УДК 004.652: 004.655: 004.432.4: 004.4'423

В.И. Есин, М.В. Есина

Харьковский национальный университет им. В.Н. Каразина, Харьков

ИНТЕРПРЕТАТОР ЯЗЫКА ДЛЯ УНИВЕРСАЛЬНОЙ МОДЕЛИ ДАННЫХ

Описывается проблема необходимости создания специального программного обеспечения, которое позволяло бы автоматически трансформировать правильно составленный запрос в терминах предметной области в синтаксически и терминологически корректный запрос к конкретной базе данных с универсальной моделью данных. В качестве решения данной проблемы предлагается интерпретатор языка для универсальной модели данных.

Ключевые слова: *универсальная модель данных, база данных, язык модели данных, интерпретатор языка модели данных.*

Введение

Описание проблемы. Проблема, связанная с созданием формального языка, который позволял бы формировать данные и запросы с использованием естественного языка, волнует мировое сообщество баз данных уже много лет [1, 2], но до сих пор она остается открытой. Двигаясь в направлении решения этой проблемы в рамках универсальной модели данных (УМД) [3], основывающейся на модели данных «объект-событие» [4], был разработан специальный язык модели данных (ЯМД) [5]. Однако для эффективного его использования необходимо специальное программное обеспечение, которое позволяло бы автоматически трансформировать правильно составленный запрос в терминах предметной области в синтаксически и терминологически корректный запрос к конкретной базе данных (БД).

Постановка задачи. Таким образом, возникает задача разработки инструмента автоматической трансформации семантически правильного составленного запроса в терминах модели данных «объект-событие» в синтаксически и терминологически корректный запрос к конкретной БД с УМД.

Интерпретатор ЯМД

Таким программным инструментом может служить так называемый интерпретатор языка модели данных. В настоящее время интерпретатор ЯМД реализован с помощью серверных хранимых процедур, функций и пакетов на языке PL/SQL для СУБД Oracle.

Не вдаваясь в подробности программной реализации интерпретатора ЯМД, рассмотрим его программный интерфейс, который определяется набором необходимых процедур, их параметров, способов обращения к ним, и некоторые положительные стороны его использования.

В качестве процедур интерфейса интерпретатора ЯМД выступают две основные хранимые процедуры, интерпретирующие метаданные и данные любой предметной области, описанные на языке модели данных, в данные, которые записываются, считываются, модифицируются или удаляются из базовых таблиц схемы БД с УМД, назовем их: *DATA_UMD_P* – для данных и *META_DATA_UMD_P* – для метаданных.

Хранимая процедура *DATA_UMD_P* интерпретирует данные любой предметной области, описанные на языке модели данных, в данные, которые записываются, считываются, модифицируются или удаляются из базовых таблиц схемы БД с УМД.

Состав параметров хранимой процедуры *DATA_UMD_P* и их описание:

```
DATA_UMD_P(text_vx_str in var-
char2, pr_type in number,
pr_write_id in number, id in out
number, name_session_proc out
varchar2, text_exit out var-
char2, error_text out varchar2,
cur_name_end_text out
ref_cursor),
```

где *text_vx_str* – (входной параметр) – входная строка метаописаний на ЯМД (запрос в терминах модели данных «объект-событие»); *pr_type* – (входной параметр) – признак необходимости учета типа объекта при выполнении операторов строки метаописания (в некоторых случаях позволяет уменьшить время выполнения); *pr_write_id* – (входной параметр) – признак записи строки метаописания ЯМД и некоторой другой дополнительной информации в таблицу-журнал измененных данных: *1* – записывать данные в таблицу, *0* – не записывать; как правило, *pr_write_id*=1 указывается при выполнении значимых операций записи, удаления, изменения данных для контроля и возможности их

восстановления; *id* – (входной/выходной параметр) – номер кода последнего выбранного или записываемого в БД с УМД элемента данных (класса объекта или события, типа объекта, типа любой характеристики объекта или события, значения списочной характеристики экземпляра объекта или события определенного типа и т.д.), указанного в строке *text_vx_str*. В том случае, если входная строка метаописания составлена с нарушениями правил синтаксиса ЯМД,

id принимает значение равное *–1*;

name_session_proc – (выходной параметр) – имя текущей сессии;

text_exit – (выходной параметр) – имя таблицы БД с УМД, для которой сформирован *id*;

если *id=-1*, то значение параметра *text_exit* (имя таблицы) отсутствует;

error_text – (выходной параметр) – сообщение об ошибке или уведомлении о действиях, которые могут быть совершены;

cur_name_end_text – (выходной параметр – курсорная переменная ссылочного типа – *ref cursor*) – считываемые, записываемые, изменяемые, удаляемые в соответствии со входной строкой метаописания (входной параметр *text_vx_str*), значения.

