно федеративной схемы на каноническом языке запросов SOQL, в запросы на языке, понятном коллекции. Адаптеры также обеспечивают получение результата запроса от коллекции и преобразование результата из модели данных коллекции в каноническую модель данных посредника. В рамках проекта было создано несколько адаптеров для различных моделей данных (SRS, XML, Irbis) [4, 5].

После получения глобального запроса от пользователя, посредник осуществляет декомпозицию данного запроса в несколько локальных запросов и составляет план выполнения запросов. В соответствии с планом, посредник посылает каждый локальный запрос соответствующему адаптеру. Подробную информацию о структуре посредника можно найти в [2].

Адаптер получает от посредника запрос в форме SELECT/FROM/WHERE, выраженный на языке запросов SOQL. Так как разные коллекции обладают различными способностями (например, из-за различной мощности языков запросов), то, очевидно, что для некоторых коллекций невозможно реализовать все запросы на языке SOQL. В связи с этим посредник может задавать различные подмножества запросов языка SOQL для различных коллекций. Взаимодействие посредника с адаптером происходит посредством интерфейса JDBC.

В данной работе исследуется проблема полуавтоматической генерации адаптеров, целью которой является упрощение процесса подключения новых коллекций к посреднику. Опыт разработки адаптеров вручную показал, что адаптеры имеют ряд общих структурных элементов, которые могут быть обобщены и использованы при конструировании различных адаптеров. Некоторые компоненты выполняют одинаковые функции (например, преобразование запросов и результата, полученного от коллекции), которые заслуживают автоматизации их генерации. Задача полуавтоматической генерации адаптера разбивается на несколько подзадач:

- определение структуры адаптера и выделение в ней компонентов, являющихся обобщаемыми для многих адаптеров;
- разработка методов реализации компонентов, которые являются специфическими для конкретных адаптеров;
- выделение компонентов, которые могут быть созданы на основе конкретизируемых спецификаций, выраженных на специальном языке;
- определение архитектуры и интерфейсов адаптера (внешних и внутренних).

2. Анализ существующих подходов

В настоящее время существует несколько проектов, в рамках которых решается задача интеграции разнородных информационных коллекций (например, TSIMMIS, DISCO [10], MIX, Meta-Wrapper [8]). В этих проектах используются различные модели данных в качестве канонической (OEM, XML, объектная модель), различные подходы к описанию способностей коллекций.

Например, в проекте Disco в качестве канонической модели данных используется объектная модель, а в качестве языка запросов – язык OQL. В этом проекте запросы, посылаемые адаптерам, формулируются не на языке OQL, а в форме комбинации операций selection, projection и join. Язык спецификации способностей коллекции в данном случае описывает всевозможные комбинации этих операций, выполнимые над конкретной коллекцией.

В проекте MIX в качестве канонической модели данных используется модель данных XML. В этом проекте не предусмотрено использование спецификаций способностей языка запросов конкретной коллекции, и адаптер должен выполнять все запросы к коллекции. Если же адаптер не может выполнить какой либо запрос, то он возвращает пустое результирующее множество. Таким образом, результатом запроса может быть неполное множество объектов, удовлетворяющих запросу.

Полуавтоматическая генерация адаптерных функций используется в проекте Araneus. В этом проекте создан инструмент Minerva [1], который сочетает декларативный подход к описанию структур данных в коллекциях в виде грамматики с элементами структурного программирования для описания действий по преобразованию данных.

Подход, изложенный в данной работе, наиболее близок к методу [6, 7], используемому в проекте TSIMMIS, с изменениями и расширениями, обусловленными спецификой модели данных языка СИНТЕЗ. В проекте TSIMMIS в качестве канонической модели данных используется модель данных OEM (Object Exchange Model). Объекты в OEM представляют собой тройки: идентификатор объекта, метка, значение. В качестве языка запросов к OEM используется язык MSL. Запросы на MSL являются шаблонами объектов OEM. Способности языка запросов конкретной коллекции задаются путём определения шаблонов MSL-запросов; с каждым шаблоном может быть связано действие на языке C. Набор шаблонов без связанных с

[1] Данная работа выполняется в рамках проектов РФФИ: (гранты 01-07-90084 и 00-07-90086)

ними действий составляет спецификацию множества запросов, выполняемых над данной коллекцией. В проекте есть разделение спецификаций на два класса: зависимые от схемы и независимые. Если же с шаблонами связаны действия по преобразованию запросов, то данная спецификация является зависимой от схемы коллекции, т.к. преобразование имён происходит непосредственно в блоке действий. В методе, изложенном в настоящей работе, данная проблема решается за счёт использования трансформационной схемы коллекции.

