

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

*Факультет*  
вычислительной техники

*Кафедра*  
МОиПЭВМ

Направление подготовки 09.04.02 «Информационные системы и технологии»  
Профиль Проектирование, разработка и эксплуатация информационных систем

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ  
РАБОТА МАГИСТРА**

на тему

**Моделирование баз данных на основе объектно-реляционного  
отображения**

Студент \_\_\_\_\_ Вагапов Рустам Ильдусович  
(подпись, дата) (ФИО полностью)

Руководитель \_\_\_\_\_ Шибанов С. В.  
(подпись, дата) (фамилия, инициалы)

Нормоконтролер \_\_\_\_\_ Такташкин Д.В.  
(подпись, дата) (фамилия, инициалы)

Рецензент \_\_\_\_\_ Васин Л. А.  
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от \_\_\_\_\_ № \_\_\_\_\_)

Заведующий кафедрой \_\_\_\_\_ Макарычев П.П.  
(подпись) (фамилия, инициалы)

Работа защищена с отметкой \_\_\_\_\_ (протокол заседания ГЭК от \_\_\_\_\_ № \_\_\_\_\_)

Секретарь ГЭК \_\_\_\_\_ Попова Н.А.  
(подпись) (фамилия, инициалы)

Пенза, 2017

## Содержание

Введение.....	4
1 Проблематика проведения объектно-реляционного отображения.....	8
1.1 Методики проектирования баз данных.....	8
1.2 Обзор и оценка стратегий объектно-реляционного отображения .....	12
1.2.1 Элементы объектно-ориентированной и реляционной модели данных, участвующие в отображении .....	13
1.2.2 Стратегии и шаблоны объектно-реляционного отображения .....	14
1.2.3 Применение шаблонов объектно-реляционного отображения .....	18
1.3 Обзор программных и инструментальных средств объектно-реляционного отображения.....	20
1.3.1 Классификация средств объектно-реляционного отображения.....	20
1.3.2 Сравнительный обзор ORM библиотек .....	21
1.3.3 Сравнительный обзор визуальных средств объектно-реляционного отображения.....	26
1.4 Требования к средствам объектно-реляционного отображения .....	28
Выводы .....	29
2 Модельные представления и операции отображения схем баз данных .....	30
2.1 Модельное представление объектно-ориентированных схем баз данных .....	30
2.2 Модельное представление реляционных схем баз данных .....	36
2.3 Операции отображения объектно-ориентированных схем в реляционные ..	38
2.4 Обобщенный алгоритм отображения объектно-ориентированных схем баз данных в реляционные.....	40
Выводы .....	51
3 Программная реализация и экспериментальная оценка эффективности разработанных алгоритмов .....	52
3.1 Назначение и основные возможности программных средств объектно- реляционного отображения.....	52
3.2 Архитектура программных средств объектно-реляционного отображения .	57

3.3 Спецификации программных средств объектно-реляционного отображения.....	60
3.4 Экспериментальная оценка применения программных средств объектно-реляционного отображения.....	67
Выводы .....	74
Заключение .....	75
Список использованных источников .....	77

## Введение

**Актуальность темы работы.** В последние десятилетия мир информационных технологий активно развивается, почти с каждым годом совершенствуется аппаратная составляющая информационных систем (ИС). Также немало изменений произошло в области программирования, усложняются задачи, которые ставятся перед разработчиками, появляются новые технологии и методологии разработки. Важной частью ИС является централизованная база данных, которая тоже должна отвечать заданным требованиям и в результате чего претерпела немало изменений.

Но, пожалуй, в число важнейших событий, произошедших в сфере информационных технологий, входят появление объектно-ориентированного подхода в программировании и появление реляционных баз данных.

Объектно-ориентированный подход дает возможность сформулировать более сложные и объемные задачи программирования, позволяет больше сосредоточиться на предметной области. Также использование данного подхода дает следующие преимущества:

- структурированность программы в целом и ее элементов;
- простота проектирования;
- возможность интеграции в рамках одного информационного массива разнородных данных и функций их обработки;
- механизм событий и исключительных ситуаций.

Благодаря всем вышеперечисленным преимуществам объектно-ориентированный подход стремительно проникает во многие сферы прикладного программирования и проектирования.

