Методы и системы моделирования и обработки данных, ориентированные на эволюцию схем

Д.В. Луцив
dluciv@math.spbu.ru

СПбГУ

18 февраля 2006

Содержание

Реляционные

Специфика

1. Схема первична и «непререкаема»
2. Если данные не соответствуют схеме, см (1)
3. Стандартизованы подмножества реальных схем

Результат

Только специфические системы трансформации данных при преобразовании схемы в полной мере могут использовать схему данных.
Пример: «родные» ETL для MSSQL и Oracle

Реляционные

Специфика

1. Схема первична и «непререкаема»
2. Если данные не соответствуют схеме, см (1)
3. Стандартизованы подмножества реальных схем

Результат

Только специфические системы трансформации данных при преобразовании схемы в полной мере могут использовать схему данных.
Пример: «родные» ETL для MSSQL и Oracle

TIGUKAT (University of Waterloo)

«tee-goo-kat» – «объект» на языке канадских эскимосов

Факты

- Активно разрабатывалась в 1991 – 1999 годах
- Предоставляет развитую формальную модель данных и поведения объектов
- Модель основана на традиционных для ОО принципах и построена на теории множеств
- Исследования Группы теории БД СПбГУ показали применимость подхода к другим методологиям
- Принципиальное преимущество модели перед другими заключается в возможности указания «существенных» данных и предков класса, которые система стремится сохранять при трансформации

Другие

- TIGUKAT - 9 инвариантов, 8 правил
- Orion - 5 инвариантов, 12 правил
- Современные СУБД для серверов приложений
  - Часто представляют собой Object – Relational persistence layer
  - 0 инвариантов, 0 правил

Другие

Типичны

- Описательная схема
- Явная трансформация исключительно данных
  - Средствами API
  - Средствами XQuery/XUpdate

Примеры

- Sedna
- eXist

Ориентированные на документы

XSLT

- – Неудобный синтаксис
- \+ Синтаксис соответствует XML
- \+ Декларативна
- – Ничего не знает о схеме

XPathScript

- ○ Часть Apache AxKit
- \+ Позволяет вставлять большие фрагменты на Perl
- – В основном, императивна
- – Ничего не знает о схеме

STX

- ○ Похож на XSLT, но построен на SAX

Ориентированные на документы

XSLT

- – Неудобный синтаксис
- \+ Синтаксис соответствует XML
- \+ Декларативна
- – Ничего не знает о схеме

XPathScript

- ○ Часть Apache AxKit
- \+ Позволяет вставлять большие фрагменты на Perl
- – В основном, императивна
- – Ничего не знает о схеме

STX

- ○ Похож на XSLT, но построен на SAX

Ориентированные на документы

XSLT

- – Неудобный синтаксис
- \+ Синтаксис соответствует XML
- \+ Декларативна
- – Ничего не знает о схеме

XPathScript

- ◦ Часть Apache AxKit
- \+ Позволяет вставлять большие фрагменты на Perl
- – В основном, императивна
- – Ничего не знает о схеме

STX

- ◦ Похож на XSLT, но построен на SAX

Ориентированные на схемы

EvoXSD

- ◦ Ориентирована на трансформацию исходя из изменений схемы
- \+ Манипулирует схемой
- \+\+ Позволяет описывать «существенные» и «несущественные» данные
- – Не полностью поддерживает схему XML
- ◦ Система реализована к.ф.-м.н. Группы теории БД СПбГУ и отражает достижения, описанные в его диссертации.

TransMorpher

- ◦ Более ориентирован на документ
- \+ В описании трансформации используются сущности схемы

Ориентированные на схемы

EvoXSD

- ◦ Ориентирована на трансформацию исходя из изменений схемы
- \+ Манипулирует схемой
- \+\+ Позволяет описывать «существенные» и «несущественные» данные
- – Не полностью поддерживает схему XML
- ◦ Система реализована к.ф.-м.н. Группы теории БД СПбГУ и отражает достижения, описанные в его диссертации.

