МОДЕЛИРОВАНИЕ И АВТОМАТИЗАЦИЯ ИНТЕГРАЦИИ ОБМЕНА ДАННЫХ

Д. В. Мельников (Санкт-Петербург)

Методы имитационного моделирования позволяют решать широкий круг задач, возникающих при проектировании АСУ, что дает возможность одновременного рассмотрения и оценки нескольких альтернативных вариантов проектных решений и в целом повышает достоверность и качество окончательно выбранного варианта. Имитационное моделирование является одним из методов, позволяющих оценить систему и ее реакцию на возмущения по ряду показателей. С помощью моделирования при создании АСУ могут решаться следующие задачи: определение путей совершенствования системы на основе моделирования различных вариантов технической, технологической, а также организационной перестройки и исследование последствий принятых решений. При имитационном моделировании остро встает задача интеграции данных, полученных при моделировании, для последующего анализа и обработки. В данной статье представлена система автоматизации интеграции обмена данных, полученных при моделировании сложного гетерогенного объекта. Интеграция данных осуществлялась по модели XML-СУБД.

С появлением стандарта XML обмен гетерогенными данными предприятиями в значительной мере упростился, но существует еще ряд проблем, присущих обмену данными между различными приложениями. Одной из таких проблем является сложность реализации интеграции данных ввиду большого количества систем управления базами данных (СУБД) и различных типов представления данных в них. Известно, что каждая из СУБД часто имеет свое собственное расширение стандарта SQL. В настоящее время чаще всего используется следующий подход для реализации интеграции данных: для каждой из информационных систем предприятия создается свой модуль для интеграции сторонних данных в СУБД системы. Данный подход довольно трудоемок и часто требует постоянного рефакторинга программного кода системы, особенно если данные обмена часто меняются и требуют поддержания версионности. Примером данных, требующих версионности, являются структуры документов (документы ВЭД, документы партнерских информационных систем, государственные документы). Поддержание версионности данных для интеграции – довольно трудоемкая операция, требующая большого количества времени высококвалифицированных специалистов. Чтобы упростить данную процедуру предлагается оригинальная идея системы оптимизации интеграции гетерогенных данных информационных систем.

Сама идея заключается в использовании (создании) библиотеки для интеграции данных с помощью скриптов, содержащих данные для интеграции, а также создании редактора вышеупомянутых скриптов. Схема модулей системы представлена на рис. 1.

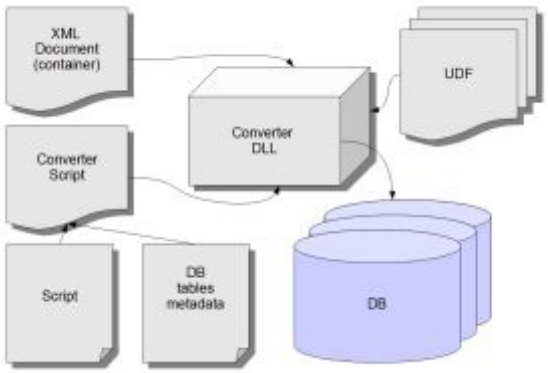

Рис. 1. Схема работы конвертера данных

Основные преимущества данного подхода прозрачны и заключаются в том, что скрипты с помощью графического редактора может править специалист в предметной области без участия высококвалифицированных кадров для модифицирования информационных систем. Говоря простым языком, данный подход позволяет избавиться от участия разработчиков при изменении структуры данных обмена в процессе интеграции в БД. Это позволяет уменьшить затраты на разработку информационных систем.

Также достоинством можно считать то, что, однажды спроектировав конвертер для какого-либо набора СУБД, будем иметь интеграцию с этим набором СУБД и для других информационных систем. Таким образом, можно в конвертер изначально включить поддержку конвертации основных используемых СУБД предприятия (Oracle, Microsoft SQL Server, FireBird), и, тем самым, забыть о гетерогенности информационных систем и интеграции данных в них.

Конвертация и интеграция данных с использованием различных СУБД реализуется методом накопления поддержки. Технологически это требует использования специального метаслоя данных, который содержит данные в промежуточном состоянии, для которых правила преобразования уже применены, но окончательной интеграции еще не произведено. На данном этапе подход с использованием метаслоя реализован следующим образом: создается DOM-структура (Document Object Model), которая, по сути, повторяет структуру таблиц БД, а дочерними элементами табличных элементов являются данные, повторяющие поля таблиц и данных после применения правил преобразования (рис. 2).

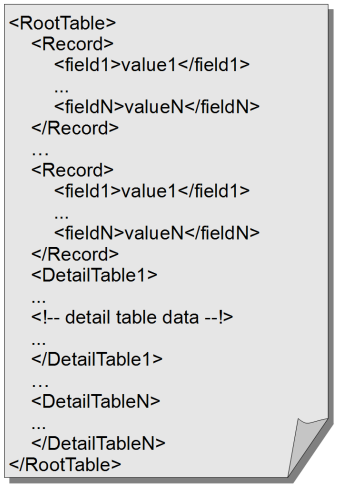

Рис. 2. Структура метаслоя данных

После создания такой DOM-структуры дальнейшая интеграция не составит больших трудностей. Используя один и тот же алгоритм занесения данных в БД, имеем мультиплатформенную с точки зрения баз данных систему, для занесения данных в которую требуется лишь замена драйверов СУБД.

