DOI: 10.25728/avtprom.2021.05.14

Метод матричного преобразования при конвертации баз данных

С.В. Маличенко (ФГБОУ ВО «МИРЭА - Российский технологический университет»)

Разработан новый автоматизированный конвертер, преобразующий данные и процедуры между электронными базами данных. Впервые предложен подход, основанный на использовании алгебраического метода для конвертации данных. Это позволяет сократить трудозатраты и сохранить данные в актуальном формате, так как представленный алгоритм не зависит от предметной области и базы знаний. Разработанный алгоритм способен производить конвертацию между базами, различающимися как моделью данных, так и структурой.

Ключевые слова: база данных, онтология, матрица, конвертер.

Введение

С ростом знаний об окружающем мире продолжает расти объем генерируемых данных. Это требует своевременного документирования новой информации и актуализации уже существующей. При этом возникают сложности, связанные с преобразованием бумажных данных в электронный формат и миграцией электронных данных между разными цифровыми форматами. Конвертация данных — это эффективный способ управления информацией, при котором пользователь может систематизировать свои данные и преобразовать их в формат, который можно использовать на разных цифровых платформах. Конвертация данных дает ряд преимуществ: позволяет объединить несколько устаревших компьютерных систем и преобразовать их в современные аналоги. Такие преобразования позволяют обеспечить актуальность существующих данных.

После преобразования данные могут иметь меньшую латентность[1], что позволяет им оставаться актуальными еще долгое время.

Конвертация также позволяет при работе с современными системами управления бизнес-процессами использовать старые базы данных.

Постановка задачи

Факты окружающего мира могут быть структурированы и описаны человеком. Но ценность этой информации может быть утеряна без возможности хранения накопленных знаний. Для упорядоченного хранения ее создаются специализированные кластеры, в которых содержится информативная часть этих знаний в специальном формате. Для представления централизованного доступа к этим данным создаются базы данных.

База данных (ГОСТ 20886-85) – это совокупность массивов и файлов данных, организованная по определенным правилам, предусматривающим стандартные принципы описания, хранения и обработки данных независимо от их вида.

База данных имеет строгую структуру и организована по определенным правилам. То есть описания ее элементов и отношений формализованы и поддаются группировке по каким-либо критериям. По сути, такой подход реализует онтологическую функцию - любая концептуальная модель может быть представлена в виде формализованных структур с релевантными объектами. А связи объектов, содержащихся в ней, также формализованы. Это дает инструмент для представления знаний о реальном мире или его части.

Современные системы управления БД имеют синтаксис для описания предметной области знаний и специфическое внутреннее представление. При этом формирование структуры производится на специальном языке, который при выполнении на ЭВМ меняет внутреннее представление модели.

Основными понятиями баз данных являются домен, атрибут, кортеж, ключ и отношение [1, 2]. Понятие типа данных полностью схоже с понятием типа данных в языках программирования. Обычно современные базы данных поддерживают множество типов данных как символьных, так и числовых, а также специальных – относящихся к узкой специфике предметной области. Домен определяется заданием некоторого базового типа данных, к которому относятся все элементы базы. Кортеж представляет собой отношение пар имя - значение и содержит в себе одно вхождение каждого имени атрибута домена.

Цель настоящего исследования: разработка механизма конвертации данных в условиях интенсивного расширения доменной модели.

Задачи, выполнением которых достигается эта цель:

- определить особенности конструирования ядра механизма конвертации.

[1] Латентность - задержка или ожидание, которая увеличивает реальное время отклика по сравнению с ожидаемым.

- обозначить особенности конвертации хранимых процедур онтологий в моделях предметной области.

В настоящей работе рассматривается применение алгебраического подхода к конвертации данных.

Алгоритм конвертации

В линейной алгебре векторное пространство, как правило, характеризуется базисом и размерностью (n). Базисом векторного пространства из n векторов называется последовательность из n векторов, причем таких, что любой вектор пространства может быть представлен единственным образом в виде линейной комбинации базисных векторов [3-5].

То есть базисом может быть любая комбинация из n линейно-независимых векторов в данном пространстве. Проведя параллель с базами данных, находим несколько схожих элементов. В частности, атрибуты БД представимы в виде вектора базиса. Кортеж по аналогии сопоставим с вектором в пространстве размерностью (n). Точка в базисе в данном сравнении – это запись в ячейке таблицы.

Предлагаемый алгоритм конвертации сводится к нахождению матрицы перехода от базиса $e_1, e_2, e_3,.....,e_n$ к базису $e_1, e_2, e_3,.....,e_n$, где e_n, e_n' – атрибуты БД. В строгих математических трактовках данный метод можно описать следующим образом.