Перечисленные подходы предназначены для класса весьма ограниченных канонических моделей и не предлагают максимальной автоматизации процесса создания адаптеров. Настоящая работа ориентирована на восполнение этих пробелов.

3. Структура адаптеров

На рисунке 1 изображена структура адаптеров; здесь закрашенными прямоугольниками изображены компоненты адаптера, создаваемые полуавтоматически.

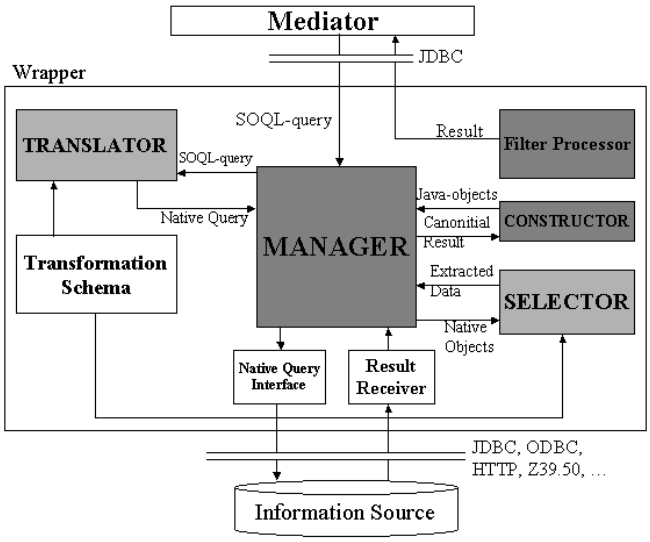

Рисунок 1 Архитектура адаптера

В набор спецификаций, на основе которых создаются компоненты адаптера, входят: спецификация множества запросов на языке запросов SOQL, потенциально реализуемых адаптером; спецификация правил преобразования SOQL-запросов в запросы к коллекции; спецификация компонента адаптера SELECTOR, обеспечивающего преобразование результата запроса к коллекции в каноническую форму. Компонентами адаптеров, создаваемых на основе данных описаний, являются компоненты TRANSLATOR и SELECTOR.

Некоторые компоненты предлагаемой структуры адаптеров являются унифицированными для всех адаптеров, созданных по предлагаемой схеме. Это компоненты TRANSLATOR, Filter Processor, MANAGER.(последний является унифицируемым лишь отчасти).

3.1 Компоненты адаптера

MANAGER

Это - центральный компонент в архитектуре адаптера. В его функции входит управление остальными компонентами и хранение промежуточных данных. MANAGER получает посредством JDBC запрос на языке SOQL. Он имеет доступ к трансформационной схеме данной коллекции. Доступ к ней от остальных компонентов адаптера осуществляется посредством обращений к MANAGER.

TRANSLATOR

Компонент TRANSLATOR адаптера преобразует запросы на языке SOQL в запросы на языке запросов системы управления коллекцией. Правила, по которым TRANSLATOR преобразует запрос, содержатся в спецификации способностей коллекции. TRANSLATOR определяет также возможность выполнения входящего запроса. TRANSLATOR генерируется специальным инструментом на основе спецификации способностей языка запросов коллекции.

Native Query Interface и Result Receiver

Эти компоненты призваны выполнить запрос к коллекции и получить результат запроса. Они не генерируются автоматически и реализуют стандартный интерфейс взаимодействия адаптера и системы управления коллекцией.

SELECTOR

Задачей компонента SELECTOR является выделение необходимых данных из результата запроса к коллекции и их преобразование в каноническую форму. Данный компонент создаётся при помощи Minerva на основе специального описания.

CONSTRUCTOR

Компонент CONSTRUCTOR создаёт Java-объекты, являющиеся Java-представлением объектов канонической модели данных посредника. Т.к. этот компонент получает от MANAGER'а объекты коллекции в "удобном" виде, а также список Java-классов, то этот компонент должен последовательно вызывать конструкторы этих классов и составить массив объектов, который передаётся MANAGER'у.