Если в программировании популярность набрал объектно-ориентированный подход, в мире СУБД сложилась иная картина. В наше время доминирующую позицию на рынке баз данных заняли реляционные СУБД. За долгое время своего существования реляционные базы данных набрали большую популярность и стали промышленным стандартом в своей области, во многом благодаря относительной простоте проектирования данных и наличию

четкой математической теории в этой области. Также одной из серьезных причин успеха реляционных баз данных является поддержка ими наиболее стандартизованного языка коммуникаций с базой данных SQL.

Сложившаяся ситуация накладывает некоторые неудобства на разработку крупных информационных систем с единой базой данных. Проектирование приложения и базы данных происходят отдельно и на основе разнородных моделей, в результате чего возникают проблемы с поддержкой диалога между объектно-ориентированным приложением и реляционной базой данных. Кроме того, программный интерфейс реляционных БД, прежде всего, разрабатывался для структурного подхода программирования и не ориентирован под особенности объектно-ориентированного подхода. В результате образовывается объектно-реляционный импеданс и актуальным остается вопрос его эффективного разрешения.

**Цель и задачи работы.** Целью данной работы является исследование различных подходов и методов объектно-реляционного отображения, выявление границ их применения и разработка программных средств, выполняющих эффективное объектно-реляционное отображение.

Были выдвинуты следующие задачи, которые планируется решить для достижения цели работы:

- исследование проблемы проведения объектно-реляционного отображения, исследование и оценка различных подходов проектирования базы данных, стратегий отображения и программных средств-аналогов;
- исследование и создание формализованного модельного представления схем БД а также выявление операций и алгоритмов отображения элементов схем;
- разработка программных средств, позволяющих выполнить эффективное отображение объектно-ориентированной схемы БД в реляционную.

**Методы исследования.** При выполнении работы применялись следующие методы исследования:

- математический аппарат теории множеств;

– методы объектно-ориентированного анализа и проектирования программного обеспечения.

**Объектом исследования** в данной работе являются методики проектирования баз данных и соответствующих программных средств.

**Предметом исследования** работы являются методы проектирования баз данных с применением объектно-реляционного отображения.

**Достоверность научных результатов** подтверждена теоретическими вкладами и результатами тестирования, а также сравнением полученных результатов с результатами, приведенными в научной литературе.

**Научная новизна** работы заключается в следующем:

- разработана методика проведения эффективного отображения объектно-ориентированных схем баз данных в реляционные схемы баз данных;
- предложено решение проблемы объектно-реляционного импеданса, возникающего при взаимодействии объектно-ориентированных приложений с реляционными базами данных.

**Основные положения, выносимые на защиту:**

- методика проведения эффективного отображения объектно-ориентированных схем баз данных в реляционные схемы баз данных.

**Практическая и теоретическая значимость работы.** Полученные результаты применимы в теоретических и практических работах по моделированию баз данных и по разработке программных средств, реализующих объектно-реляционное отображение. Основываясь на результатах исследования, можно организовать эффективное взаимодействие объектно-ориентированных программных средств с реляционными базами данных. Рассматриваемый в работе подход проектирования базы данных позволяет на основе единой, платформонезависимой, объектно-ориентированной модели предметной области создавать как программные средства, так и соответствующие базы данных для информационных систем.

**Апробация работы.** Основные положения и результаты диссертационной работы докладывались на конференциях:

- «Информационные технологии в науке и образовании. Проблемы и перспективы» в городе Пенза, ПГУ, 2016;
- «Новые информационные технологии и системы» в городе Пенза, ПГУ, 2016;
- «Информационные технологии в науке и образовании. Проблемы и перспективы» в городе Пенза, ПГУ, 2017.

**Структура и объем работы.** Работа состоит из введения, трех тематических глав, заключения и списка литературы. Общий объем основного текста – 76 страниц, включая 27 рисунков и 24 таблицы. Список литературы изложен на 2 страницах и содержит 19 наименований.

## **1 Проблематика проведения объектно-реляционного отображения**

Сегодня во многих сферах деятельности можно наблюдать стремительное развитие информационных систем (ИС) и технологий. ИС становятся все более сложными, более требовательными к ресурсам, наряду с аппаратной составляющей, усложняется и программное обеспечение (ПО). Важной частью ИС является централизованная база данных, которая тоже должна отвечать заданным требованиям.

Процесс разработки многих автоматизированных систем включает этапы, связанные с моделированием и проектированием как самого ПО, так и базы данных. Модели определяют основные составляющие системы на некотором уровне абстракции, и служат основой для взаимодействия участников проекта, особенно на ранних стадиях разработки. Качество модели во многом зависит также и от выбранного подхода к моделированию [1].