Представим, что первичная база имеет атрибуты i_1, i_2 и данные в базе – кортежи $[a_1, a_2]$ и $[b_1, b_2]$. Соответственно, в алгебраическом виде эти записи имеют вид:

$$\bar{a} = a_1(\bar{i}_1) + a_2(\bar{j}_1)$$
$$\bar{b} = b_1(\bar{i}_1) + b_2(\bar{j}_1)$$

или $\bar{a}\,(a_1, a_2)$ и $\bar{b}\,(b_1, b_2)$.

Систематизируем алгоритм решения данной задачи: даны два произвольных базиса пространства размерностью $n = 2$: $(\bar{i}_1, \bar{j}_1)$ и $(\bar{i}_2, \bar{j}_2)$, при этом векторы второго базиса можно выразить через векторы первого следующим образом:

$$\bar{j}_2 = a_{12}(\bar{i}_1) + a_{22}(\bar{j}_1)$$

В данном контексте a_{11} - первая координата первого вектора, a_{11}– первая координата второго вектора и. т. д. В контексте БД это означает: a_{11} – отношение атрибута первой записи исходной БД к целевой, a_{11} – отношение атрибута второй записи исходной БД к целевой базе. Данное значение составляет комбинацию данных и операции преобразования этих данных. Операция скалярного умножения является элементарной операцией перевода (TRL) значения из первой БД во вторую.

Так коэффициенты разложений записываются в отдельную матрицу:

$$A=\begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix}$$

Далее находим матрицу перехода и вычисляем произведение $V' = A^{-1} V$. При этом никаких изменений данных не происходит, а лишь меняется порядок их следования. Непосредственно само произведение V' следует рассматривать как элементарную операцию перевода TRL, при которой на вход подаются данные первой базы, а на выходе результатом преобразования получаем те же данные, но в целевой базе.

Отметим, что разработанный алгоритм позволяет абстрагироваться от типа исходной и целевой базы, модели данных и способа их хранения. Переход между разными видами БД осуществляется лишь путем умножения матрицы V' на матрицу L'. Последняя матрицу, которая учитывает специфические особенности хранения данных в отдельной базе и правила доступа к ним. Так данные в целевой базе можно записать в виде:

$$\bar{a}' = V'a$$
$$\bar{b}' = V'b$$

В функциональном стиле данное выражение имеет вид:

$$a_2 = TRL'(a_1, L\text{^}')$$
$$b_2 = TRL'(b_1, L\text{^}')$$

Данный подход не зависит от конкретной БД, вся реализация возлагается на L'. Метод конвертации также наследует принцип Divide&Conquer, при котором отдельные преобразования данных выполняются элементарными функциями, например: извлечение записи, занесение данных, преобразование типов и. т. д. Это значительно упрощает процесс реализации конвертера и позволяет использовать уже готовые решения при переносе разнотипных баз.

Сложность данного алгоритма состоит в том, что базовые функции, используемые при работе с базами данных разных моделей различны. Согласно стандартной классификации функций по манипуляции данными (CRUD) выделяются четыре базовые функции при работе с данными: создание (Create), чтение (Read), модификация (Update) и удаление (Delete). При использовании матричного подхода для конвертации задействуются только две из них: создание (Create) и чтение (Read), то есть основная сложность при проектировании конвертера заключается в нахождении оператора L', который реализует в себе операции чтения и создания новых данных.

Преобразование хранимых процедур

Данный алгоритм служит для переноса данных, но его можно распространить на хранимые процедуры БД.

По аналогии с векторным пространством вводится понятие линейной формы, при которой линейный оператор из векторного пространства M_k действует в другое векторное пространство G_k: $M_k \to G_k$ над одним полем K. Поле K в алгебре представляет собой множество, над которым определены операции сложения, вычитания, умножения и деления. Но по отношению к БД поле K является множеством над операциями CRUD. Введем понятие элементарного преобразования функции (TRF), в котором процедура a_1 из области определения M_k ставит в соответствие процедуру a_2 из области значений G_k. То есть, если f: A→B, то a_2=TRF (a_1,L'), где L' в данном

Таблица. Автомашины

№	Модель	Год выпуска	Пробег	Цвет
1	Acura Legend	1993	20000	Red
2	Tata-Daewoo Novus SE	2016	15000	White
3	Chery Arrizo 5e 450	2018	10000	Green

```
57
58  if __name__ == '__main__':
59      CONVERT = convertor()
60      A = InverseMatrix()
61      D = psqlQuery()
62
63      ans = CONVERT * (A * D)
64      print(ans)
65      pass
66
```

Основная функциональная часть конвертера

контексте является оператором сопряжения баз. TRF в свою очередь представляет композицию *V'* и элементарных операций перевода хранимых процедур.

