В.А. Филипович, А.В. Параничев

Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, г. Санкт-Петербург

УДК 004.4

ИНСТРУМЕНТАРИЙ LINQ ДЛЯ ПРОГРАММНОЙ ОБРАБОТКИ РЕЗУЛЬТАТОВ ЗАПРОСОВ С ПОМОЩЬЮ ОПЕРАЦИЙ НАД МНОЖЕСТВАМИ

Аннотация. В работе обсуждаются инструменты LINQ для составления запросов на платформе dotNet на языке C#. Приведена программная реализация обработки результатов запросов с помощью операций над множествами.

Ключевые слова: язык запросов, база данных, LINQ, dotNet, C#, SQL, диаграммы Эйлера-Венна.

Введение

LINQ (Language INtegrated Query, интегрированный язык запросов) представляет собой структурированный язык запросов (SQL, Structured Query Language), интегрированный в язык программирования C# для удобного обращения к источнику данных на платформе dotNet [1], [2]. В качестве источника базы данных (БД) LINQ может выступать объект, в котором реализуется интерфейс IEnumerable (в частности, стандартные коллекции и массивы), либо представлен XML-документ, БД SQL (как правило, используется тип DataSet) или веб-сервис [1].

Постановка задачи

В языке C# определены ключевые слова для интеграции с БД, реализованной или совместимой с LINQ (представлены в пространстве имен System.Linq); основные ключевые слова приведены в табл. 1 [3-4]. Кроме того, существует набор методов расширения (extension methods),

позволяющих реализовать простые операции над множествами ([5; 19-24]; [5; 384-389]) как показано в табл. 2 (такие методы определены в пространстве имен System.Collections.Generic) [7].

Таблица 1. Основные ключевые слова LINQ

Набор ключевых слов	Описание использования в LINQ
from, in	определяет каркас выражения LINQ для извлечения данных
where	определяет условия, при которых происходит извлечение данных
select	определяет выборку считываемых данных
join, on, equal, into	объединяет (*join*) выборки данных по заданному ключу на уровне запроса
orderby, ascending, descending	упорядочивает результат выборки данных по возрастанию (*ascending*) или убыванию (*descending*)
group, by	генерирует подмножество данных, группируемых по определенному значению

Таблица 2. Операции расширения в LINQ для работы с множествами

Операция над множествами (результатами запросов) Q_a и Q_b	Операция расширения LINQ для результатов запросов	Описание результата
$Q_a \cup Q_b$	query_a. *Union*(query_b)	объединяет результаты запросов, исключая дублирующие записи
$Q_a \cap Q_b$	query_a. *Intersect*(query_b)	вычисляет общие записи для результатов запросов, исключая одинаковые

$Q_a \setminus Q_b$	query_a. *Except*(query_b)	выполняет операцию разности множеств, исключая одинаковые записи
$Q_a \Delta Q_b$	–	выполняет операцию симметричной разности множеств, исключая дублирующие записи

Таким образом, инструменты LINQ позволяют (табл. 1-2):

- выполнить построение SQL-подобных запросов к БД, встраивая такие запросы в код на языке C#;
- подключить методы расширения к LINQ, в частности, для выполнения операций с результатами запросов как со множествами (операции объединения, пересечения и разности).

Программная реализация

Продемонстрируем выполнение операций, представленных в табл. 2, на примере двух множеств (рис. 1), соответствующих результатам запроса к БД (схематично представлены на рис. 2 и 3). Результат представим в виде приложения на языке C# (рис. 4), используя онлайн-компилятор для работы с кодом в консоли [8].

Рис. 1. Пример результатов запроса к БД в виде двух множеств.

