Полтавцев А.А., Чибисов Ю.А. ADT как средство поддержки реляционным сервером объектной разработки информационной системы. // Проблемы информатики в образовании, управлении, экономике и технике: Сб. статей XVIII Междунар. научно-техн. конф. – Пенза: ПДЗ, 2018. – С. 222-226.

УДК 004.822

ADT КАК СРЕДСТВО ПОДДЕРЖКИ РЕЛЯЦИОННЫМ СЕРВЕРОМ ОБЪЕКТНОЙ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

А.А. Полтавцев, Ю.А. Чибисов

ADT AS A RELATIONAL SERVER SUPPORTING FACILITY AN INFORMATION SYSTEM OBJECT DEVELOPMENT

A.A. Poltavtsev, Yu.A. Chibisov

Аннотация. В статье рассматриваются современные подходы к проектированию и реализации программного обеспечения информационных систем. Предлагается использовать такие объектно-реляционные технологии, как абстрактные типы данных (ADT) в форме сложных пользовательских типов (UDT) и объектно-реляционные трансляторы.

Ключевые слова: информационная система, проектирование, модель, структуры, задачи.

Abstract. The article reviews modern approaches to the design and implementation of software information systems. It is proposed to use such object-relational technologies as abstract data types (ADT) in the complex user-defined types form (UDT) and object-relational translators.

Keywords: information system, design, model, structures, tasks.

Изменения в программировании и расширение типов данных в реляционных серверах обусловлены SQL стандартами, в основном SQL:1999 и SQL:2003. Как указывается в [1], наиболее серьезные изменения языка SQL, специфицированные в части 2 стандарта SQL:2003, касаются следующих аспектов: типы данных; подпрограммы, вызываемые из SQL; расширенные возможности оператора CREATE TABLE; новый объект схемы – генератор последовательностей; новые виды столбцов, идентифицирующие столбцы (identity column) и генерируемые столбцы (generated column); новый оператор MERGE. С точки зрения поддержки объектной разработки одним из наиболее важных требований стандарта является расширение системы типов.

Наиболее распространенным вариантом использования пользовательских типов является добавление новых скалярных типов (UDT). UDT расширяет встроенный тип данных (например, INTEGER или VARCHAR) специальной семантикой. Например, может быть определен тип данных JPEG. Этот тип хранится в реляционном сервере, например, SQL Server как тип данных IMAGE, но он может быть расширен путем добавления "поведения" через пользовательские функции, такие как get_background _color или get_foreground_color. Добавление функций к простому типу

IMAGE позволяет сформировать SQL-запросы "выбрать все строки в базе данных, где JPEG столбец x имеет цвет фона красного цвета". Стандарт не только ввел понятие расширение типа, но и определил правила его использования. Например, если тип JPEG и тип GIF являются разными типами, которые используют общий базовый тип IMAGE, то типы JPEG и GIF не могут сравниваться без использования оператора CAST. CAST указывает, что программист знает, что сравниваются два разных типа. В частности, использование get_background_color типа JPEG может привести к неправильным результатам для столбца с типом GIF.

Сложные типы содержат несколько атрибутов. Примером сложного типа является тип "person" (человек), который состоит из имени, адреса и номера телефона (рисунок). Сложные типы данных нарушают первую нормальную форму:

OrderID	OrderDate	Customer					
		Name	Address				Phone
			Street	City	Province	Postcode	

Тип person может быть использован в нескольких таблицах с сохранением его структуры, т.е. одни и те же атрибуты и функции применимы к типу person, даже если столбец с типом person используется в трех несвязанных таблицах.

В дополнение к разрешению сложных типов в стандарте определены типы, которые могут быть ссылками на сложные типы. Например, тип человека может содержать ссылку на тип "адрес" в другой таблице:

```
CREATE TYPE PERSON (
first_name VARCHAR(30),
last_name VARCHAR(30),
pers_address REF(ADDRESS SCOPE ADDR_TAB)
);
CREATE TYPE ADDRESS (
street VARCHAR(20),
city VARCHAR(30),
province VARCHAR(10),
postal_code VARCHAR(10)
);
CREATE TABLE ADDR_TAB (
addr_oid BIGINT,
addr ADDRESS
);
```

В сложном типе могут быть определены методы, а язык SQL был расширен для поддержки использования атрибутов сложного типа в запросах. Пример сложного типа и оператора SELECT, который его использует, может выглядеть следующим образом:

```
SELECT ADDRESS FROM ADDR_TAB
WHERE ADDR.addr_city like 'Sea%'
```

Конечно, необходимо отметить следующее. Некоторые объектно-ориентированные методы можно реализовать на сервере СУБД через сложные пользовательские типы, но объектные технологии в значительно большей степени используются на уровнях клиента и бизнес-логики. Большинство систем графического интерфейса пользователей, клиентских приложений и даже самих библиотек классов (например, .NET) основаны на объектно-ориентированной реализации. После того, как Буч, Якобсон и Рамбо популяризировали декомпозицию любой компьютерной проблемы в набор хорошо определенных методов моделирования, в вычислительном мире популярным стало высказывание: "Программисты любят программировать объектами. Это облегчает их работу, поскольку облегчает визуализацию проблемной области ". С тех пор такие технологии, как проектирование сверху вниз и структурированное программирование, отошли на задний план. Но в реляционной СУБД данные разбиваются на кортежи и отношения, а не на графы объектов. В этом заключается знаменитое объектно-реляционное "несоответствие импеданса" (object-relational impedance mismatch).

Одним из вариантов решения этой проблемы является ObjectSpaces – разработанная фирмой Microsoft технология объектно-реляционной интеграции на стороне клиента. Эта технология является частью ADO.NET и решает проблему перевода объектных графов, составляющих объектную модель программирования в реляционные структуры. Поскольку большинство приложений построено вокруг объектных моделей, стандартный набор API для облегчения взаимодействия между реляционными данными и объектами является необходимым дополнением.

Популярная концепция "объект-сущность" показала неудовлетворительные результаты. Использование объектов-сущностей приводило к дублированию информации и/или блокировкам базы данных, слишком длительным, чтобы быть допустимыми для использования. В ObjectSpaces используются специальные методы оптимизации, такие как частичная и ленивая загрузка (partial and lazy loading) графов объектов, включая программную спецификацию глубины загрузки графа и интеллектуальное обновление и удаление строк базы данных, соответствующих экземплярам графа, для повышения производительности. В ObjectSpaces запросы и мутаторы для доступа к графам объектов можно указать в качестве хранимых процедур, чтобы воспользоваться вычислительной мощью СУБД. Кроме того, можно запрашивать графы объектов на стороне клиента через класс ObjectQuery и через простой, но мощный язык предикатов, известный как

OPath. Использование OPath напоминает использование XPath для выбора подмножества узлов в документе XML или использование предложения WHERE в SQL-запросе.

Библиографический список

1. Andrew Eisenberg, Jim Melton, Krishna Kulkarni, Jan-Eike Michels, Fred Zemke. SQL:2003. ACM SIGMOD Record 33, No. 1 (March 2004).

Полтавцев Анатолий Алексеевич
Тверской государственный технический университет, г. Тверь, Россия

Poltavtsev A.A.
Tver State Technical University, Tver, Russia

Чибисов Юрий Алексеевич
Тверской государственный технический университет, г. Тверь, Россия

Chibisov Yu.A.
Tver State Technical University, Tver, Russia