УДК 004

Сеняшина Ксения Павловна,
студент,
4 курс, факультет «Информатика и вычислительная техника»
Казанский государственный энергетический университет
Россия, г. Казань
Научный руководитель: Хуснутдинов Рамиль Миннегаязович,
кандидат физико-математических наук, доцент
Казанский государственный энергетический университет
Россия, г. Казань

ИСПОЛЬЗОВАНИЕ ПАТТЕРНА UNIT OF WORK ДЛЯ РАБОТЫ С ИЗМЕНЕНИЯМИ БАЗЫ ДАННЫХ

Аннотация: *Статья посвящена исследованию архитектурного паттерна Unit of Work, применяемого для управления изменениями данных и транзакциями в корпоративных информационных системах. Рассматриваются основные принципы работы паттерна, его структура и место среди других архитектурных решений. Анализируются особенности реализации Unit of Work в современных ORM-фреймворках. Приводятся примеры практического применения, а также рассматриваются преимущества и ограничения использования данного подхода.*

Ключевые слова: *архитектурный паттерн, Unit of Work, транзакции, ORM, база данных.*

Annotation: *This article examines the architectural pattern Unit of Work, which is used for managing data changes and transactions in enterprise information systems. The core principles of the pattern, its structure, and its place among other architectural solutions are considered. The specifics of implementing Unit of Work in modern ORM frameworks are analyzed. Examples of practical*

application are provided, along with a discussion of the advantages and limitations of using this approach.

Key words: *architectural pattern, Unit of Work, transactions, ORM, database.*

Современные информационные системы предъявляют высокие требования к надежности, согласованности и эффективности работы с данными. Для корпоративных приложений, включая банковские и торговые системы, критически важно обеспечить корректную обработку транзакций и целостность данных. В связи с этим архитектурные паттерны, предназначенные для управления изменениями данных, играют ключевую роль при проектировании программных систем.

Паттерн Unit of Work занимает особое место среди методов организации доступа к данным и описывается как архитектурное решение, предназначенное для группировки изменений в объектной модели и их последующего сохранения в базе данных в рамках одной логической транзакции [1]. Использование данного паттерна позволяет централизовать управление состоянием данных и упростить реализацию транзакционной логики в объектно-ориентированных приложениях.

1. Транзакции и управление целостностью данных

Транзакции – это логическая единица работы с базой данных, состоящая из операций, которые выполняются полностью или не выполняются вовсе. Главная идея – обеспечить целостность данных даже при сбоях, ошибках или сбое системы.

Транзакция обладает четырьмя основными свойствами, известными как ACID [1]:

– Атомарность (Atomicity) – все операции в транзакции либо выполняются полностью, либо не выполняются вовсе.

– Согласованность (Consistency) – транзакция переводит базу данных из одного допустимого состояния в другое.

– Изолированность (Isolation) – транзакции не влияют друг на друга, изменения внутри транзакции не видны другим до её завершения.

– Долговечность (Durability) – после фиксации транзакции её изменения сохраняются даже при сбое системы.

Паттерн Unit of Work применяется для практической реализации транзакционного подхода в объектно-ориентированных приложениях, позволяя рассматривать набор изменений данных как единую логическую единицу работы и выполнять их согласованную фиксацию или откат.

2. Основные идеи и концепции паттерна Unit of Work

Концепция паттерна Unit of Work была предложена для управления изменениями в объектной модели и их согласованного сохранения в базе данных. Его архитектурные принципы и место среди других паттернов корпоративных приложений подробно рассмотрены в работе М. Фоулера [1]

Структура паттерна включает несколько основных компонентов, каждый из которых выполняет определенную роль в управлении изменениями данных и транзакциями [1]. Для наглядного понимания данный паттерн можно рассматривать как «буфер изменений», в котором накапливаются все операции над объектами до момента их единовременного применения к базе данных. К таким компонентам относятся репозитории, модель изменений, контекст (или сессия) и транзакционный менеджер:

– Репозитории представляют собой интерфейсы или классы, предназначенные для работы с конкретными сущностями предметной области. Они служат посредниками между бизнес-логикой и базой данных, предоставляя абстракцию для выполнения операций создания, чтения, обновления и удаления (CRUD). Использование репозиториев позволяет

изолировать бизнес-логику от деталей реализации доступа к данным. Пример интерфейса репозитория приведен в листинге 1.