Filter Processor

Компонент Filter Processor получает от компонента TRANSLATOR фильтр, который он должен применить к множеству Java-объектов, полученных от компонента MANAGER. Фильтр является условием, которому должны удовлетворять объекты результата, в фильтр входит список атрибутов, на которые осуществляется проекция. Фильтры выражаются на языке запросов SOQL. Для определения фильтра используется подмножество SOQL, а именно запросы вида:

```
SELECT <список атрибутов> FROM <результат> WHERE <условие>
```

3.2 Сценарий работы адаптера

Последовательность действий адаптера после получения запроса состоит в следующем: SOQL-запрос передаётся компоненту MANAGER адаптера; MANAGER инициализирует компонент TRANSLATOR и ожидает результата преобразования SOQL-запроса. MANAGER может получить следующий SOQL-запрос, который будет выполняться адаптером параллельно с предыдущим и т.д.

После получения запроса и трансформационной схемы от компонента MANAGER, TRANSLATOR создаёт объект QueryObject, который содержит последовательность терминальных символов грамматики, определяющей множество запросов к адаптеру, и начинает последовательно применять к данной строке символов грамматические правила, содержащиеся в спецификации способностей коллекции. Таким образом, выбирается шаблон, которому соответствует данный запрос. Во время построения дерева разбора TRANSLATOR выполняет действия, связанные с нетерминальными символами грамматики, определяющей множество допустимых SOQL-запросов. В результате выполнения этих действий компонент получает запрос к коллекции. После этого TRANSLATOR сообщает компоненту MANAGER об окончании работы и передаёт полученный запрос.

В том случае, если SOQL-запрос не может быть выполнен коллекцией непосредственно, но TRANSLATOR может создать специальные фильтры, после применения которых к результату какого либо выполнимого запроса будет получен результат SOQL-запроса, то в этом случае выполняется такой выполнимый запрос и полученные фильтры передаются компоненту Filter Processor. В этом случае посылается сигнал MANAGER'у об изменении запроса и передаётся изменённый запрос.

Далее преобразованный запрос через Native Query Interface передаётся конкретной электронной коллекции. В свою очередь Result Receiver получает результат выполнения данного запроса коллекцией. После получения результата MANAGER инициализирует компонент SELECTOR, который выбирает из результата необходимые данные (например, SELECTOR может убрать символы разметки, которые не несут никакой информации) и преобразует их в каноническую форму для передачи компоненту MANAGER, который передаёт данные в канонической форме компоненту CONSTRUCTOR.

В соответствии с семантикой запроса и данными, полученными из трансформационной схемы, CONSTRUCTOR составляет результирующее множество Java-объектов, являющихся Java-представлением объектов СИНТЕЗа. К полученным Java-объектам применяются фильтры, созданные в результате работы компонента TRANSLATOR. Компонент Filter Processor реализует фильтрацию объектов. После применения фильтров, компонент передаёт окончательный вариант результирующего множества посреднику, а также сообщает компоненту MANAGER об окончании работы. После этого компонент MANAGER заканчивает сессию по данному SOQL-запросу.

Некоторые преобразования, связанные, например, с несоответствием имён в схеме посредника и конкретной коллекции, составляют трансформационную схему для данной коллекции и могут быть получены адаптером посредством определённого интерфейса. Трансформационная схема используется компонентами адаптера TRANSLATOR, SELECTOR и Filter Processor в процессе преобразования запросов и результата.

4. Спецификация способности языка запросов (query capabilities)

Как уже говорилось ранее, различные коллекции обладают различными способностями. В первую очередь это касается различных способностей языков запросов. Для описания подмножества выполнимых адаптером запросов предлагается подход, основанный на описании возможных запросов в виде шаблонов, определённых грамматическими правилами. Спецификация возможных запросов является набором грамматических правил, определяющих нетерминальные символы контекстно-свободной грамматики. Подмножество SOQL-запросов определяют особенные символы грамматики с идентификатором Query.