Логическая модель ПО и модель базы данных, так или иначе, затрагивает понятия и отношения предметной области ИС. На сегодняшний день есть множество методов создания модели ПО и модели базы данных. Но часто при проектировании базы данных и ПО используются различные подходы к созданию логической модели.

### **1.1 Методики проектирования баз данных**

В основе проектирования базы данных лежит определенный подход и методология. Они определяют основные этапы проектирования, модели для представления предметной области и способы их преобразования в структуры самой базы данных. В этой работе будет рассмотрен довольно распространенный классический подход проектирования с использованием нотации сущность-связь для представления логической структуры предметной области, также в качестве альтернативы будет рассмотрен подход с использованием объектной семантической модели с применением объектно-реляционного отображения (ОРО) в рамках подхода архитектура, управляемая моделью (Model Driven Architecture, MDA).



Классический подход предполагает использование широко известного способа представления предметной области для реляционных баз данных - модель «сущность-связь» (ER-модель). Данная модель была предложена Питером Ченом в 1976 году, с тех пор она развивалась и стала поддерживаться множеством CASE-средств [2].

Так как нет единого стандарта ER-модели, сейчас можно встретить несколько его вариаций: нотация Баркера, IDEF1x и др. Они уже отличаются от классической диаграммы сущность-связь. Но, не смотря на все различия различных нотаций, основными компонентами любой ER-модели являются сущности, атрибуты и связи между сущностями [3]. За время своего существования данная ER-модель максимально приблизилась к реляционной модели данных, это можно заметить в ее реализации в популярных CASE средствах в виде нотации IDEF1x. Модель относительно проста для понимания, однако предназначена для спецификации только логической структуры данных, их статической составляющей, не затрагивая операций с данными.

Современную ER-диаграмму довольно легко преобразовать в схему реляционной БД, так как для этой цели она и создается. Сущностям сопоставляются таблицы, атрибутам столбцы таблиц, элементы сущности представляют собой строки таблицы, а связи реализуются с помощью миграции внешних ключей, для реализации связи типа «многие ко многим» создается вспомогательная таблица пересечения.

Проблемы и некоторые трудности при моделировании БД при помощи модели сущность-связь в основном связаны с тем, что данная модель акцентирует внимание на сущностях предметной области и связях между ними, не затрагивая операций с данными. Помимо этого типы атрибутов должны быть простыми, не являться массивами и коллекциями простых типов.

Таким образом, модель «сущность-связь» предназначена для описания логической структуры данных, показывает взаимосвязи между элементами данных, легко преобразуется в реляционную БД. Но современные приложения, которые взаимодействуют с данной БД, оперируют элементами объектно-

ориентированной модели, в результате чего получается объектно-реляционный импеданс.

Чтобы избежать данной проблемы предлагается рассмотреть объектно-ориентированную модель как способ представления данных о предметной области.

С развитием объектно-ориентированного подхода, с увеличением сложности разрабатываемых проектов и других факторов появилась потребность в использовании шаблонов проектирования и архитектурных подходов.

Одним из этих подходов является архитектура, управляемая моделью - Model Driven Architecture (MDA). В ее основе лежит идея построения компонентов ПО на основе независимого от платформы модели, ее детализации и последующего перехода к платформозависимой модели и к исходному коду [4]. Данный подход выбран из-за того, что он не привязывает разработчиков к конкретной платформе, предполагает использование единой модели предметной области и автогенерацию части исходного кода. Кроме того, модель предметной области хорошо подходит и для разработки базы данных для проектируемой системы. В качестве модели представления предметной области вполне можно использовать семантическую объектно-ориентированную модель данных.

Стандартизованная объектно-ориентированная модель описана в рекомендациях стандарта ODMG (Object Database Management Group – группа управления объектно-ориентированными базами данных). Объектная модель данных, наряду со всеми возможностями модели «сущность-связь» обладает рядом существенных преимуществ, по сравнению с последней. Эти преимущества заключаются, прежде всего, в способности интегрировать в рамках единого информационного массива разнородные данные и функции их обработки [5]. Кроме того объектная модель поддерживает механизмы наследования, позволяет легко описать иерархически вложенные и агрегированные данные. Достигается это благодаря довольно большой схожести объектов с объектами реального мира, благодаря чему легко

оперировать понятиями из рассматриваемой предметной области, не вникая в понятия из области баз данных.