Листинг 1. Интерфейс репозитория

```
public interface IRepository<T> where T : class
{
    void Add(T entity);
    void Remove(T entity);
    T FindById(int id);
    IEnumerable<T> GetAll();
}
```

– Модель изменений используется внутри Unit of Work для регистрации объектов, состояние которых было изменено в ходе выполнения бизнес-операций. В рамках данных моделей отслеживаются новые, измененные и удаленные сущности, которые подлежат последующей обработке при фиксации транзакции.

– Контекст или сессия объединяет все зарегистрированные изменения и управляет их применением к базе данных. В современных ORM-фреймворках контекст отвечает за отслеживание состояния объектов и координацию операций сохранения данных. В частности, в Entity Framework данные функции реализованы в классе DbContext [3].

– Транзакционный менеджер обеспечивает управление жизненным циклом транзакции, включая её открытие, фиксацию и откат при возникновении ошибок. В большинстве ORM-фреймворков данный функционал инкапсулирован внутри контекста или сессии, что соответствует концепции Unit of Work, описанной в архитектурных источниках [1].

Пример интерфейса Unit of Work, инкапсулирующего контекст и транзакционный механизм, приведён в листинге 2.

Листинг 2. Интерфейс Unit of Work.

```
public interface IUnitOfWork : IDisposable
{
    IRepository<Entity> Entities { get; }
    void Commit();
    void Rollback();
}
```

3. Рабочий процесс

Рабочий процесс при использовании паттерна Unit of Work заключается в последовательном управлении изменениями данных в рамках одной транзакции:

1. Создаётся экземпляр Unit of Work, связанный с контекстом данных.

2. В ходе выполнения бизнес-логики регистрируются операции добавления, изменения и удаления сущностей.

3. Выполняется фиксация изменений. С точки зрения разработчика это означает, что вся бизнес-логика может быть выполнена без немедленного обращения к базе данных, а фактическое сохранение результатов происходит один раз – в заранее определённой точке выполнения.

4. В случае успешного выполнения всех операций изменения сохраняются в базе данных, а при возникновении ошибок выполняется откат транзакции.

Схема принципа работы паттерна Unit of Work(UoW) представлена на рис. 1.