Спецификация множества запросов состоит из набора троек. Первая часть каждой тройки – это правило вывода грамматики, в левой части которого стоит нетерминальный символ, а в правой определение этого символа в виде последовательности терминальных и нетерминальных символов грамматики. Вторая часть тройки является списком метапредикатов, которые являются дополнительным условием, которому должна удовлетворять строка, соответствующая данному правилу. Третья часть – это блок действий, которые выполняются в случае соответствия строки символов данному правилу.

Правила вывода

Правила грамматики используются для задания множества выполняемых над коллекцией запросов. Каждое правило состоит из двух частей: нетерминального символа и определения этого символа: Пример правила вывода:

```
Query ::= SELECT $X FROM $X IN &Path WHERE _Cond($X)
```

Метапредикаты

Для задания дополнительных условий, которым должен удовлетворять шаблон, в языке предусмотрено использование метапредикатов. В языке существует множество встроенных метапредикатов для задания условий, которым должны удовлетворять константы, переменные, списки, фреймы, пути и нетерминалы. Помимо встроенных метапредикатов в языке существует возможность определять новые метапредикаты. Определение метапредиката является функцией, написанной на языке программирования Java. Правила использования созданных метапредикатов в спецификации не отличается от использования встроенных метапредикатов.

Блок действий

С нетерминальным символом грамматики, определяющей множество SOQL-запросов к коллекции, должен быть связан блок действий, результатом которых является строка символов; для нетерминала Query эта строка является запросом к коллекции данных, соответствующем запросу, определяемому данным символом. Действия в спецификации задаются при помощи языка программирования Java. В блоке действий возможно использование всех символов, присутствующих в грамматическом правиле, с которым связаны эти действия. Для каждого вида символов существуют правила их использования в Java..

Пример преобразования SOQL запроса в запрос к коллекции данных

В качестве примера возьмём коллекцию persons, содержащую информацию о группе людей (имя, фамилию, возраст и т.д.). Поиск по данной коллекции осуществляется при помощи процедуры lookup, которая вызывается из командной строки:

```
Query ::= SELECT $X FROM $X IN person WHERE $X.lastname = #C
{
        return new String("lookup –ln "+Const_C.getValue());
}
```

Пусть посредник посылает адаптеру на выполнение следующий запрос:

```
SELECT Y FROM Y IN person WHERE Y.lastname = "Smith"
```

Данный запрос удовлетворяет правилу, описанному выше. При сопоставлении запроса определению нетерминального символа Query, происходит также сопоставление термов запроса со специальными обозначениями этих термов в правиле вывода. Таким образом, данный SOQL-запрос преобразуется в строку вида:

```
lookup –ln Smith
```

5. Процесс создания адаптеров

При создании адаптера электронной коллекции необходимо задать спецификацию способностей языка запросов этой коллекции средствами языка спецификации, предлагаемого в настоящей работе. При этом происходит анализ возможности создания спецификации, независимой от схемы конкретного ресурса и применимой ко всем коллекциям в соответствующей модели данных. Создание такого рода спецификаций возможно путём отказа от использования имён и элементов схемы конкретной коллекции в описании. Необходимо заметить, что независимая от схемы спецификация предпочтительнее зависимой. Преимущества таких спецификаций заключаются в том, что адаптер, созданный на их основе, является устойчивым по отношению к изменению схемы коллекции, такой адаптер может использоваться для любой коллекции в конкретной системе управления данными.

После определения спецификации способностей языка запросов, создаётся спецификация компонента SELECTOR, выбирающего необходимые данные из результата, полученного от коллекции. Эта спецификация является описанием грамматики, которая определяет результирующее множество запроса к коллекции. Компонент SELECTOR генерируется на основе спецификации инструментом Minerva.

После этого, на основе составленных спецификаций, создаются необходимые компоненты адаптера и реализуется интерфейс взаимодействия адаптера с коллекцией. Процесс создания адаптера иллюстрируется рисунком 2.