Основными компонентами рассматриваемой модели данных являются понятия объекта, класса объектов и связей между ними. Они составляют основу, на которой строится данная модель. В отличие от сущностей, атрибуты объектов логически хранятся именно внутри этих объектов, а не в наборах связанных объектов.

Как и в модели «сущность-связь» между объектами можно установить связи. Связи бывают нескольких типов, это связи агрегирования, композиции, которые указывают на то, что домен атрибута одного класса, по сути, является другим классом, и отношения наследования. В модели «Сущность-связь» это отношения типов один ко многим и многие ко многим, причем последний тип в объектной модели реализуется без проблем и не требует введения промежуточного класса пересечения.

Хоть объектно-ориентированная модель данных изначально поддерживается объектно-ориентированными СУБД (ООСУБД), в данной работе они не рассматриваются, так как такие СУБД не нашли столь широкого применения во многом из-за уже большой популярности реляционных СУБД на рынке. Кроме того у ООСУБД нет такого мощного языка запросов как SQL, из-за чего возникают некоторые неудобства при обработке данных.

Исходя из концепции MDA, на роль первоначальной платформонезависимой модели хорошо подходит объектно-ориентированная модель данных, так как она позволяет наиболее естественным образом описать структуру и поведение реальных объектов предметной области, при этом, не касаясь ее реализации в БД. Учитывая основные преимущества и распространенность реляционных СУБД, на роль платформозависимой модели подходит физическая схема реляционной базы данных. Преобразование моделей осуществляется с помощью процесса объектно-реляционного отображения.

Таким образом, рассматриваемый в работе подход за счет использования единой модели устраняет объектно-реляционный импеданс,

сокращает время разработки проектов, может улучшить качество разработки при этом сохранить преимущества использования реляционных баз данных.

## **1.2 Обзор и оценка стратегий объектно-реляционного отображения**

Задача отображения объектно-ориентированной модели в реляционную схему базы данных имеет множество решений и одной объектной модели предметной области могут соответствовать различные схемы реляционной базы данных, результат прежде всего зависит от выбора подхода и стратегии отображения.

Можно выделить схемо-зависимый и схемо-независимый подходы отображения.

Схемо-независимый подход ориентирован на использование единой реляционной схемы для представления произвольных объектно-ориентированных моделей данных. Такой подход предпочтителен для приложений, работающих с несколькими меняющимися моделями предметных областей. Единая неменяющаяся схема БД хорошо поддается администрированию и сопровождению. К недостаткам данного подхода можно отнести необходимость ведения словарей метаданных и вследствие чего несколько меньшую производительность, чем при использовании схемо-зависимого подхода.

Схемо-зависимый подход больше ориентирован на приложения, работающие только с одной моделью данных. При таком подходе схемы БД будут различаться в зависимости от того, какую предметную область они отражают. Это позволяет построить более эффективную и производительную схему, но при отображении сложных предметных областей могут получиться еще более сложные схемы БД, что негативно сказывается на поддержке и администрировании такой базы.

Также выделяют подход, при котором применяется схемо-зависимое отображение и ведутся словари метаданных. Это позволяет использовать информацию из словарей метаданных для эффективной реализации сложных запросов.

### 1.2.1 Элементы объектно-ориентированной и реляционной модели данных, участвующие в процессе отображения

В соответствии с выбранным подходом, происходит отображение следующих видов элементов объектно-ориентированной модели:

- простой атрибут базового типа;
- перечисление;
- атрибут-массив;
- атрибут объектного типа;
- коллекция объектов;
- иерархия наследования.

Данные элементы объектно-ориентированной модели могут быть отображены в следующие элементы реляционной модели данных:

- столбец базового типа;
- столбец бинарного типа;
- таблица базы данных;
- связь типа «один к одному»;
- связь типа «один ко многим»;
- связь типа «многие ко многим».

В таблице 1 показано, в какие элементы реляционной модели данных (РМД) могут быть отображены элементы объектно-ориентированной модели данных (ООМД).

Таблица 1 – Варианты отображения элементов ООМД

Элементы ООМД	Элементы РМД
Простой атрибут базового типа	Столбец базового типа
	Столбец бинарного типа
Перечисление	Столбец базового типа
Атрибут-массив	Таблица
	Столбец бинарного типа
Атрибут объектного типа	Связь типа «один к одному»

Продолжение таблицы 1

Элементы ООМД	Элементы РМД
Коллекция объектов	Связь типа «один ко многим»
	Связь типа «многие ко многим»
Иерархия наследования	Таблица
	Группа таблиц

Таким образом, были перечислены все основные элементы объектно-ориентированной и реляционной модели данных.