Хранимая процедура *META_DATA_UMD_P* интерпретирует метаданные любой предметной области, описанные на ЯМД, в данные, которые записываются, считываются, модифицируются или удаляются из базовых таблиц схемы БД с УМД. Состав параметров хранимой процедуры *META_DATA_UMD_P* и их описание:

```
METADATA_UMD_P(text_vx_str in
varchar2, pr_write_id in number,
id in out number,
name_session_proc out varchar2,
text_exit out varchar2, er-
ror_text out varchar2,
cur_name_end_text out
ref_cursor),
```

где назначение параметров хранимой процедуры соответствуют ранее определенным для хранимой процедура *DATA_UMD_P*.

Пример обращения к хранимой процедуре *DATA_UMD_P* из программы, написанной на языке PL/SQL, для получения списка всех экземпляров объектов, всех типов класса объектов «Техническое средство» подразделения «Хар_УГЭС»:

```
declare
id number;
pr_type number;
text_vx_str varchar2(5000);
temp_char varchar2(255);
name_session_proc var-
char2(255);
text_exit varchar2(250);
error_text varchar2(5000);
cur_name_end_text
my_ref_cursor;
pr_write number;
i number;
begin
pr_type:=1;
pr_write:=0;
text_vx_str:='{<Раздел>=Хар_УГЭ
С;/<КлассО>=Техническое средст-
во; <ТипО>=*.*; <ЭкзО>=*.*?;}';
DATA_UMD_BIG_P(text_vx_str,
pr_type, pr_write, id,
name_session_proc, text_exit,
error_text, cur_name_end_text);
i:=0;
LOOP
      fetch cur_name_end_text
into temp_char;
      exit when
cur_name_end_text%NOTFOUND;
      i:=i+1;
dbms_output.put_line('name'||'(
'||i||')='||temp_char);
      END LOOP;
CLOSE cur_name_end_text;
end;.
```

Использование этих интерфейсных процедур интерпретатора ЯМД позволяет не только «освободить» программистов от знания модели базы данных [5], но и «помогает» им решить проблему несоответствия и сложности согласования типов данных (*impedance mismatch*): примитивные типы в языках программирования обычно не соответствуют типам в базе данных; примитивные типы в разных базах данных обычно различаются; поведение null-значений в SQL отличается от поведения null-значений в большинстве процедурных объектно-ориентированных языков и т.д. [6]. А в виду того, что типы всех параметров (входных и выходных) интерфейсных процедур интерпретатора ЯМД, соответствуют типам параметров современных языков программирования (тип данных *varchar2* – соответствует символьной строке переменной длины; тип данных *number* – соответствует целым и вещественным типам; ссылочный тип *ref cursor* – соответствует типу указателей в языках программирования высокого уровня) и не требуют их дополнительного преобразования эту проблему можно избежать.

Таким образом, рассмотренный интерфейс интерпретатора ЯМД обеспечивает наилучшие возможности и для баз данных, и для языков программирования при оптимизации, согласовании типов и модульной разработке. Его хранимые процедуры легко встраиваются в существующие языки программирования, а предлагаемый синтаксис основных операторов ЯМД, указанный во входном параметре *text_vx_str* этих хранимых процедур, поз-

воляет использовать их, не прибегая к сложным прекомпиляторам, ориентированным на тот или иной язык программирования. Предполагается, что модули прикладных программ компилируются независимо.