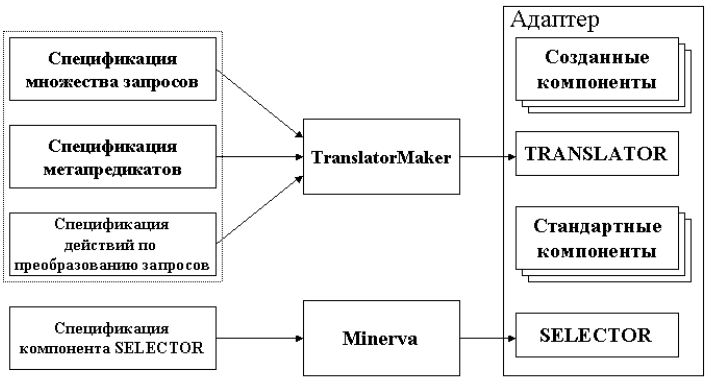

Рисунок 2 Процесс создания адаптеров

Набор средств для полуавтоматического конструирования компонентов адаптеров реализован на языке программирования Java в среде Windows NT. К этим средствам относятся следующие программы: утилита TranslatorMaker, создающая компонент TRANSLATOR адаптера, набор утилит Minerva, при помощи которого создаётся компонент SELECTOR, а также стандартные компоненты адаптера: MANAGER, CONSTRUCTOR и Filter Processor.

Для создания адаптера разработчику необходимо составить спецификации способностей языка запросов коллекции, для которой создаётся адаптер и спецификацию для Minerva, далее необходимо применить программы TranslatorMaker и Minerva к этим спецификациям для получения компонентов TRANSLATOR и SELECTOR. На следующем этапе разработчик адаптера должен реализовать компоненты Native Query Interface и Result Receiver

6. Заключение

В данной работе предложен подход к полуавтоматической генерации адаптеров. В частности предложена унифицированная архитектура адаптера, определены компоненты адаптера, которые являются унифицированными для всех адаптеров. Предложены методы генерации некоторых компонентов адаптеров на основе спецификаций. К таким спецификациям относятся: описание способностей языка запросов и действий по преобразованию запросов, а также описание правил выбора необходимой информации из результата запроса к коллекции данных.

Перечисленные решения экспериментально проверены: разработан TranslatorMaker, создающий компонент TRANSLATOR адаптера, проверены способы генерации компонента SELECTOR посредством Minerva, реализованы Native Query Interface и Result Receiver для конкретного вида коллекций (СУБД Tamino[9]).

Список литературы

[1] S. Grumbach, G. Mecca “In Search of the Lost Schema” - In Proceedings of Intern. Conference on Database Theory (ICDT'99), 1999

[2] Kalinichenko L.A., Briukhov D.O., Skvortsov N.A., Zakharov V.N., “Infrastructure of the subject mediating environment aiming at semantic interoperability of heterogeneous digital library collections”, Institute of Problems of Informatics RAS, Second All-Russian Conference Digital Libraries 2000

[3] Л.А. Калиниченко, “СИНТЕЗ – язык определения, проектирования и программирования интероперабельных сред неоднородных информационных ресурсов”, ИПИ РАН, Москва, 1993

[4] Котляров Ю.В., Подколодный Н.Л., “Подключение баз молекулярно-генетических данных к посреднику среды создания интегрированных электронных библиотек”, Вторая Всероссийская конференция “Электронные библиотеки”, сентябрь 26-28, 2000, Протвино

[5] Осипов М.А., Калиниченко Л.А., “Интеграция XML-коллекций данных в посреднике неоднородных коллекций электронных библиотек”, Вторая Всероссийская конференция “Электронные библиотеки”, сентябрь 26-28, 2000, Протвино

[6] Yannis Papakonstantinou, Ashish Gupta, Hector Garcia-Molina, Jeffrey Ullman “A Query Translation Scheme for Rapid Implementation of Wrappers”, Deductive and Object-Oriented Databases (DOOD) 95

[7] Yannis Papakonstantinou, Ashish Gupta, Laura Haas “Capabilities-Based Query Rewriting in Mediator Systems”, Parallel and Distributed Information Systems (PDIS) 96. Selected in the "Best of PDIS"

[8] Louiqa Raschid, Maria Esther Vidal, Jean-Robert Gruser “A Flexible Meta-Wrapper Interface for Autonomous Distributed Information Sources”, Technical Report AR 309, University of Maryland, Institute for Advanced Computer Studies- Dept. of Computer Science, March 1997

[9] Tamino Platform Homepage, http://www.softwareag.com/taminoplatform/

[10] Anthony Tomastic, Louiqa Raschid, Patrick Valduriez “Scaling Heterogeneous Databases and the Design of Disco”, Technical Report 2704, INRIA, November 1995