### 1.2.2 Стратегии и шаблоны объектно-реляционного отображения

Для отображения элементов объектно-ориентированной модели применяются следующие шаблоны (паттерны) отображения:

- прямое отображение базовых типов;
- отображение перечислений;
- отображение в бинарный поток;
- таблица для каждого типа массива;
- таблица для конкретного атрибута-массива;
- отображение «один к одному»;
- отображение «один ко многим»;
- включение в состав другой таблицы;
- отображение «многие ко многим»;
- одна таблица на иерархию наследования;
- горизонтальное отображение;
- вертикальное отображение.

Прямое отображение ставит в соответствие базовым типам атрибутов класса аналогичные типы столбцов таблиц в целевой базе данных. Количество поддерживаемых типов в данном случае ограничено возможностями СУБД, поэтому данный паттерн подходит только для отображения базовых типов данных, таких как: числа, строки, дата, время и т.д.

Отображение перечислений можно применять для атрибутов с ограниченным доменом, например, пол у человека, тип кузова у автомобиля. При таком подходе возможным значениям атрибута ставятся в соответствие их сокращения, которые будут храниться в БД. Это позволяет уменьшить объем памяти, выделяемый для хранения столбцов таблиц.

Отображение в бинарный поток можно применять для объемных, сложно-структурированных атрибутов или для атрибутов-массивов. Данный атрибут преобразуется в бинарный поток и в таком виде хранится в соответствующем столбце таблицы БД. Но это неудобно в тех случаях, когда требуется доступ к отдельным элементам хранимого атрибута, так как независимо от структуры данный атрибут хранится в атомарном виде со всеми вытекающими отсюда последствиями. Если в запросах необходимо обращаться к внутренним элементам атрибутов-массивов, то они отображаются с помощью следующих паттернов:

- таблица для каждого типа массива;
- таблица для конкретного атрибута-массива;

Паттерн «Таблица для каждого типа» предполагает создание по одной таблицы для каждого элементарного типа массива. В данной таблице присутствуют такие столбцы как идентификатор объекта, класса, атрибута, номер элемента в массиве и само значение элемента. Данный паттерн позволяет сохранить в БД массивы любых базовых типов данных. Однако если хранить большое количество объектов, содержащих большие по объему данных массивы, то таблицы этих массивов могут достигать внушительных размеров. Из-за этого снижается производительность запросов к элементам массива.

Для достижения большей производительности запросов можно использовать паттерн «Таблица для конкретного атрибута-массива». В данном случае атрибут преобразуется в отдельную таблицу, где будут храниться только его элементы. В таблице будут присутствовать такие столбцы как идентификатор объекта, класса, номер элемента в массиве и само значение элемента. Но если в классах отображаемой модели присутствует большое количество атрибутов-массивов, то каждая из них будет отображаться в

отдельную таблицу, что может заметно усложнить восприятие и администрирование такой схемы БД [7].

Наиболее оптимальным вариантом является комбинирование этих двух подходов, учитывая то, что для больших по объему данных атрибутов рекомендуется использовать отдельные таблицы, а все мелкие атрибуты можно хранить в таблице типа.

Если класс является составным, то есть, имеет в составе атрибутов объекты или коллекции объектов, то для их отображения можно использовать следующие паттерны:

- отображение с миграцией внешнего ключа;
- включение в состав главной таблицы;
- отображение «многие ко многим».

Отображение с миграцией внешнего ключа предполагает замену ссылки на объект его идентификатором в таблице БД, то есть реализуется связь типа «один к одному» в случае объектного атрибута и «один ко многим» в случае коллекции объектов.

При наличии между классами связи типа «один к одному» все же следует прибегнуть к паттерну «Включение в состав главной таблицы». При таком подходе в таблице к атрибутам основного класса добавляются все атрибуты вложенного класса. Этот паттерн также применяется если между классами прослеживается связь агрегация, при котором объект одного класса существует только в составе других классов.

Отображение «многие ко многим» применяется для отображения коллекций объектов в тех классах, между которыми можно установить связь ассоциации с кардинальностью «многие ко многим», когда оба класса содержат ссылки друг на друга. В результате в БД кроме таблиц классов создается еще таблица пересечения, необходимая для реализации связи «многие ко многим».

