Полтавцев А.А., Чибисов Ю.А. Особенности реализации и использования ADT в реляционной СУБД SQL Server. // Проблемы информатики в образовании, управлении, экономике и технике: Сб. статей XVIII Междунар. научно-техн. конф. – Пенза: ПДЗ, 2018. – С. 219-222.

УДК 004.822

ОСОБЕННОСТИ РЕАЛИЗАЦИИ И ИСПОЛЬЗОВАНИЯ ADT В РЕЛЯЦИОННОЙ СУБД SQL SERVER

А.А. Полтавцев, Ю.А. Чибисов

FEATURES ADT IMPLEMENTATION AND USE IN THE RELATIONAL RDBMS SQL SERVER

A.A. Poltavtsev, Yu.A. Chibisov

Аннотация. В статье рассматриваются современные подходы к проектированию и реализации программного обеспечения информационных систем. Анализируется технология использования абстрактных типов данных и функций агрегирования в SQL SERVER для поддержки хранения объектов.

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

Abstract. The article reviews modern approaches to the design and implementation of software information systems. The technology of using abstract data types and aggregation functions in SQL SERVER to support object storage analyzes.

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

SQL Server поддерживает собственную концепцию пользовательского типа данных. Эти типы данных известны как "типы псевдонимов" (alias types) [1] и определяются с помощью системной хранимой процедуры sp_addtype. Эти типы данных разделяют некоторую общую функциональность с типами SQL. Они должны быть получены из встроенного типа данных SQL Server. Можно добавить ограничения целостности с помощью правил SQL Server. Правило создается с использованием CREATE RULE и связывается с пользовательским типом с помощью sp_bindrule. Один пользовательский тип данных может использоваться в нескольких таблицах, а одно правило может быть привязано к нескольким пользовательским типам:

```
— define two user-defined types
EXEC sp_addtype iq, 'FLOAT', 'NULL'
EXEC sp_addtype shoesize, 'FLOAT', 'NULL'
— specify constraints
CREATE RULE iq_range AS @range between 1 and 200
CREATE RULE shoesize_range AS @range between 1 and 20
— bind constraint to type
EXEC sp_bindrule 'iq_range', 'iq'
EXEC sp_bindrule 'shoesize_range', 'shoesize'
```

Поскольку правило можно считать поведением, то пользовательские типы добавляют определяемое пользователем поведение. В отличие от других типов SQL, они могут не иметь ассоциированных пользовательских функций, которые привязаны к типу. Например, хотя мы определили тип и ограничили его значения числами с плавающей запятой от 1 до 20, мы не можем связать функцию derive_suit_size_from_shoesize с данным типом. Это было бы возможно, если бы shoesize был стандартным производным типом SQL. Кроме того, пользовательские типы SQL Server сопоставимы без использования оператора CAST, если сопоставимы базовые встроенные типы. Спецификация SQL указывает, что определяемый пользователем тип должен быть добавлен к встроенному или определяемому пользователем типу, прежде чем он может использоваться в операции сравнения, а попытка применить оператор CAST к определенному пользователем типу в SQL Server вызовет ошибку:

```
— use the type
CREATE TABLE people (
personid INTEGER,
iq iq,
shoe shoesize,
spouse_shoe shoesize
)
SELECT * from people WHERE iq < shoe
— SQL:1999 syntax (invalid in SQL Server)
— SELECT * FROM people WHERE CAST(iq AS shoesize) <
shoe
```

Начиная с версии SQL Server 2005 в СУБД усилена поддержка сложных пользовательских типов SQL:1999. Расширенные типы данных должны быть определены как классы .NET и не могут быть определены в Transact-SQL, хотя они доступны в хранимых процедурах T-SQL, пользовательских функциях и другом процедурном коде. Эти классы (типы) могут иметь функции-члены, доступные в T-SQL, и, кроме того, они могут иметь функции-мутаторы, которые можно использовать в операторах UPDATE.

В дополнение к возможности определять много типов на основе одного встроенного типа данных SQL Server 2005 позволяет пользовательским типам иметь несколько атрибутов. Таким образом, это действительно сложный тип SQL:1999. После определения в каталоге SQL Server на новом типе может быть определен столбец таблицы. Переменные данного типа могут использоваться в хранимых процедурах, а атрибуты и методы этого типа могут использоваться в вычислениях и пользовательских функциях. Рассмотрим пример определения сложного типа ComplexNumber и его использования в качестве столбца таблицы.

```
CREATE TYPE ComplexNumber
EXTERNAL NAME SomeTypes:ComplexNumber
GO
CREATE TABLE Transforms(
transform_id BIGINT,
transform_input  ComplexNumber,
transform_result ComplexNumber)
GO
```

Таким образом, сложные типы SQL Server действительно расширяют реляционные типы данных. SQL Server сервер не знает отношения наследования между типами (хотя наследование может использоваться в реализации) или полиморфизм методов (как и в традиционных объектно-ориентированных системах). То есть, хотя мы можем определить сложный пользовательский тип, называемый Person, который содержит несколько значений данных (переменные-члены) и методы, и определить сложный тип, называемый Author, который наследуется от Person, мы не можем вызывать методы типа Person при использовании Author тип или сравнивать экземпляры типов Author и Person. Для этого лучше создать и применить определенный на домене скалярный тип, как например ComplexNumber.

Помимо поддержки пользовательских типов, SQL Server начиная с версии 2005 поддерживает пользовательские функции агрегирования. Они (типы) расширяют концепцию пользовательских функций, могут быть записаны на любом языке .NET (но не в T-SQL). Спецификация стандарта SQL определяет пять агрегатов, которые должны поддерживать базы данных: MAX, MIN, AVG, SUM и COUNT. SQL Server реализует надмножество спецификации, включая такие агрегаты, как стандартное отклонение и дисперсию. Используя поддержку SQL Server языков .NET, пользователям не нужно дожидаться, когда механизм базы данных сможет реализовать свой конкретный агрегат, специфичный для домена. Определяемые пользователем агрегаты могут быть даже определены по пользовательским типам, например агрегата для типа данных ComplexNumber.

Поддержка пользовательских типов и агрегатов позволяет SQL Server приблизиться к соблюдению стандартов SQL: 1999 и SQL: 2003.

Библиографический список

1. Bob Beauchemin, Niels Berglund, Dan Sullivan. First Look at SQL Server 2005 for Developers. Addison-Wesley. 2004. 737 с.

Полтавцев Анатолий Алексеевич
Тверской государственный технический университет, г. Тверь, Россия
E-mail: aapolt@gmail.com

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

Чибисов Юрий Алексеевич
Тверской государственный
технический университет,
г. Тверь, Россия
E-mail: yura.chibisov@gmail.com

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