РЕШЕНИЕ ДЛЯ ЭФФЕКТИВНОГО ХРАНЕНИЯ БОЛЬШИХ ОБЪЕМОВ ДАННЫХ

Лагута Алексей Сергеевич

магистрант, Белорусский государственный университет информатики и радиоэлектроники, РБ, г. Минск

I. Обоснование актуальности проблемы

По данным на 2019 год количество пользователей сети Интернет превысило 4 миллиарда человек [1]. Все они пользуются различными веб-сервисами. В связи с этим возрастает значимость такой характеристики сервисов как высокая производительность. Данная характеристика показывает, насколько эффективно может работать сервис при большом количестве клиентов, подключенных к нему одновременно. Из этого следует, что важной задачей при реализации любого сервиса для работы с данными является хранение информации. Именно эффективное хранение данных, а также быстрый доступ к ним позволяют увеличить пропускную способность и время отклика сервера, что существенным образом отражается на его производительности. Пример структуры такого сервиса представлен на рисунке 1.

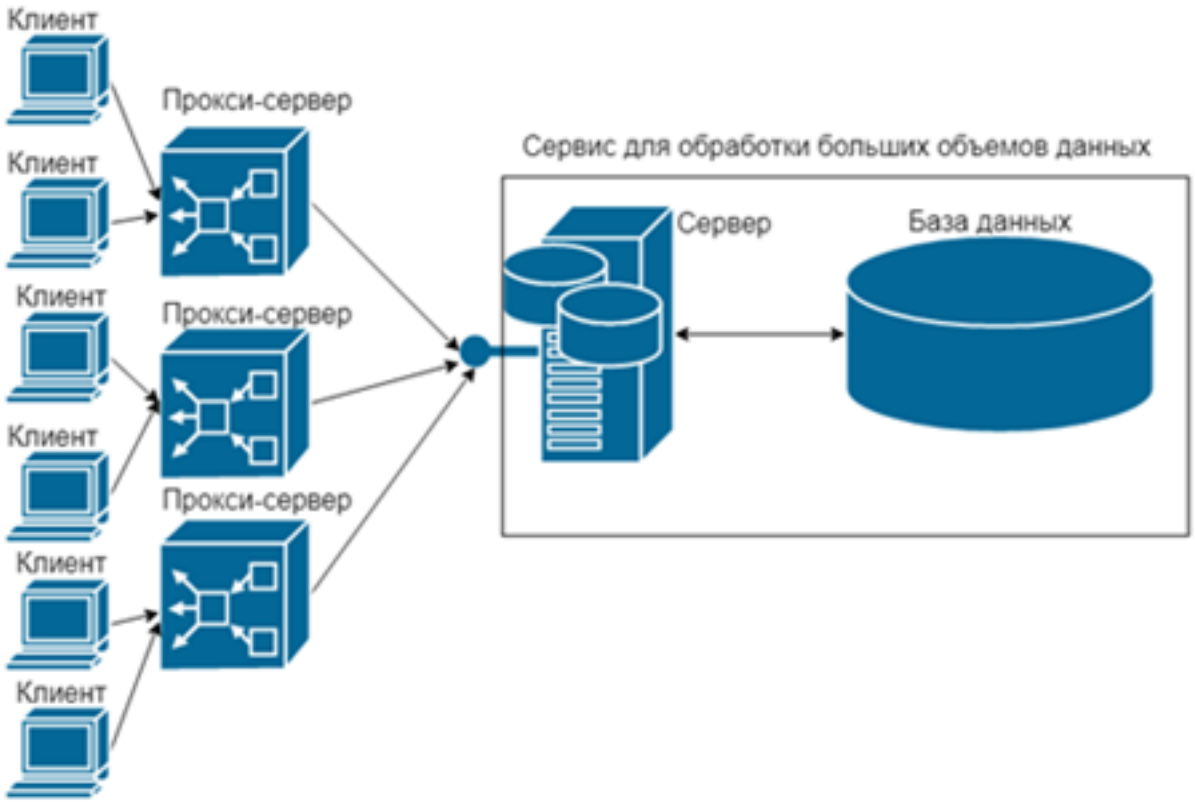

Рисунок 1. Пример структуры веб-сервиса

Существует большое количество различных способов хранения информации. Одним из самых эффективных является система хранения с помощью баз данных. Среди основных преимуществ можно выделить следующие: уменьшение избыточности данных, их целостность, согласованность и безопасность, а также резервное копирование и восстановление [2]. Вместе с тем, существует огромное количество различных баз данных. Можно выделить их основные типы: реляционные (к примеру, MySQL, PostgreSQL, Oracle и другие), NoSQL, которые в свою очередь подразделяются на документные, колоночные, графовые, типа ключ-значение и др. Зачастую тяжело выбрать подходящую базу данных в начале разработки сервиса среди такого многообразия, именно поэтому иногда приходится менять базу данных на более поздних этапах разработки. Среди основных причин, по которым может понадобиться изменение типа или сущности базы данных, можно выделить следующие:

1. Неправильное определение необходимого типа базы данных на ранних этапах разработки.