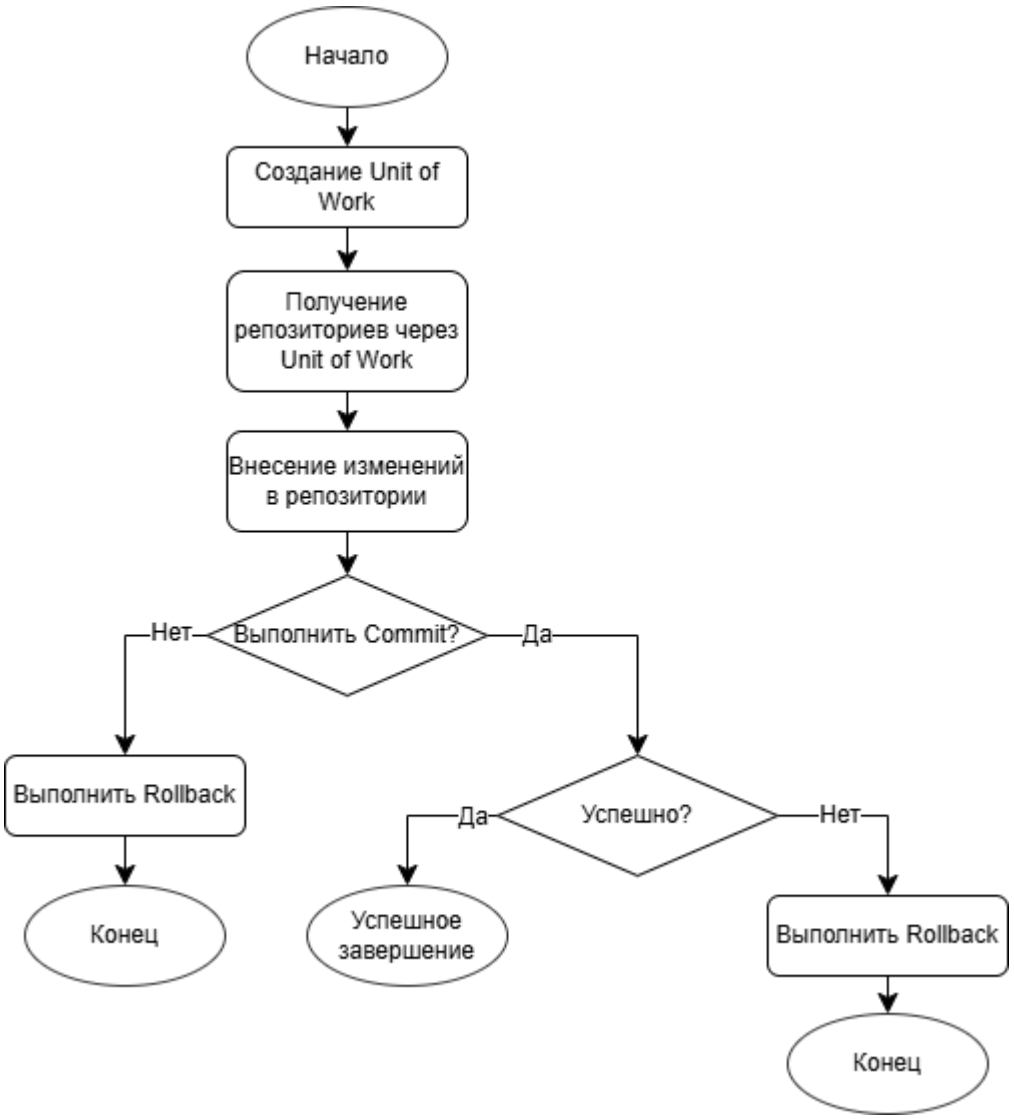

Рисунок 1. Блок - схема принципа работы паттерна Unit of Work

4. Реализация паттерна в различных ORM

В ORM-фреймворке Entity Framework паттерн Unit of Work реализован на уровне класса DbContext, который автоматически отслеживает изменения сущностей и сохраняет их в базе данных при вызове метода SaveChanges [3]. Пример использования Entity Framework приведён в листинге 3.

Листинг 3. Пример работы Entity Framework.

```
using(var context = new AppDbContext())
{
    context.Users.Add(new User());
    context.Orders.Add(new Order());
    context.SaveChanges(); // атомарное завершение
}
```

В Hibernate управление изменениями данных осуществляется через объект Session, связанный с транзакцией. Фиксация и откат транзакции выполняются с использованием объекта Transaction, что соответствует архитектурной модели Unit of Work [1]. Пример использования Hibernate приведён в листинге 4.

Листинг 4. Пример работы Hibernate(Java).

```
Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();
try {
    // операции
    tx.commit();
} catch (Exception e) {
    tx.rollback();
} finally {
    session.close();
}
```

5. Практическое применение, преимущества и ограничения

При внедрении паттерна Unit of Work на практике необходимо учитывать ряд технических особенностей, включая отслеживание состояния объектов, управление кэшем и механизмами отложенной загрузки, а также корректную обработку ошибок с гарантированным откатом транзакций. Кроме того, Unit of Work часто используется совместно с другими архитектурными паттернами, такими как Repository и Service Layer, что рекомендуется в практических руководствах по разработке корпоративных приложений [2].

Паттерн Unit of Work широко применяется в прикладных системах, где требуется согласованная работа с несколькими сущностями. Примерами являются заказные системы, в которых необходимо обеспечить согласованное обновление таблиц Order, OrderItems и Inventory, а также банковские приложения, выполняющие атомарные переводы средств между счетами.

Преимущества и ограничения использования паттерна Unit of Work целесообразно представить в табличной форме (см. таблицу 1).

Таблица 1.

Преимущества и ограничения использования паттерна UoF

Преимущества	Ограничения
Высокая атомарность операций и соблюдение ACID-свойств	При длительном времени жизни контекста возможно повышенное потребление памяти
Предотвращение ошибок при выполнении множества взаимосвязанных операций	Сложность реализации в распределённых и микросервисных системах
Упрощение управления изменениями состояния объектов	Требует тщательного архитектурного проектирования
Оптимизация производительности за счёт группировки операций в одной транзакции	Возможные сложности масштабирования при неправильной конфигурации

Заключение

Паттерн Unit of Work является важным архитектурным инструментом при проектировании систем, требующих надёжного управления транзакциями и согласованности данных. Его применение позволяет централизовать управление изменениями, повысить атомарность операций и упростить взаимодействие с базой данных. Использование Unit of Work в сочетании с современными ORM-фреймворками способствует созданию масштабируемых и сопровождаемых программных систем.

Список литературы:

1. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2003. – [Электронный ресурс] – URL: https://martinfowler.com/eaaCatalog/unitOfWork.html

2. Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application. Microsoft Docs, 2022. [Электронный ресурс] – URL: https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application

3. Working with DbContext. Microsoft Docs, 2020. [Электронный ресурс] – URL: https://learn.microsoft.com/en-us/ef/ef6/fundamentals/working-with-dbcontext