УДК 004.6

Реализация метаграфового хранилища на языке программирования F#
F# implementation of metagraph storage

Гапанюк Ю.Е., *к.т.н., доцент,* gapyu@bmstu.ru
Азибекян А.С., *студент,* aren.azibekyan@mail.ru
Лещев А.О., *студент,* matshch@gmail.com
Лясковский М.А., *студент,* maxavatar007@gmail.com
Мельников К.И., *студент,* hexagramg@gmail.com
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

В настоящее время графовые модели на основе сложных сетей все чаще используются для решения широкого класса задач. В качестве одного из вариантов такой модели нами предлагается использовать метаграфовый подход. Рассмотрим основные элементы метаграфовой модели на основе статей [1, 2].

Атрибут представляет собой пару ключ-значение: $atr =< key, value >$, где key – наименование атрибута; value – значение атрибута.

Вершина метаграфа v характеризуется множеством атрибутов atr_k: $v = \{atr_k\}$

Ребро метаграфа e характеризуется множеством атрибутов, исходной v_S и конечной v_E вершинами: $e = \langle v_S, v_E, \{atr_k\} \rangle$.

Фрагмент метаграфа: $MG_i = \{ev_j\}, ev_j \in (V \cup E \cup MV)$, где MG_i – фрагмент метаграфа; ev_j – элемент, принадлежащий объединению множеств вершин V, метавершин MV и ребер E метаграфа.

Метавершина метаграфа $mv_i = \langle \{atr_k\}, \{ev_j\} \rangle$ характеризуется множеством атрибутов и вложенным фрагментом метаграфа.

Одним из важных вопросов работы с метаграфами является вопрос хранения метаграфовой модели данных и разработки программных модулей для доступа к хранилищу. Для реализации программных модулей была

использована платформа .NET/Mono. Структура метаграфового хранилища представлена на рис. 1.

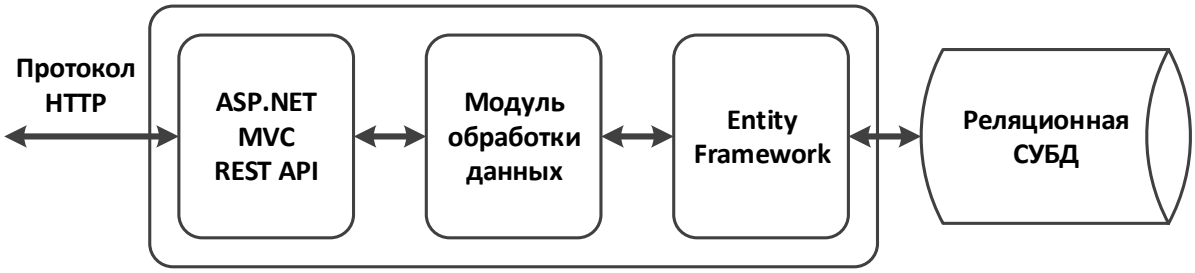

Рис. 1. Структура метаграфового хранилища.

Рассмотрим компоненты структуры более подробно. Для хранения данных используется реляционная СУБД. В настоящее время эксперименты были проведены с СУБД MySQL и PostgreSQL. Результаты экспериментов показали, что прототип на основе PostgreSQL обладает лучшей производительностью.

Библиотека объектно-реляционного отображения Entity Framework используется для отображения метаграфовой модели, представленной в виде классов языка F#, в схему реляционной СУБД. Entity Framework обладает хорошей производительностью и используется в большом количестве .NET/Mono проектов. Изначально Entity Framework был разработан для использования с языком программирования C#, но в настоящее время также адаптирован для использования с F#.

Для предоставления программного интерфейса к хранилищу используется REST API на основе протокола HTTP. Для реализации REST API используется библиотека разработки веб-приложений ASP.NET MVC, реализующая шаблон проектирования MVC (Model-View-Controller). Как и Entity Framework, библиотека ASP.NET MVC была разработана для использования с языком программирования C#, но в настоящее время адаптирована для использования с F#.

Основным модулем метаграфового хранилища является модуль обработки данных. Именно при разработке этого модуля были широко использованы преимущества языка F#, рассмотренные в нашей статье [3].

В отличие от классического объектно-ориентированного языка C#, язык F# является мультипарадигменным и поддерживает объектно-ориентированное (ООП) и функциональное программирование, при этом в языке активно используются обобщенные типы. Поддержка ООП позволяет использовать F# с объектно-ориентированными инструментами, в частности с библиотекой объектно-реляционного отображения Entity Framework. Одновременно с этим, можно использовать функциональные возможности, как для описания сложных типов, так и для обработки данных этих типов.