TransMorpher

- ◦ Более ориентирован на документ
- \+ В описании трансформации используются сущности схемы

Extract – Transform – Load

ETL

1. Загружает данные из [унаследованной] информационной системы
2. Применяет к ним бизнес-правила трансформации данных
   - Правила обычно довольно простые
   - Правила связываются в сложные структуры
3. Сохраняет данные в целевую информационную систему

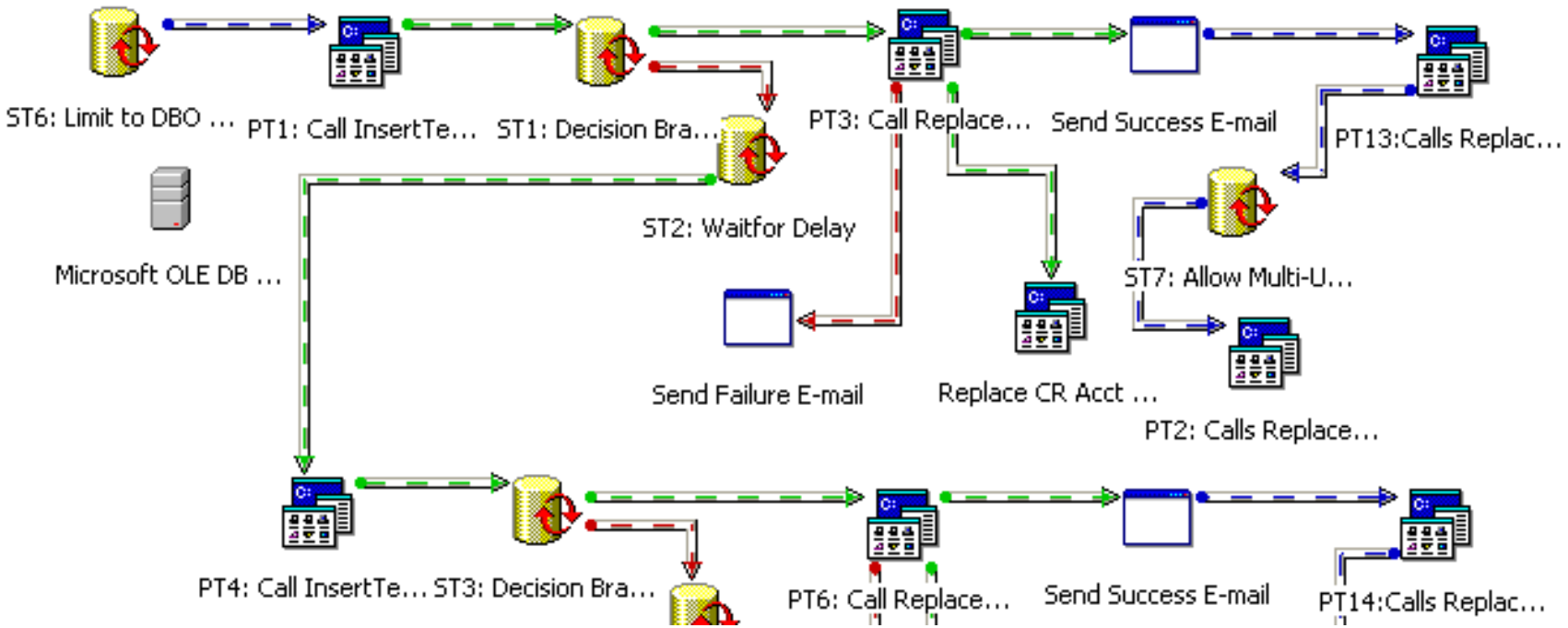
Примеры

- SQL Server Integration Services (Included in MSSQL 2005)
- Data Transformation Services (Included in MSSQL 2000)
- Oracle Warehouse Builder
- Informatica PowerCenter & PowerExchange

Типичные трансформации

- Проекция (выбор колонок)
- «Перешифровка»
  М и Ж $\longrightarrow$ 1 и 2, соответственно
- Кодирование свободных форм
  Ж, Жен, ... $\longrightarrow$ 2
- Вычисление вычислимого
  sale_amount = qty * unit_price
- Слияние данных из разных источников
  соединение, объединение, ...
- Агрегирование, факторизация, более высокоуровневые операции
- Генерация сурогатных ключей
- Транспонирование (см. первый пункт)

Пример: MS Data Transformation Services



Is ETL Becoming Obsolete?