Еще одним преимуществом данного подхода является более гибкий подход к преобразованию данных, участвующих в интеграции. На практике часто требуется значение XML-тега, скажем, разбить на несколько полей в БД или (не)преобразовывать данные при каком-либо условии. Например, заполнять финансовые реквизиты контрагента, если в данных содержится признак юридического лица. От представленных сложностей интеграции с помощью скриптов позволяют избавиться так называемые правила преобразования, которые поддерживает библиотека (далее конвертер) и редак-

тор скриптов. Эмпирическим путем были получены требования к данным правилам. Количество правил на данный момент реализации не превышает 10.

Рассмотрим механизм интеграции более подробно. Представим, что у нас есть XML-документ, который нужно преобразовать, но не тривиальным преобразованием, а с использованием относительно сложных правил.

Скрипт преобразования состоит из двух частей (двух файлов):

- описание структуры данных, куда происходит интеграция (описание таблиц, их подчиненности, а также полей таблиц, их типов и признаков первичных ключей);
- непосредственно скрипт с указанием правил преобразования.

Подробно останавливаться на описании структуры данных не будем, так как это достаточно примитивная структура описания подчиненности таблиц, полей и ключей. Но скрипт преобразования опишем подробнее.

Скрипт представляет собой XML-документ, который повторяет структуру исходного XML-документа с данными, но вместо данных в нем в качестве параметров тегов описаны правила преобразования. Также отличием структуры от файла обмена является отсутствие повторения тегов. Каждый тег исходного файла в скрипте встречается один раз. Если встречаются структуры наподобие choice или list, то каждый тег из списка помещается в скрипт. Отсутствие повторений тегов спроектировано с целью избежания дублирования данных и упрощения модели интеграции.

Алгоритм работы конвертера следующий: на первом шаге осуществляется перебор всех тегов XML-документа обмена, затем ищутся соответствующие правила преобразования, анализируются и, в зависимости от результата, помещаются в промежуточную структуру, после чего записываются в БД. Описанный выше алгоритм можно записать в виде следующего псевдо-кода:

```
для каждого тег из файл_обмена
      тег_скрипта = Найти_в_скрипте(тег)
      если тег_скрипта существует и не пустой
            если Требуется_новая_запись_в_метаслое()
                  Добавить_запись_в_метаслой()
            если правило_преобразования не пустой
                  данные = Выполнить(правило_преобразования)
 результат = данные + Параметры(тег_скрипта)
            Записать_в_метаслой(результат)
      тег = Следующий_тег()
```

Рассмотрим математическую модель преобразования тегов при помощи правил.

Пусть существует множество тегов $T = \{t_1, t_2, ...t_n \mid A(t)\}$, каждый из которых обладает свойством $A(t)$, которое идентифицирует данные теги в соответствии с некоторым значением. Пусть существует множество тегов скрипта $S = \{s_1, s_2, ...s_n \mid B(s)\}$, каждый из которых обладает свойством $B(s)$ соответственно. Если для $t_i \in T$

$$\exists \quad A(t_i) = B(\forall s \in S) \exists \ ,$$

тогда

$$D_{ij} = f_k(t_i, s_j),$$

где D_{ij} будет результатом правила преобразования f_k для данных тегов. Функция f зависит от конкретного правила преобразования.

Опишем правила, реализованные на данном этапе реализации конвертера данных. В дальнейших выкладках под функцией $f(t_i)$ будем понимать значение тега t_i в файле обмена.

- Add

Функция добавляет значение к уже заполненному полю. Требуется, когда несколько тегов преобразуются в одно поле.

Формальное представление:

$$f_{Add}(\{t_i : i = m..k\}, s_j) = f(t_m) + f(t_{m+1}) + \ldots + f(t_k)$$

- AutoGen

Функция генерирует преобразование тегов, которых в XML-документе нет, но которые должны присутствовать в результате интеграции (например, нужно к каждой записи добавлять порядковый номер, которого в документе обмена нет).

Формальное представление:

$$T = \{t_1, t_2, \ldots t_n\}, \quad f_{AutoGen}(\{t_{n+1}, \ldots, t_{n+m}\}) : \; T = T \cup \{t_{n+1}, \ldots, t_{n+m}\}$$

- Concat

В теге, для которого было вызвано данное правило преобразования, производится поиск дочерних тегов $\{t_i, t_{i+1}, \ldots t_k\}$, и все они конкатенируются в одно значение с указанным разделителем. Формальное представление:

$$f_{Concat}(\{t_i : i = m..k\}, s_j) = f(t_m) + f(t_{m+1}) + \ldots + f(t_k)$$

- Date

Функция преобразует дату из формата XML (yyyy-mm-dd) в формат dd.mm.yyyy

- DiffValues

Данное правило служит для преобразования повторяющихся тегов в различные поля DBF (часто требуется первый тег из нескольких преобразовать в одну таблицу, а остальные в другую).