2. Изменение типов данных, которые необходимо хранить.

3. Необходимость различных типов баз данных для тестирования и развертывания.

4. При разработке унифицированного сервиса – на стадии разработки неизвестно, как именно данные будут храниться.

Из этого следует, что замена базы данных в структуре сервиса может происходить довольно часто. Кроме того, могут быть случаи, когда необходимо поддерживать сразу несколько типов баз данных одновременно. В зависимости от стадии, на которой находится разработка сервиса, изменение базы данных может стать довольно затратным материально, а также и по времени, необходимому для переписывания данной логики сервиса. В связи с этим далее будет рассмотрено решение данной актуальной проблемы.

II. Решение для эффективного хранения больших объемов данных

Платформа .NET Core является стремительно развивающейся на протяжении последних лет. В 2020 году вышла новая версия платформы .NET Core 5.0 [3]. Она предлагает несколько технологий для управления базами данных – NHibernate, Entity Framework, ADO.NET. Последняя из них является наиболее производительной, хоть и не поддерживает такого широкого функционала, как Entity Framework и NHibernate [4-6]. Однако, функциональности ADO.NET достаточно, чтобы покрыть все необходимые действия с базой данных.

Идея решения состоит в том, чтобы иметь один интерфейс для работы с базой данных, вне зависимости от её типа. В таком случае, переход от одной базы данных к другой не будет трудоемким – необходимо изменить только строку подключения к ней и запросы, если изменилась её структура. Поставщик данных .NET абстрагирует взаимодействие базы данных с приложением и, следовательно, упрощает разработку приложения.

ADO.NET предоставляет такой интерфейс – а именно классы DbProviderFactory и DbConnection [7-8]. С помощью данных классов можно только при помощи строки подключения получать доступ к различным типам базы данных, а потом с помощью унифицированных команд совершать операции над данными, такие как чтение из базы данных и запись в неё. Список провайдеров (фактически различных сущностей базы данных), поддерживаемых технологией ADO.NET, велик [9]. Среди них можно выделить MySQL, SQLite, SQL Server, PostgreSQL, Oracle и другие. Для работы с большими объемами данных есть поставщик облачной службы базы данных Azure Cosmos DB. Таким образом, платформа .NET предоставляет широкий спектр инструментов, которые позволяют эффективно хранить данные. Решение, описанное в данной статье, позволяет менять хранилища без особых затрат по времени, а также использовать различные хранилища одновременно, меняя лишь строку подключения к ним.

Список литературы:

1. Internet usage worldwide – statistics & facts [Электронный ресурс] / Statista. – Нью-Йорк, 2020. – Режим доступа: https://www.statista.com/topics/1145/internet-usage-worldwide/ (дата обращения: 20.03.21)

2. Advantages of Database Management System [Электронный ресурс] / Tutorials Point. – Хайдарабад, 2018. – Режим доступа: https://www.tutorialspoint.com/Advantages-of-Database-Management-System (дата обращения: 20.03.21)

3. Breaking changes in .NET 5 [Электронный ресурс] / Microsoft. – Вашингтон, 2020. – Режим доступа: https://docs.microsoft.com/en-us/dotnet/core/compatibility/5.0 (дата обращения: 21.03.21)

4. ADO.NET [Электронный ресурс] / Microsoft. – Вашингтон, 2019. – Режим доступа: https://docs.microsoft.com/enus/dotnet/framework/data/adonet. (дата доступа: 21.03.2021)

5. Entity Framework [Электронный ресурс] / Microsoft. – Вашингтон, 2019. – Режим доступа: https://docs.microsoft.com/en-us/ef/. (дата доступа: 21.03.2021)

6. NHibernate [Электронный ресурс] / NHibernate. – Нью-Йорк, 2019. – Режим доступа: https://nhibernate.info. (дата доступа: 21.03.2021)

7. DbProviderFactories / Microsoft. – Вашингтон, 2017. – Режим доступа: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/dbproviderfactories. (дата доступа: 21.03.2021)

8. DbConnection, DbCommand and DbException / Microsoft. – Вашингтон, 2017. – Режим доступа: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/dbconnection-dbcommand-and-dbexception. (дата доступа: 21.03.2021)

9. Билл Хэмильтон, Мэтью МакДональд ADO.NET in a Nutshell / Билл Хэмильтон, Мэтью МакДональд // Издательство: O’Reilly, 2013. – 703 стр. – С. 15–56.