- ETL требует детального описания бизнес-логики и, следовательно, детального понимания семантики данных до и после — Правда. А это плохо?
- ETL последовательно обрабатывает требуемые исходные данные снаружи от БД — Иногда правда
- ETL очень дорого стоят — Правда

Подход E-LT (Extract – Load, Transform)

- Трансформация внутри целевой БД
- Использование новых мощных средств реализации серверной логики (расширения SQL, хранимая SQL и не SQL логика)
- Система компилирует бизнес-правила трансформации и фильтрации в код целевой операционной среды

Is ETL Becoming Obsolete?

- ETL требует детального описания бизнес-логики и, следовательно, детального понимания семантики данных до и после Правда. А это плохо?
- ETL последовательно обрабатывает требуемые исходные данные снаружи от БД Иногда правда
- ETL очень дорого стоят Правда

Подход E-LT (Extract – Load, Transform)

- Трансформация внутри целевой БД
- Использование новых мощных средств реализации серверной логики (расширения SQL, хранимая SQL и не SQL логика)
- Система компилирует бизнес-правила трансформации и фильтрации в код целевой операционной среды

Группа управления данными и информационных систем ИСП РАН

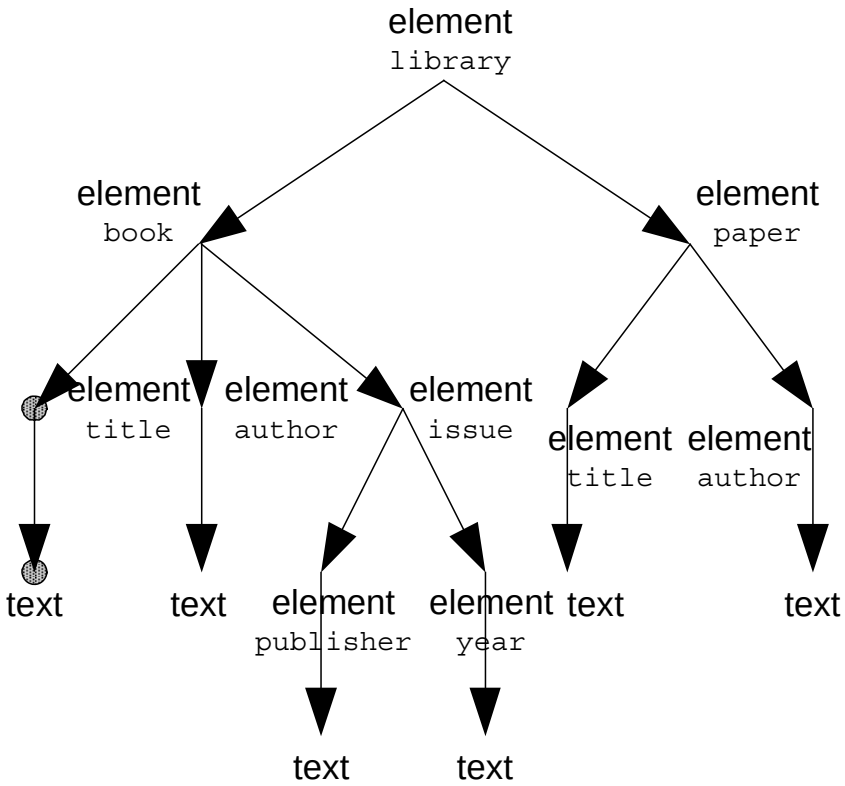
- Организация XML-данных во внешней памяти
- Индексирование XML-данных
- Методы эффективного выполнения операций физического уровня
- Методы управления транзакциями
- Методы обеспечения логической целостности данных
- Логическая оптимизация запросов

Группа теории БД СПбГУ

- Concurrent Video (Совместная обработка видео)
- Data Evolution (Эволюция данных)
- Seamless Design (Неявный дизайн)
- Мобильные системы
- Статические изображения
- Темпоральные и пространственные базы данных
- XML базы данных

Описательная схема слабоструктурированных документов СУБД «Sedna»