Часто при проектировании классов объектно-ориентированной модели прибегают к механизму наследования, в результате чего группы классов



объединяются в иерархии наследования. Для их отображения можно применить следующие паттерны:

- одна таблица на иерархию наследования;
- горизонтальное отображение;
- вертикальное отображение.

Паттерн «Одна таблица на иерархию наследования» предполагает создание единой таблицы, которая будет содержать атрибуты всех классов иерархии, в таблицу также добавляется столбец-дискриминатор, содержащий идентификаторы классов добавляемых в него объектов. Такой способ отображения обеспечивает максимальную производительность работы приложений и простоту реализации, так как все хранится в одной таблице. Но из-за того, что в строке таблицы заполнены только те столбцы, которые соответствуют атрибутам сохраняемого объекта, то при отображении классов с большим количеством уникальных атрибутов в результирующей таблице создается большое количество пустых значений, для которых выделяется память.

При горизонтальном отображении для каждого класса создается отдельная таблица, в которой сохраняются все его уникальные и унаследованные атрибуты. В результате отпадает проблема неиспользуемого пространства, сохраняется эффективность и простота реализации запросов по манипулированию данными. Однако так как создается дублирование атрибутов в таблицах, то при изменении атрибутов класса-родителя, необходимо произвести такие же изменения в классах-потомках, также затруднено изменение класса-родителя в иерархии или ее удаление.

Вертикальное отображение также предполагает создание отдельной таблицы для каждого класса иерархии, но здесь таблица класса хранит только его уникальные атрибуты, а все унаследованные хранятся в таблицах классов-предков. Использование данного паттерна исключает дублирование атрибутов, из-за чего модификация таблицы класса не затрагивает таблицы унаследованных классов. Но при такой организации схемы чтобы получить

доступ ко всем атрибутам класса, необходимо обращаться ко всем таблицам родительских классов. Это порождает сложные запросы для манипулирования данными и чем глубже класс находится в иерархии, тем медленнее будут выполняться запросы к его объектам.

### 1.2.3 Применение шаблонов объектно-реляционного отображения

В таблице 2 показано, какие паттерны применяются для отображения различных элементов объектно-ориентированной модели и какие в результате получаются элементы реляционной модели данных.

Таблица 2 – Результаты паттернов отображения

№	Элемент объектно-ориентированной модели данных	Паттерн отображения	Элемент реляционной модели данных
1	Простой атрибут	Прямое отображение	Столбец базового типа
		Отображение в бинарный поток	Столбец бинарного типа
2	Перечисление	Отображение перечислений	Столбец базового типа
		Прямое отображение	
3	Атрибут-массив	Отображение в бинарный поток	Столбец бинарного типа
		Таблица для каждого типа массива (Hierarchy aggregate)	Таблица
		Таблица для конкретного атрибута-массива	

Продолжение таблицы 2

4	Объектный атрибут	Включение в состав главной таблицы	Набор столбцов таблицы
		Отображение «один к одному»	Связь типа «один к одному»
5	Коллекция объектов	Отображение «один ко многим»	Связь типа «один ко многим»
		Отображение «многие ко многим»	Связь типа «многие ко многим»
6	Иерархия наследования	Одна таблица на иерархию наследования (Filtered mapping)	Таблица
		Горизонтальное отображение	Группа таблиц
		Вертикальное отображение	

Как видно из таблицы 2, один и тот же элемент объектной модели часто можно отобразить с помощью нескольких паттернов.

В результате исследования стратегий и подходов отображения можно прийти к выводу, что все они находят свое применение. Однако среди них нет универсальных, которые могли бы в равной степени сочетать производительность, гибкость запросов, оптимизацию ресурсов памяти. Прежде всего, выбор стратегий зависит от особенностей самой прикладной модели данных, от требований к конечной информационной системе, от решаемых системой задач и т.д. Выбор оптимальной стратегии отображения позволит максимально эффективно организовать работу информационной системы с базой данных.

### **1.3 Обзор программных и инструментальных средств объектно-реляционного отображения**

В данное время существуют различные инструментальные средства (ORM), способные отображать таблицы, связи и другие элементы реляционной модели на объектную модель и наоборот. Они чаще всего представляют собой библиотеки, которые программист может использовать, чтобы обращаться в своем программном коде к реляционной БД, как к объектной базе. Существуют также программы, которые позволяют генерировать классы и конфигурационные файлы для таких библиотек.