Формальное представление:

$$T = \{t_{i1}, t_{i2}, \ldots t_{in}\}, \quad f_{DiffValues}(t_i) = \begin{cases} f(t_{i1}, s_j) = D_{i1j}, \; j = 1 \\ \ldots \\ f(t_{in}, s_j) = D_{inj}, \; j = n \end{cases},$$

- Gen

Функция Gen при выполнении ищет указанный параметр в ini-файле настроек, и, если находит такой, то возвращает значение данного ключа. Данное правило удобно для получения значений констант (юридический адрес, номер версии скриптов, ИНН).

Формальное представление:

$f(t_i, s_j) = \alpha$, где α – константа

- If

Правило реализует оператор условия. В результате преобразования могут быть получены значения тегов или применены другие правила в зависимости от логического условия.

- Numrec

Возвращает порядковый номер текущей записи в указанной таблице метаслоя данных. Правило может возвращать порядковый номер с учетом совпадения заданных условий.

- RecordCount

Возвращает количество записей в указанной таблице после всех преобразований.

- SetParam

Устанавливает в runtime-режиме работы конвертера указанные для тегов значения параметров преобразования. Работает аналогично правилу DiffValues.

- SpecValue

Возвращает специальные значения, определенные предметной областью и реализованные в программном коде конвертера.

- Формальное представление:

$f(t_i, s_j) = \beta$, где β – константа

- TagNumber

Находит указанный тег на том же уровне, что и тег, для которого осуществляется преобразование, и возвращает его порядковый номер в повторяющейся последовательности тегов, начиная с 1. При необходимости вводится параметр инкремента результата.

- Value

Возвращает указанное в параметрах преобразования значение. Можно комбинировать произвольный текст и значение тегов.

В результате комбинирования и использования рассмотренных правил автоматизация интеграции данных упрощается. Производительность данной библиотеки конвертации достаточно высока. Интеграция в СУБД FireBird XML-файла размером > 2 Мб занимает в среднем 5–6 секунд. В это время входит построение DOM-модели исходного XML, всех скриптов (их для подобного документа 5–6), построение метаслоя данных с учетом правил преобразования и запись в БД.

Для упрощения и ускорения создания скриптов был создан графический редактор с интеллектуальным анализом введенной информации (рис. 3).

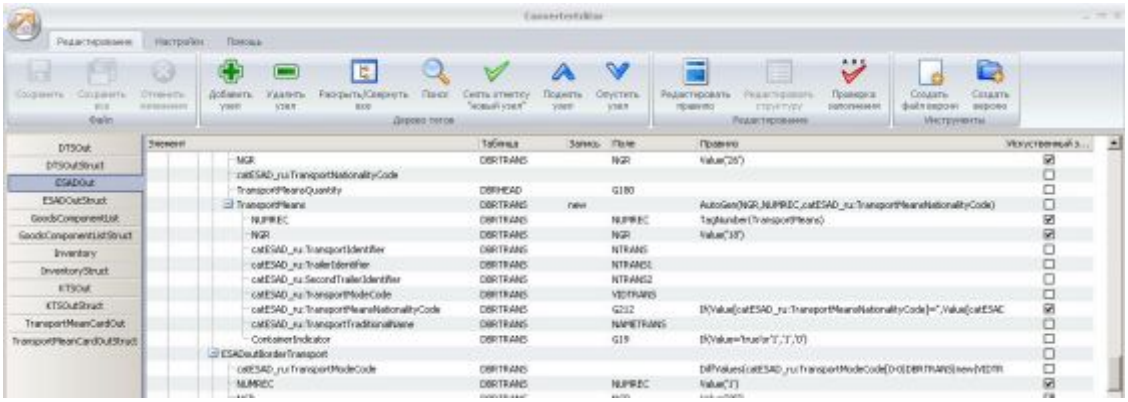

Рис. 3. Редактор скриптов

В результате проделанной работы при помощи методов имитационного моделирования был спроектирован и разработан конвертер для интеграции данных по модели XML-СУБД.

Выводы. Представленный подход интеграции данных при моделировании сложных гетерогенных систем особенно эффективен при агентном моделировании, когда анализируемая система децентрализована. Рассмотренный метод интеграции данных для анализа позволяет избежать проблемы репликации данных, дает возможность конвертировать данные в различные СУБД, что дает возможность использовать любые пакеты имитационного моделирования (AnyLogic, eM-Plant, GPSS). Универсальность и независимость от разработчика позволяет накапливать данные моделирования без привлечения высококвалифицированных кадров.