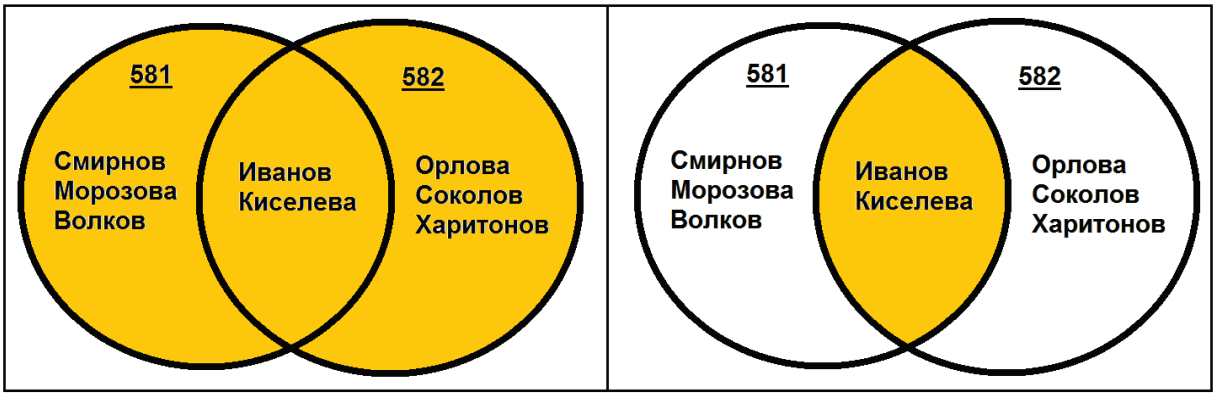

Рис. 2. Схематичное представление результатов операций объединения ($Q_{581} \cup Q_{582}$: слева) и пересечения ($Q_{581} \cap Q_{582}$: справа)