Интерпретатор разрабатывался и тестировался с учетом возможности работы с БД с УМД в многопользовательском режиме. Основные алгоритмы, реализованные в нем, позволяют не допускать некоторых распространенных ошибок, совершаемых прикладными программистами с малым опытом разработки приложений для БД. Классический пример – *потерянное изменение* (*обновление*). Эта проблема баз данных проявляется постоянно, когда прикладные программисты, не имеющие достаточного опыта работы с базами данных (а имея лишь общее представление об операторах `SELECT`, `INSERT`, `UPDATE`, `DELETE` и ничего не знающих о блокировках – механизме, используемом для управления одновременным доступом к общему ресурсу), получают задание создать для них приложение. Они начинают писать программы, которые очень часто ведут себя неадекватно: получаемые с помощью них результаты работы с БД кажутся случайными и абсолютно невоспроизводимы в управляемой среде тестирования, что приводит разработчика к мысли, что это, возможно, ошибка пользователя. В итоге пользователи полностью перестают им доверять.

Если обращение к БД с УМД происходит с применением строк метаописания ЯМД, которые обрабатываются интерпретатором, то ошибок, связанных с потерянным обновлением, принципиально не может произойти, так как в строке метаописания указывается и «старое», и «новое» (модифицируемое) значение. Поэтому если «старое» значение в БД с УМД уже отсутствует (изменено другим пользователем), то текущему пользователю выдается ошибка о несоответствии значений, указанных в строке метаописания ЯМД и хранящимся в БД с УМД.

Код интерпретатора, представляющего собой набор серверных процедур, функций и пакетов, компилируется заранее, а это значит, что благодаря возможности проведения, так называемой, статической оптимизации запросов выполняться он будет быстрее, чем некоторый подобный обычный запрос. Основное преимущество статической оптимизации запросов состоит в устранении дополнительной нагрузки из-за проведения процедур оптимизации в процессе выполнения запроса. В результате появляется возможность более тщательного подбора оптимальной стратегии за счет анализа большего числа возможных вариантов. Поэтому вероятность обнаружения более приемлемой стратегии значительно возрастает. В случае многократно выполняемых запросов дополнительные затраты времени на поиск самых оптимальных планов их выполнения могут дать существенный выигрыш в производительности.

Основной недостаток этого метода состоит в том, что выбранная стратегия выполнения запроса, которая была оптимальной при компиляции запроса, в момент выполнения запроса может оказаться уже не оптимальной. Для преодоления этого недостатка может применяться комбинированный подход, предусматривающий повторную оптимизацию запроса в том случае, когда система обнаруживает, что статистические показатели базы данных существенно изменились с момента предыдущей оптимизации данного запроса. Альтернативный вариант предусматривает автоматическую компиляцию процедур интерпретатора ЯМД при первом его выполнении в каждом из сеансов.

Интерпретатор ЯМД имеет достаточно мощную систему диагностики некорректно введенных строк метаописания и операторов языка, а также поясняющих сообщений при правильном исполнении операторов, которые можно использовать при отладке строк метаописаний ПрО и тестирования части схемы БД с УМД. Ниже приведены некоторые такие сообщения и их смысл:

- *«Не задан раздел во входной строке»* – в строке метаописания не задан оператор <*Раздел*>;
- *«Не определен раздел – <name>»* – в строке метаописания задано имя <*name*> не существующего в БД с УМД раздела;
- *«Объект – <NAME_OBJECT> типа – <TYPE_NAME> отсутствует в БД и данный объект может быть занесен в БД»* – в строке метаописания после служебного слова <*ЭкзО*> задан экземпляр объекта с именем <*NAME_OBJECT*> и типом <*TYPE_NAME*>, который отсутствует в БД с УМД, и который при необходимости может быть занесен в нее;
- *«Событие – <EVENT_NAME> для объекта <NAME_OBJECT> отсутствует, и оно может быть занесено в БД»* – в строке метаописания задан экземпляр события класса <*EVENT_NAME*> для экземпляра объекта с именем <*NAME_OBJECT*>, который отсутствует в БД с УМД, и этот экземпляр события при необходимости может быть занесен в нее;
- *«У экземпляра объекта не может быть подчиненных объектов с одинаковым именем <NAME_OBJECT> хотя и разных типов»* – в строке метаописания указан экземпляр объекта с именем <*NAME_OBJECT*> и определенным типом. В тоже время в БД уже есть объект с таким же именем <*NAME_OBJECT*>, но другим типом. Это может привести к нарушению целостности данных, так как у любого объекта не может быть, подчиненных объектов с одинаковыми именами, хотя и разных типов;

- *«Нет заданного класса <OBJ_CLASS_NAME> объекта»* – при вводе данных в строке метаописания после служебного слова <*КлассО*> задано имя <*OBJ_CLASS_NAME*> не существующего в БД с УМД класса объекта (не занесены соответствующие метаданные о классе объекта – его имя, или имя класса объекта указано неверно);
- *«Экземпляр объекта <id> переименован на <NAME_UPDATE_OBJECT>»* – указанное в строке метаописания после служебного слова <*ОбновитьЭкзО*> новое имя экземпляра объекта будет внесено в БД с УМД вместо старого для экземпляра объекта с номером <*id*> (имя которого указано в строке метаописания за служебным словом <*ЭкзО*>), и некоторые другие сообщения.