Одним из основных преимуществ языка F# для описания сложных типов является возможность использования размеченных объединений (discriminated unions). Синтаксически в F# для этой конструкции используется вертикальная черта. Рассмотрим фрагмент описания типов на языке F#, соответствующий приведенному выше описанию метаграфовой модели (код снабжен детальными комментариями):

```
/// класс Атрибут
type public Attribute<'T when 'T:null>() =
    /// наименование атрибута
    member val name:string="" with get, set
    /// значение атрибута, 'T - обобщенный тип
    member val value:'T=null with get, set

/// класс Вершина
type public Vertex<'T when 'T:null>() =
    /// множество (список) атрибутов
    member val attributes:seq<Attribute<'T>>=null with get, set

/// класс Ребро
type public Edge<'T when 'T:null>() =
    /// множество (список) атрибутов
    member val attributes:seq<Attribute<'T>>=null with get, set
    /// исходная вершина
    member val vs:Option<Vertex<'T>>=None with get, set
    /// конечная вершина
    member val ve:Option<Vertex<'T>>=None with get, set

/// класс Элемент фрагмента метаграфа
type public MetagraphFragmentItem<'T when 'T:null> =
    | MetagraphFragmentVertex of Vertex<'T>
    | MetagraphFragmentEdge of Edge<'T>
    | MetagraphFragmentMetaVertex of MetaVertex<'T>

/// класс Метавершина метаграфа
and public MetaVertex<'T when 'T:null>() =
    /// множество (список) атрибутов
```

```
member val attributes:seq<Attribute<'T>>=null with get, set
/// фрагмент метаграфа
member val fragment:seq<MetagraphFragmentItem<'T>>=null with get, set
```

В приведенном выше коде необходимо отметить следующие моменты:

- Значением атрибута является значение обобщенного типа 'T (перед названием обобщенного типа в F# принято ставить кавычку).
- Элемент фрагмента метаграфа определяется с использованием размеченного объединения.
- Элемент фрагмента и метавершина метаграфа определены друг через друга. Для ООП языков реализация такого определения может вызвать проблему. В функциональном подходе типы данных, реализующие такое определение, называются взаимно-рекурсивными. В языке F# для задания таких типов используют конструкцию type … and …

Для обработки данных были использованы такие функциональные возможности языка F# как «list comprehensions», лямбда-выражения, функции высших порядков.

Работа программного модуля была протестирована как в среде .NET под управлением ОС Windows, так и в среде Mono под управлением ОС Linux. Результаты тестирования показали, что разработанная библиотека надежно функционирует в обеих средах под обеими операционными системами. Компилятор с языка F# надежно функционирует и в среде .NET и в среде Mono.

В настоящее время компания Microsoft активно ведет разработку кроссплатформенной среды .NET Core, которая функционирует под управлением ОС Windows, macOS, различных версий Linux. Версия .NET Core 2.0 была выпущена в августе 2017 года.

В перспективе предполагается что среда .NET Core будет использоваться в большинстве современных операционных систем вместо .NET (ОС Windows) и Mono (macOS, Linux). Поэтому перспективным направлением работы является портирование метаграфового хранилища с платформы .NET/Mono на платформу .NET Core.

Таким образом, использование объектно-ориентированных и функциональных возможностей языка F# позволило реализовать модуль метаграфового хранилища. Использование размеченных объединений и взаимно-рекурсивных типов позволило упростить описание классов хранилища. Реализация заняла около полутора тысяч строк кода на языке F#.

Список литературы

1. Черненький В.М., Гапанюк Ю.Е., Ревунков Г.И., Терехов В.И., Каганов Ю.Т. Метаграфовый подход для описания гибридных интеллектуальных информационных систем. Прикладная информатика. 2017. № 3 (69). Том 12. С. 57–79.
2. Гапанюк Ю.Е., Ревунков Г.И., Федоренко Ю.С. Предикатное описание метаграфовой модели данных. Информационно-измерительные и управляющие системы. 2016. Выпуск № 12. С. 122-131.
3. Лещев А.О., Мельников К.И. F#: функциональный язык платформы .NET. Молодежный научно-технический вестник МГТУ им. Н.Э. Баумана. 2016. Выпуск №9.