#### **1.3.1 Классификация средств объектно-реляционного отображения**

Можно выделить следующие виды средств ОРО:

- ORM-библиотеки;
- Визуальные CASE-средства.

ORM библиотеки предоставляют программные интерфейсы для сохранения в реляционной БД бизнес-объектов приложения и позволяют работать с ними как с объектами. ORM привязаны к конкретному языку программирования и часто к одной платформе, представляют собой библиотеку или фреймворк, который содержит набор классов и интерфейсов, реализующий функционал ORM.

Визуальные средства ОРО могут поставляться как дополнения к существующим ORM библиотекам либо как самостоятельные продукты. Эти средства облагают графическим пользовательским интерфейсом и не всегда привязаны к платформе и языку программирования. Эти средства позволяют генерировать БД из объектно-ориентированной модели данных и наоборот, генерировать программный код классов приложения на основе структуры реляционной БД.

#### **1.3.2 Сравнительный обзор ORM библиотек**

В таблице 3 приведен список рассматриваемых в данной работе ORM-библиотек.

Таблица 3 – Список рассматриваемых ORM-библиотек

Название	Платформа, сфера применения	Поддерживаемые СУБД	Текущая версия	Поддержка
Hibernate NHibernate	Java .NET	MSSQL, Oracle, Microsoft Access, Firebird, PostgreSQL, DB2 UDB, MySQL, SQLite	5.2.10 апрель 2017 4.1.1	Есть
Entity Framework	.NET	MSSQL, IBM DB2, Sybase SqlAnywhere, Oracle, любая, если есть ADO.NET Data Provider	Entity Framework 6 октябрь 2013	Есть
Linq to SQL	.NET	MS SQL Server 2000 и выше		Нет
ORMLite	Java, Android	MySQL, MS SQL Server, Postgres, SQLite, DB2, Oracle	ORMLite 5.0 июль 2016	Есть
GreenDAO	Android	SQLite	GreenDAO 3	Есть
Active Record	Web	MSSQL, Oracle, Microsoft Access, Firebird, PostgreSQL, DB2 UDB, MySQL, SQLite		Есть

Hibernate – это бесплатная Java библиотека с открытым исходным кодом, представляющая собой инструмент объектно-реляционного отображения. Его основной задачей является преобразование данных реляционной БД в объектно-ориентированные модели и обратно. NHibernate реализован как .NET-аналог библиотеки Hibernate, он и будет рассматриваться в данном обзоре.

Установка в Visual Studio производится при помощи менеджера пакетов NuGet. Параметры отображения классов приложения в таблицы БД можно описать 3 способами: при помощи xml файлов, через аннотации (в дополненной версии) и с помощью создания классов отображения, соответствующих классам сущностей. Для сохранения и извлечения классов-сущностей из БД используется класс Session, который содержит соответствующие методы [8].

NHibernate позволяет отобразить в БД абсолютно любой класс приложения, поддерживает основные стратегии отображения и механизмы управления транзакциями, показывает относительно высокую производительность, является настраиваемой и расширяемой библиотекой [9]. Nhibernate поддерживает работу с такими СУБД как MSSQL, Oracle, Microsoft Access, Firebird, PostgreSQL, DB2 UDB, MySQL, SQLite.

Но, не смотря на настраиваемость библиотеки, Hibernate не поддерживает автоматическое отображение классов, для отображения требуется вручную установить соответствие между элементами объектной и реляционной модели. Библиотека является относительно сложной для освоения ввиду большого количества свойств и настроек.

Entity Framework – это средство объектно-реляционного отображения для платформы .NET от компании Microsoft. Представляет собой набор библиотек и инструментов с графическим интерфейсом для более удобной работы с данной технологией. Установка в Visual Studio производится при помощи менеджера пакетов NuGet. В данной фреймворке реализованы три разных способа создания классов и базы данных, это - Database first, Model first и Code first. Database first предполагает генерацию классов приложения на основе уже существующей базы данных, Model first предполагает создание графической модели базы данных, по которой создается сама база данных или генерируются классы приложения. Последний подход Code first позволяет генерировать базу данных уже на основе кода классов приложения. Работа с хранимыми классами осуществляется с помощью класса-контекста, где имеются необходимые методы для извлечения и сохранения изменений в БД [10].

