

РАЗРАБОТКА СИСТЕМЫ ИНТЕГРАЦИИ ДАННЫХ

В.Ю. Сакович, П.А. Дробушевич

Белорусский государственный университета, кафедра МО ЭВМ
пр. Независимости, 4, Минск, Беларусь
телефон: + (375) 17 2095334; факс(ы): + (375) 17 2265548; e-mail: sakovich@bsu.by
web: www.bsu.by

В работе изучены методы построения систем интеграции данных. Рассмотрена концепция ETL (Extract, Transform, and Load). Разработана платформа для построения систем интеграции данных.

ИНТЕГРАЦИЯ ДАННЫХ, МОДУЛЬ, ПРАВИЛО, РЕФЛЕКСИЯ

Сейчас наступила эпоха интегрированных корпоративных информационных систем, которые отличаются тем, что все приложения, функционирующие в рамках соответствующей системы, так или иначе связаны между собой, и обычно требуется масштабирование и модификация системы.

Интеграция данных необходима в следующих случаях:

1. Получение данных для хранилищ данных и систем бизнес-аналитики.
2. Создание интегрированных хранилищ основных, или мастер-данных.
3. Миграция/преобразование.
4. Синхронизация данных между приложениями, поддерживающими оперативную деятельность [1].

Существует три основных метода интеграции данных: консолидация, федерализация и распространение.

Извлечение, преобразование и загрузка, известные под аббревиатурой ETL, – это основные этапы переноса информации из одного приложения в другое. Для успешного переноса данных из одной системы в другую крайне важно четко представлять процессы ETL, а также структуру исходного приложения и приложения назначения.

Приложения ETL извлекают информацию из исходного источника данных, преобразуют ее в формат, поддерживаемый источником данных назначения, возможно, применяют дополнительные преобразования, а затем загружают преобразованную информацию. Для того, чтобы инициировать процесс ETL, применяются программы извлечения данных для чтения записей в исходной источнике и для подготовки информации, хранящейся в этих записях, к процессу

преобразования. Чтобы извлечь данные из исходного источника, можно выбрать один из трех вариантов: создать собственные программы, обратиться к готовому специализированному инструментарию ETL или использовать сочетание и того, и другого [2].

Нами разработана система, позволяющая эффективно решать эти проблемы. Она состоит из следующих семи модулей:

1. Job – модуль системы, который отвечает за обработку однотипных данных, состоит из подмодулей:

а) Metadata – описывает возможные зависимости job от других jobs, правила запуска, возможности параллельного запуска. Для описания используется язык xml.

б) Rule – правила чтения, трансформации и обработки данных, и загрузки в источник. Для описания используется специально разработанный язык.

2. Compiler – компилятор Rule из специального языка в java байт-код.

3. Engine – модуль, позволяющий запускать Rule, использует модуль Compiler для предварительной компиляции Rule.

4. DependenceRule – специальные Rule, которые используются для определения возможности запуска определённой Job.

5. Checker – модуль, который используется для определения, какую Job можно запустить, и использует DependenceRules.

6. Scheduler – использует Checker для определения, какую Job можно запустить, также используется Metadata Job, чтобы определить корректность запуска Job, для запуска используется Engine.

7. Server – модуль, который используется для ручного запуска Job, занимаясь отслеживанием их работы.

Модуль Engine, используя средства рефлексии платформы java, загружает построенный компилятором класс и вызывает у него метод execute, который стартует процесс интеграции.

Компонента Scheduler, основываясь на метадате репозитория правил, запускает компоненту Checker с правилами проверки условий, и для тех правил, которые оказались успешными, запускаются исполняемые правила. Статистику выполнения правил можно просмотреть с помощью компоненты Server.

Взаимосвязи между модулями приведены на рис. 1.

Так как система имеет компонентно-ориентированную архитектуру, то ключевой частью Rule является язык описания компонент и связей между ними. Все компоненты делятся на группы:

- а) Readers – компоненты для получения данных из хранилища.
- б) Transformers – компоненты для выполнения преобразований над данными.
- в) Filters – фильтры для отсева тех записей, которые не требуются в выходном хранилище.
- г) Joiners – компоненты для объединения данных.
- д) Aggregate – компоненты агрегации данных.
- е) Writers – компонента для записи, обновления и удаления документов в целевом хранилище данных.