+ Генерируется (и поддерживается) динамически на основе хранимых данных
– Не может быть использована при оптимизации запросов так же эффективно, как объявляющая



Трансформация в соответствии с MDA (М. Кузнецов)

Входные данные

- одна или более моделей
- метамодель каждой из моделей
- описание трансформации на специальном языке
  - зависит от содержания и количества метамоделей
  - не зависит от моделей

Выходные данные

- модифицированные исходные модели
- ноль или более новых моделей, каждая соответствует какой-либо входной метамодели
- информация об отображениях и связях между моделями, возникших в процессе трансформации

Базовая модель данных и формализация эволюции схем XML

- Построена на основе адаптированных инвариантов и операций ОСУБД TIGUKAT
- Поддерживает определение иерархии тэгов и содержащихся в схеме документов
- Поддерживает определение наборов «существенных» атрибутов и связей в иерархии
- Предоставляет 8 операций эволюции
- Была использована при реализации системы EvoXSD

Базовая модель данных и формализация эволюции схем XML

- Построена на основе адаптированных инвариантов и операций ОСУБД TIGUKAT
- Поддерживает определение иерархии тэгов и содержащихся в схеме документов
- Поддерживает определение наборов «существенных» атрибутов и связей в иерархии
- Предоставляет 8 операций эволюции
- Была использована при реализации системы EvoXSD