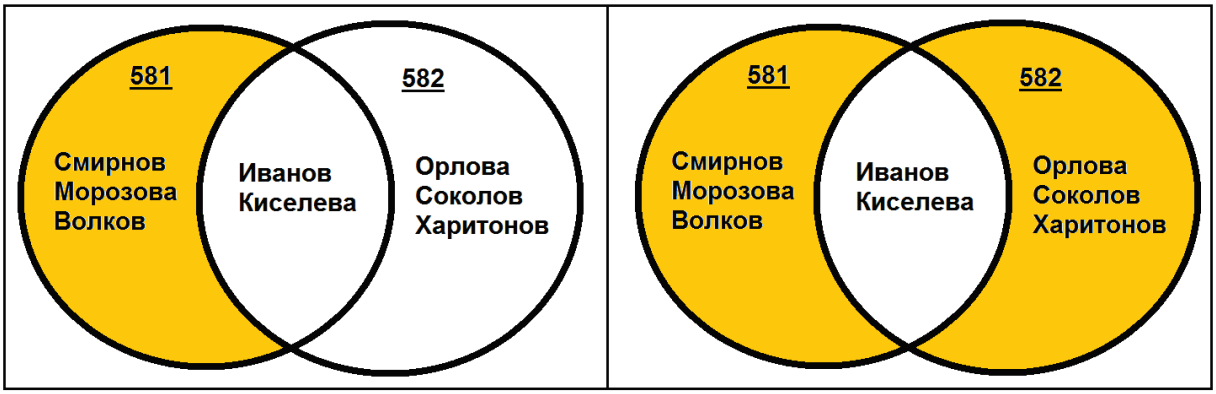

Рис. 3. Схематичное представление результатов операций несимметричной ($Q_{581} \setminus Q_{582}$: слева) и симметричной разности ($Q_{581} \Delta Q_{582}$: справа)

```
using System; using System.Linq; using System.Collections.Generic;
public class Program
{   public static void Main()
    { // создаем 2 списка преподавателей для двух групп и выводим на экран
List<string> k581 = new List<string> {"Иванов", "Смирнов", "Киселева", "Морозова", "Волков"};
List<string> k582 = new List<string> {"Иванов", "Орлова", "Киселева", "Соколов", "Харитонов"};
    Console.Write("--- Список преподавателей группы К581: --- ");
    foreach(string s in k581) Console.Write(String.Format("{0} ", s));
    Console.Write("\n--- Список преподавателей группы К582: --- ");
    foreach(string s in k582) Console.Write(String.Format("{0} ", s));

    Console.Write("\n\n---Результат операции Union():--- ");
    var list_union = (from a in k581 select a).Union(from b in k582 select b);
    foreach(string s in list_union) Console.Write(String.Format("{0} ", s));

    Console.Write("\n\n---Результат операции Intersect():--- ");
    var list_intersect = (from a in k581 select a).Intersect(from b in k582 select b);
    foreach(string s in list_intersect) Console.Write(String.Format("{0} ", s));

    Console.Write("\n\n---Результат операции Except():--- ");
    var list_except = (from a in k581 select a).Except(from b in k582 select b);
    foreach(string s in list_except) Console.Write(String.Format("{0} ", s));

    Console.Write("\n\n---Результат операции симметричной разности:--- ");
    var list_distinct = (from a in k581 select a).Except(from b in k582 select b).
                  Union((from a in k582 select a).Except(from b in k581 select b));
    foreach(string s in list_distinct) Console.Write(String.Format("{0} ", s));
    }
}
```

```
- Список преподавателей группы К581: --- Иванов Смирнов Киселева Морозова Волков
- Список преподавателей группы К582: --- Иванов Орлова Киселева Соколов Харитонов

-Результат операции Union():--- Иванов Смирнов Киселева Морозова Волков Орлова Соколов Харитонов

-Результат операции Intersect():--- Иванов Киселева

-Результат операции Except():--- Смирнов Морозова Волков

-Результат операции симметричной разности:--- Смирнов Морозова Волков Орлова Соколов Харитонов
```

Рис. 4. Программный код и результат его выполнения онлайн [8].

Как видно из рис. 4, выполнены операции над множествами (представленные на рис. 2-3 в виде диаграмм Эйлера-Венна), при этом операция симметричной разности также выполнена корректно, используя пересечение для двух несимметричных разностей.

Выводы

1. Систематизированы основные ключевые слова LINQ, используемые для выполнения запросов к БД, а также операции над множествами, применяемые к результатам запросов на основе LINQ (табл. 1–2).

2. Сформулированы и схематично проиллюстрированы (с помощью

диаграмм Эйлера-Венна) исходные данные и ожидаемые результаты работы основных операций над множествами (рис. 1–3).

3. Представлена программная реализация обработки результатов запроса с помощью операций над множествами на примере обработки двух одномерных строковых массивов в онлайн-режиме (рис. 4).

СПИСОК ЛИТЕРАТУРЫ

1. Language Integrated Query (LINQ) C#. Microsoft Docs [Электронный ресурс]. – URL: https://docs.microsoft.com/en-us/dotnet/csharp/linq/ (дата обращения: 30.05.2020).

2. C# и .NET. LINQ – METANIT.COM [Электронный ресурс]. – URL: https://metanit.com/sharp/tutorial/15.1.php (дата обращения: 30.05.2020).

3. Troelsen A. Pro C# 5.0 and the .NET 4.5 Framework [Текст] / A. Troelsen // 6th Ed. – New York: APress, 2012. – 1488 p.

4. System.Linq Namespace. Microsoft Docs [Электронный ресурс]. – URL: https://docs.microsoft.com/en-us/dotnet/api/system.linq (дата обращения: 30.05.2020)

5. Новиков Ф.А. Дискретная математика для программистов [Текст] / Ф.А. Новиков. – СПб: Питер, 2000. – 304 с.

6. Корн Г. Справочник по математике (для научных работников и инженеров) [Текст] / Г. Корн, Т.Корн // под. общей ред. И.Г. Арамановича. – 2е изд. – М.: Изд-во «Наука», 1973. – 832 с.

7. System.Collections.Generic. Microsoft Docs [Электронный ресурс]. – URL: https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic (дата обращения: 30.05.2020).

8. C# Online Compiler. dotNet Fiddle. [Электронный ресурс]. – URL: https://dotnetfiddle.net/GYONHE (дата обращения: 30.05.2020).