Для разных типов баз и моделей данных можно провести соответствие процедур между M_k и G_k. Так, если каждой процедуре $a_1 \in M_k$ соответствует $a_2 \in G_k$, то такое соответствие называется взаимнооднозначным соответствием. Иначе, если для $a_1 \in M_k$ существуют $\{a_2, b_2, c_2, ...n\_2\} \in G_k$, соответствие не взаимнооднозначное. Но на практике при реализации конвертера сложность представляет ситуация, при которой для $a_1 \in M_k$ не существует $a_2 \in G_k$. Такой тип хранимых процедур сложно поддается автоматизации, но при правильном проектировании БД можно найти соответствие a_1 в целевой базе либо составить композицию из элементарных процедур $a_2, b_2, c_2, ... \in G_k$.

Как и при конвертации данных, основную сложность представляет собой поиск оператора L', который по совокупности зависит от модели БД, типов данных, содержимого. Но при формализации структуры базы и ее данных можно избежать проблемы несоответствия.

При наличии оптимизированной, нормализованной исходной базы процесс проектирования и разработки конвертера значительно упрощается. Так как TRL и TRF описывают перевод данных и функций и могут быть реализованы единожды. *L'* как матрица для конвертации данных, так и оператор сопряжения могут быть реализованы для каждого вида баз по отдельности. Так можно по-отдельности провести конвертацию данных и хранимых процедур.

Пример использования конвертера

Для демонстрации работы алгоритма в качестве исходных данных будет выступать таблица «Автомашины» реляционной БД PostgreSQL (таблица).

Для организации целевой базы использовалась резидентная СУБД класса NoSql – Redis. Исходный код конвертера написан на языке высокого уровня Python3. Основная часть, которого, показана на рисунке.

В строках 59-61 происходит инициализация основных элементов конвертера – операции конвертирования, обратной матрицы и исходных данных соответственно. В строке 63 происходит конвертация данных в целевую базу при помощи матричного перемножения обратной матрицы и данных, далее полученный результат умножается на матрицу CONVERT – прообраз L'. В 64 строке результат конвертации выводится на экран.

Фрагмент вывода работы программы:

[{'f444d_model': 'Acura Legend'},
{'f444d_year': '1993'},
{'f444d_mileage': 20000},
{'f444d_color': 'red'},
{'5a135_model': 'Buick Roadmaster'},
.....]

Из примера видно, что атрибуты БД не изменились и данные тоже. Однако данная реализация предлагаемого алгоритма является простейшей и выполнена с расчетом, что атрибуты исходной БД заранее известны.

Заключение

Применение нового метода конвертации позволяет в значительной степени упростить администрирование БД различных форматов и моделей. В статье рассмотрены принципы разработки конвертера для переноса данных между базами. Этот подход устанавливает новые требования при проектировании домена и компонентов. Реализация его позволяет решить проблему моделирования/проектирования объекта не как системы, а как совокупности не связанных или слабо связанных между собой компонент [6]. Алгоритм не требует первоначальной подготовки исходных данных, он дает возможность не учитывать явно структуру БД и ее процедур. Это позволяет сократить трудозатраты и сохранить базы в актуальном состоянии.

Список литературы

1. *David C. Lay.* Linear Algebra and Its Applications / Pearson. – 2016.
2. *Сосинская С. С., Дубинин Д. А.* Изоморфизм различных моделей структурного описания предметной области // Вестник Иркутского государственного технического университета. – 2012. – С.
3. *Кириллов В. В.* Введение в реляционные базы данных / В. В. Кириллов, Г. Ю. Громов. – СПб.: БХВ-Петербург, 2009. – 464 с
4. *Боргест Н. М.* Онтология проектирования: теоретические основы. Часть 1. Понятия и принципы / Н. М. Боргест. Уч. пособие. – Самара: Издательство СГАУ, 2010. – 91 с.
5. *Scott W. Ambler.* Mapping objects to relational databases. What you need to know and why. - developerWorks®IBM Corporation 2000. Отображение объектов в реляционных базах данных. – URL: http://www.ambysoft.com/downloads/persistenceLayer.pdf
6. *Лобановский Ю. И.* Общий классификатор системных проблем. Ч. I: Анализ, синтез, валидация и верификация // Российский технологический журнал. 2018. Т. 6 № 4. // Электронное сетевое издание. 5-12 С.

Маличенко Сергей Владимирович *– магистрант*
ФГБОУ ВО «МИРЭА - Российский технологический университет».