Министерство образования Республики Беларусь

Учреждение образования
Белорусский государственный университет
информатики и радиоэлектроники

УДК 004.652.43 + 004.414.23

Сафонов
Анатолий Анатольевич

Реактивное и функциональное программирование для обработки данных

АВТОРЕФЕРАТ

на соискание степени магистра технических
по специальности 1-40 80 04 «Математическое моделирование, численные методы и комплексы программ»

Научный руководитель
Ганжа Виктор Александрович
кандидат физико-математических наук, доцент

Минск 2016

ВВЕДЕНИЕ

В последние годы требования к приложениям значительно изменились. С ростом объёмов информации, требуются новые способы их обработки. Пользователям требуется быстрый доступ к новой информации и возможность получения данных в реальном времени. Это стимулирует разработчиков создавать отзывчивые интерфейсы и модели для обработки. Десятки серверов, время отклика в несколько секунд, оффлайновое обслуживание, которое могло длиться часами, гигабайты данных — такими были большие приложения несколько лет назад. Сегодня же приложения работают абсолютно на всём, начиная с простых мобильных телефонов и заканчивая кластерами из тысячи процессоров. Пользователи ожидают миллисекундного времени отклика и стопроцентного аптайма, в то время как данные выросли до петабайтов.

Новые требования требуют новых решений. Раньше делался акцент на повышение мощности аппаратной части системы. Масштабирование достигалось засчёт покупки более производительных серверов и использования многопоточности. Для добавления новых серверов приходилось применять комплексные, неэффективные и дорогие проприетарные решения.

Однако прогресс не стоит на месте. Архитектура приложений изменяется в соответствии с новыми требованиями. Новые архитектуру позволяют разработчикам создавать событийно-ориентированные, масштабируемые, отказоустойчивые и отзывчивые приложения — приложения, работающие в реальном времени и обеспечивающие хорошее время реакции, основанные на масштабируемом и отказоустойчивом стеке и которые легко развернуть на многоядерных и облачных архитектурах. Эти особенности критически важны для реактивности.

Стремясь к упрощению разработки сложных систем, было создано множество парадигм программирования, каждая из созданных парадигм занимает свою нишу. Все парадигмы разделяются на две большие группы:

– императивные;
– декларативные.

Также существует множество парадигм, которые имеют черты как императивных так и декларативных систем. Одна из них — Functional Reactive Programming (FRP). Она объединяет черты функционального и реактивного программирования.

Функциональное программирование предполагает обходиться вычислением результатов функций от исходных данных и результатов других функций, и не предполагает явного хранения состояния программы. Как известно, функциональный подход к программированию имеет свою специфику: в нём

мы преобразовываем данные, а не меняем их. Соответственно, не предполагает оно и изменяемость этого состояния. Но это накладывает свои ограничения, например, при создании программ активно взаимодействующих с пользователем. В императивном языке намного проще реализовать такое поведение, ведь мы можем реагировать на какие-либо события «в реальном времени», в то время как в чистых функциональных языках нам придётся откладывать общение с системой до самого конца. Реактивное программирование ориентируется на потоки данных и распространение изменений. Функциональное реактивное программирование объединяет принципы функционального программирования и реактивного программирование:

– преобразование данных;
– распространение изменений.

Приложения, разработанные на основе этой архитектуры, называются реактивными приложениями. Словарь Merriam Webster даёт определение реактивному как «готовому реагировать на внешние события», что означает что компоненты всё время активны и всегда готовы получать сообщения. Это определение раскрывает суть реактивных приложений, фокусируясь на системах, которые:

– реагируют на события;
– реагируют на повышение нагрузки;
– реагируют на сбои;
– реагируют на пользователей.

Каждая из этих характеристик существенна для реактивного приложения. Все они зависят друг от друга, но не как ярусы стандартной многоуровневой архитектуры. Напротив, они описывают свойства, применимые на всём стеке технологий.

В данной работе исследуются и реализуются некоторые механизмы реляционной алгебры, реализованные в соответствии с концепциями реактивного и функционального программирования.

В результате была разработана библиотека классов для платформы .NET, написанная на языках программирования C# и F#, пригодная для решения практических задач в коммерческих проектах. На данный момент в библиотеке реализованы структуры хранения данных и операции реляционной алгебры.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Основной целью работы является разработка модели обработки данных с помощью операций реляционной алгебры, методов реактивного и функционального программирования, анализ и тестирование получившейся модели, разработка инструмента для платформы .NET для использования в реальных проектах.