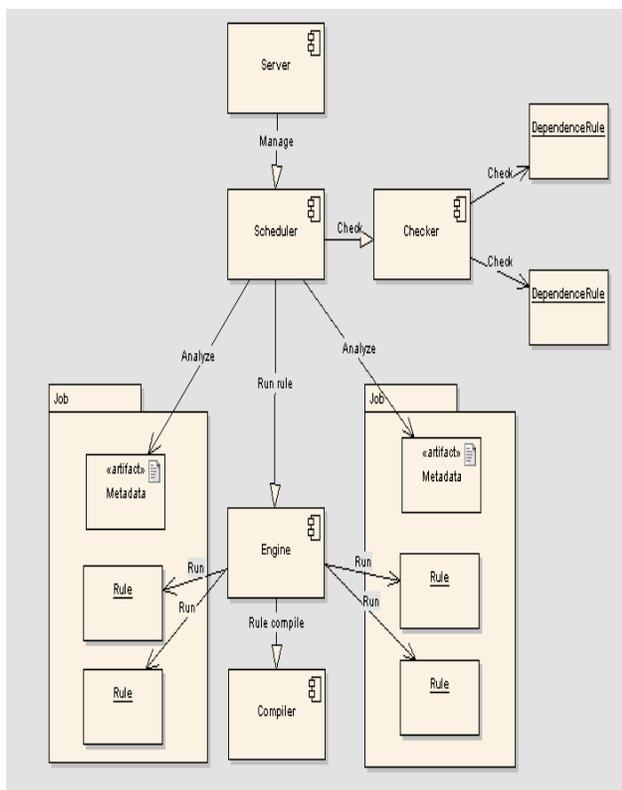


Рис. 1. Общая архитектура модулей системы ETL

Описание компоненты имеет следующую структуру:

```
reader xmlReader {
    property RunMode = "lookup";
    property FieldElement = "./field";
    property FieldNameElement =
"./name/text()";
```

```
property FieldValueElement =
"./value/text()";
type = "XmlReader";
ds = "localFileReadDataSource";
map = "//docs";
outputscheme = "None";
call = "jdbcWriter";
call = "xmlWriter";
}
```

Также возможны связи компонент между собой, задающие очерёдность вызова компонент.

```
reader xmlReader {
    call = jdbcWrite;
}
writer jdbcWrite {
}
```

Этот язык описания позволяет создавать наборы функций с java-кодом, которые можно использовать в компонентах для трансформации данных.

Компиляция правил состоит из двух частей:

1. Генерация синтаксического дерева.
2. Генерация java байт-кода правила по синтаксическому дереву.

На текущий момент для системы разработаны компоненты для работы с реляционными базами данных, файлами xml, веб-сервисами и поисковым модулем Apache Lucene. А также базовые компоненты для трансформации, использующие функции, написанные на языке java, компоненты фильтр и объединение.

Приведем пример задачи интеграции данных, которую позволяет эффективно решить разработанная система. Компания имеет приложение, с помощью которого сотрудники описывают заказы от клиентских компаний. Приложение сохраняет все данные во внутреннюю базу данных. Существует вторая компания, которая предоставляет веб-сервис со своими заказами. Для клиентов необходим поисковый сервис для просмотра статистики заказов обеих компаний.

Для решения этой задачи разработаны три правила. Первые два читают данные из базы данных и веб-сервиса и загружают их в промежуточную базу данных. Третье читает эти данные, объединяет и создаёт файлы поискового индекса для поисковой компоненты.

ЛИТЕРАТУРА

[1] Colin White: Data Integration: Using ETL, EAI, and EII Tools to Create an Integrated Enterprise [Electronic resource]. - <http://www.tdwi.org/research/display.aspx?ID=7908>.

[2] David Waddington: Federated Enterprise Data Warehousing – A Management Overview [Electronic resource]. - <http://hosteddocs.ittoolbox.com/DW041505.pdf>.