Расширения модели данных и различные подходы к программной реализации

Расширения

- Регулярные выражения DTD (варианты XSD)
- Ключи и ссылки

Подходы к реализации

- Императивный
- Декларативный

Расширения модели данных и различные подходы к программной реализации

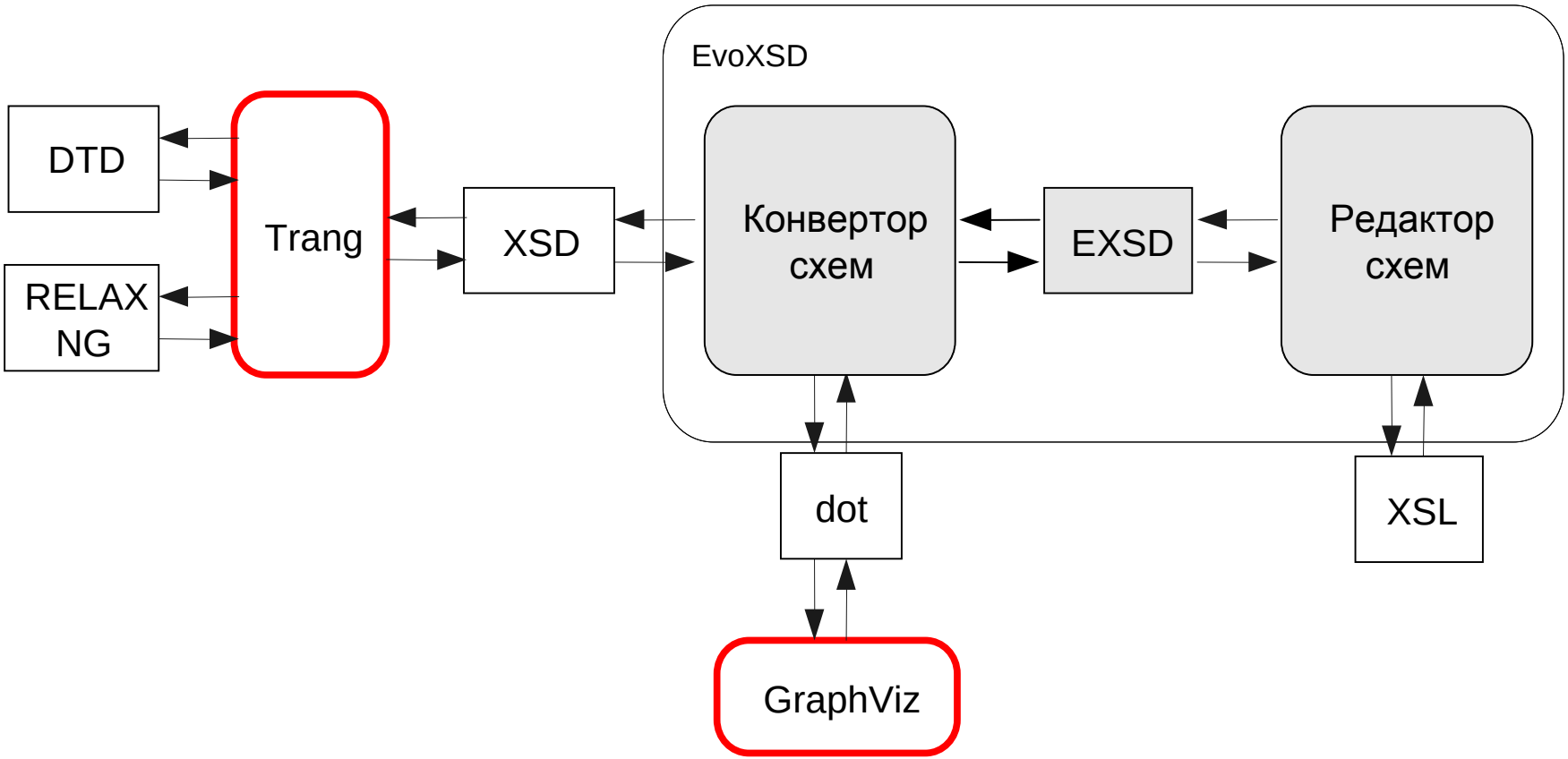
Расширения

- Регулярные выражения DTD (варианты XSD)
- Ключи и ссылки

Подходы к реализации

- Императивный
- Декларативный

Архитектура и контекст системы управления эволюцией схемы «EvoXSD»



Microsoft Research Cω (Си омега)

```
<element name="Address">
 <complexType>
  <sequence>
   <choice>
    <element name="Street"
     type="string"/>
    <element name="POBox"
     type="int"/>
   </choice>
   <element name="City"
    type="string"/>
  </sequence>
 </complexType>
</element>
```

```
class Address {
 struct {
  choice{
   string Street;
   int POBox;
   };
  string City;
 };
}

Address a = <Address>
 <Street>13 Elm St</Street>
 <City>Hollywood</City>
</Address>;
```

```
for $b in $bs/book
 return <result>{$b/title}{$b/author}<result>
```

```
foreach (b in bs.book)
 yield return <result>{b.title}{b.author}</result>;
```

Scala

Отличия

- \+ Язык объектно-функциональный
- \+ Иные вариантные структуры данных и согласование образцов (pattern matching)
- +− Язык не семейства C (Cω – почти семейства C)
- \+ Кросс-генерация в Java и .NET

CDuce

- Язык семейства ML, построен на XDuce, тот, в свою очередь, на OCaml
- Система типов адаптирована для поддержки типов XML
- Более мощное, чем в OCaml, согласование образцов (pattern matching) с частично статической типизацией данных
- Оптимизация, доступная при статической типизации дает типичную (как и для OCaml) скорость сравнимую с C и C++

Haskell

Замечание

Язык не ориентирован на работу с XML данными и схемами

$$function : State \times Input \rightarrow State' \times Output$$

Для большинства императивных языков $State = State'$ и вообще равно практически чему угодно, так как нормально не описано

Для некоторых функциональных состояние действительно имитируется, а не присутствует, что дает:

- Мощные глобальные оптимизации
- Ленивость
- Возможность описывать операции, действительно выполняющие преобразования

Prolog

Замечание

Язык не ориентирован на работу с XML данными и схемами

```
% Immediate Predecessors
inP2(T, S) :-
 inPe(T, S),
 inPe(T, X),
 X=\=S,
 not(inPL(X,S)).

inP2(T, S) :-
 inPe(T, S),
 tag(X),
 X=\=S, %only one in Pe
 not(inPe(T,X)).

inP1(T,S) :-
 inP2(T,S),!.
```

```
inP(T,S) :-
 unbound(S),
 tag(S),
 inP1(T,S).

inP(T,S) :-
 unbound(T),
 tag(T),
 inP1(T,S).

inP(T,S) :-
 bound(T), bound(S),
 inP2(T,S),!.
```

Спасибо