В заключение сделаем несколько замечаний относительно использования интерпретатора ЯМД:

- если используются служебные слова операторов ЯМД в различных национальных алфавитах [5], то интерпретатор должен их поддерживать;
- в случае, когда пользователь работает с расширенной схемой БД с УМД, обращение к данным, находящимся в его собственных таблицах с использованием (или без) разработанных им представлений и прочих объектов схемы БД, должно осуществляться пользователем самостоятельно, без применения ЯМД (например, с помощью языка SQL-запросов).

Выводы

Разработанный инструмент автоматической трансформации семантически правильного составленного запроса в терминах модели данных «объект-событие» в синтаксически и терминологически корректный запрос к конкретной БД с УМД, выполненный в виде интерпретатора ЯМД позволяет существенно «облегчить» работу программистов по написанию всевозможных приложений для обращения к базам данных с универсальной моделью данных и повысить эффективность таких приложений.

Список литературы

1. *Кузнецов С.Д. Крупные проблемы и текущие задачи исследований в области баз данных [Электронный ресурс] / С.Д. Кузнецов. – Режим доступа к ресурсу: http://www.citforum.ru/database/articles/problems/.*
2. *Кузнецов С.Д. Предвестники новых манифестов управления данными [Электронный ресурс] / С.Д. Кузнецов. – Режим доступа к ресурсу: http://www.citforum.ru/database/articles/premanifest/.*
3. *Есин В.И. Семантическая модель данных "объект-событие" / В.И. Есин // Вісник Харківського національного університету. – Х.: Харьковский национальный университет им. В.Н. Каразина, 2010. – № 925. – С. 65-73 – (Серія : Математичне моделювання. Інформаційні технології. Автоматизовані системи управління" ; вып. 14).*
4. *Есин В.И. Универсальная модель данных и ее математические основы / В.И. Есин // Системи обробки інформації. – Х.: Харківський університет Повітряних Сил, 2011. – Вип. 2 (92). – С. 21-24.*
5. *Есин В.И. Язык для универсальной модели данных / В.И. Есин, М.В. Есина // Системи обробки інформації. – Х.: Харківський університет Повітряних Сил, 2011. – Вып. 5 (95). – С. 193-197.*
6. *Кук В. Интеграция языков программирования с базами данных: в чем состоит проблема? [Электронный ресурс] / В. Кук, А. Ибрагим / Перевод: С. Кузнецов. – Режим доступа к ресурсу: http://citforum.ru/database/articles/impedance_mismatch/.*

Поступила в редколлегию 26.09.2011

Рецензент: д-р техн. наук проф. Л.С. Сорока, Харьковский национальный университет им. В.Н. Каразина, Харьков.

ІНТЕРПРЕТАТОР МОВИ ДЛЯ УНІВЕРСАЛЬНОЇ МОДЕЛІ ДАНИХ

В.І. Єсін, М.В. Єсіна

Описується проблема необхідності створення спеціального програмного забезпечення, яке дозволяло б автоматично трансформувати правильно складений запит в термінах наочної області в синтаксично і термінологічно коректний запит до конкретної бази даних з універсальною моделлю даних. Як вирішення даної проблеми пропонується інтерпретатор мови для універсальної моделі даних.

Ключові слова: *універсальна модель даних, база даних, мова моделі даних, інтерпретатор мови моделі даних.*

INTERPRETER OF LANGUAGE FOR THE UNIVERSAL DATA MODEL

V.I. Yesin, M.V. Yesina

The problem of necessity to create special software that would allow automatic transformation of a correctly formatted request in terms of subject domain in syntactic and terminology correct request for a concrete database with the universal data model is described. The language interpreter is offered as a decision of this problem for the universal data model.

Keywords: *universal data model, database, language of data model, language interpreter of data model.*