Разработанное программное обеспечение представляет из себя библиотеку для платформы .NET. Библиотека поддерживает следующие возможности реляционной алгебры: объединение, пересечение, вычитание, декартово произведение, выборка, проекция, соединение. Также в рамках проекта были разработаны дополнительные операции, часто используемые на практике: сортировка, группирование по ключу, поиск минимального и максимального значения, поиск уникальных элементов и другие.

Для реализации операций реляционной алгебры, использовалась событийная модель, а непосредственная работа с данными производится только с помощью функций. Проект содержит структуры данных, которые генерируют события изменения данных для каждой модификации исходного источника данных. В реализации библиотеки используется модель Reactive Extensions, разработанная в Microsoft Research и имеющая реализации на многих платформах(Java, Python, Javascript и другие), и стандартные структуры данных. Поэтому данную библиотеку можно портировать на другие платформы.

При разработке платформы предполагалось, что пользователи данной библиотеки знакомы с LINQ и Reactive Extensions. Поэтому реализованные операторы имеют схожий синтаксис и семантику с операторами LINQ.

Для тестирования полученной функциональности были написаны модульные тесты, покрывающие реализованные структуры данных и операции. Тестовые данные генерировались с помощью библиотеки FsCheck, которая позволяет генерировать простые структуры данных и писать генераторы для собственных. FsCheck — реализован на F#, что упростило тестирование реализованной модели. Созданные модульные тесты в рамках работы, являются практическими примерами использования написанной библиотеки.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В первой главе произведён обзор предметной области задач, решаемых в рамках данной работы; рассмотрены вопросы о сущности фунционального и реактивного программирования и принципе их работы; приведены причины использования данных подходов для решения поставленной задачи.

Во второй главе произведено обоснование выбора использованных для реализации технологий. Произведен обзор платформы .NET Framework. Рассказывается о нововведениях в последнюю версию языка C#. Рассматриваются базовые конструкции модели Reactive Extensions.

Третья глава посвящена постановке задачи и конкретизации границ работы. Предствлен план реализации проекта.

В четвёртой главе представлены центральные объекты системы. Обределены базовые сущности и способы расширения функциональности созданной модели. В главе приведены схемы, отражающие взаимоотношения компонентов спроектированной системы.

В пятой главе описаны реализованные операторы и функции реляционной алгебры. Приведено точное описание каждого метода. Описана связь с стандартными операторами реляционной алгебры и представлены примеры использования их на практике.

В шестой главе произведено описание процесса тестирования созданной библиотеки. Показан возможный способ применения данной модели в реальных проектах с иллюстрациями. Рассмотрен пример использования на основе паттерна проектирования Model-View-ViewModel.

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены вопросы использования функционального и реактивного программирования для обработки данных. Также было разработано подмножество операций реляционной алгебры для динамических данных. Была разработана библиотека типов и функций в качестве реализации данной алгебры на платформе .NET. Разработанная библиотека функций может быть использована при разработке коммерческих продуктов на платформе Microsoft .NET для реализации взаимодействий между моделями, сервисами, для построенения отзывчивых пользовательских интерфейсов, созданных для получения актуальной информации. Единственными наиболее близким по функциональности продуктом со схожей областью применения для платформы Microsoft .NET является библиотека Dynamic Data, разработанная Roland Pheasant. Отличие данной работы, что операции и функции могут использовать не только события изменения коллекций, но и изменения самих элементов.

В целом разработанная библиотека включает требуемую функциональность, необходимую для практического использования. В библиотеке присутствуют функции для решения ряда задач связанных с применением реляционной алгебры: объединение, пересечение, вычитание, декартово произведение, выборка, проекция, соединение. Помимо этого реализованы функции агрегаторы и операции сортировки.

В результате цель работы была достигнута. Было создано программное обеспечение решающие задачи, связанных с применением релационной алгебры и событийной модели на практике. Было разработан алгоритм и проверена его работоспособность на реальных данных. За рамками проделанной работы остались некоторые специфические вопросы, например, оптимизация выполнение данных операций, реализация более специфичных функций для работы с данными. Эти вопросы возникают не во всех практических задачах, но при необходимости разработанная библиотека может быть доработана. Эти задачи также являются нетривиальными и требуют детального изучения и проработки, они не рассматривались в данной работе из-за временных ограничений на их исследвание.

В дальнейшем планируется развивать и довести существующее ПО до полноценной библиотеки, способной решать более широкий класс задач, возникающих в области построения отзывчивых пользовательских интерфейсов.

СПИСОК ОПУБЛИКОВАННЫХ РАБОТ

[1 - А.] Сафонов А. А. Событийно-ориентированная модель обработки наборов данных / А. А. Сафонов // 52-я научно-техническая конференция аспирантов, магистрантов и студентов БГУИР: Тезисы доклада - Минск, 2016.