Важным преимуществом Entity Framework является возможность автогенерации БД по программному коду классов, классов программы по таблицам БД и графической модели, классы не нуждаются в аннотировании для отображения их в БД. Entity Framework не привязан к конкретному списку СУБД, и поддерживает работу с любыми из них, для которых имеется ADO.NET Data Provider. Однако данная технология поддерживает не все

основные стратегии отображения, привязана к одной платформе и отсутствует адекватная поддержка хранимых процедур и представлений БД.

LINQ to SQL – продукт Microsoft, представляет собой средство объектно-реляционного отображения, которое позволяет составлять запросу к БД в удобной форме с помощью операторов LINQ, которые затем трансформируются в sql-выражения. Для использования LINQ to SQL не требуется подключать его к проекту, так как само средство встроено в платформу .NET версии 3.5 и выше. Для сопоставления с таблицами БД классы дополняются аннотациями. Также есть возможность сгенерировать классы по БД с помощью Object Relational Designer. Для взаимодействия с БД используется класс DataContext, который составляет SQL запросы по Linq запросам [11].

Данная технология позволяет производить основные операции с данными хранимых классов, относительно проста в освоении и работе. Однако официальная версия LINQ to SQL предназначена для работы только с СУБД MS SQL Server. Данное средство поддерживает относительно небольшое количество стратегий отображения и показывает наименьшую производительность из рассматриваемых средств ОРО для .NET (рисунок 1).

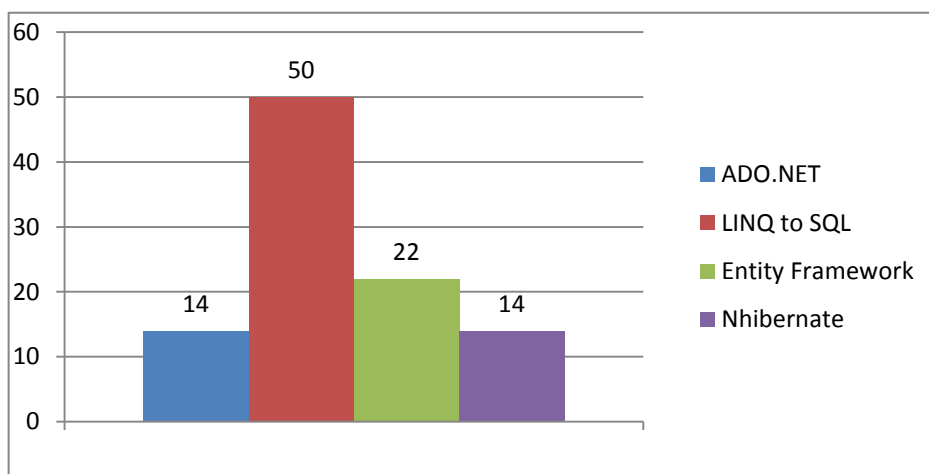


Рисунок 1 – Относительное время выполнения запросов среди средств ОРО для платформы .NET

OrmLite представляет собой популярный ORM-фреймворк с открытым исходным кодом на языке Java, который также можно применять в Android разработке. Фреймворк поставляется в виде java-библиотеки, которую можно

добавить в проект. Для отображения классов используются аннотации, которые содержат необходимые данные по сопоставлению этих классов с таблицами БД. Для каждого хранимого класса создается DAO класс, который содержит методы для сохранения и извлечения данных хранимого класса. Данная библиотека относительно проста в освоении и работе и показывает неплохую производительность [12].

Поддерживаются такие СУБД как MySQL, MS SQL Server, Postgres, SQLite, DB2, Oracle. В данной библиотеке можно выделить такие недостатки как поддержка малого количества стратегий отображения, отсутствует автогенерация БД из классов.

GreenDAO – ORM фреймворк, ориентированный исключительно под Android разработку. Подключить библиотеку можно добавив ее в Gradle. Так же как и в OrmLite, в GreenDAO используются аннотирование классов и DAO объекты для взаимодействия с хранимыми классами.

Библиотека показывает лучшую производительность среди рассматриваемых ORM для Android (рисунок 2) и также позволяет генерировать DAO классы на основе классов-сущностей. Так как продукт ориентирован исключительно на Android разработку, обеспечена поддержка только SQLite баз данных. В библиотеке также нет поддержки основных стратегий отображения и автогенерации базы данных.

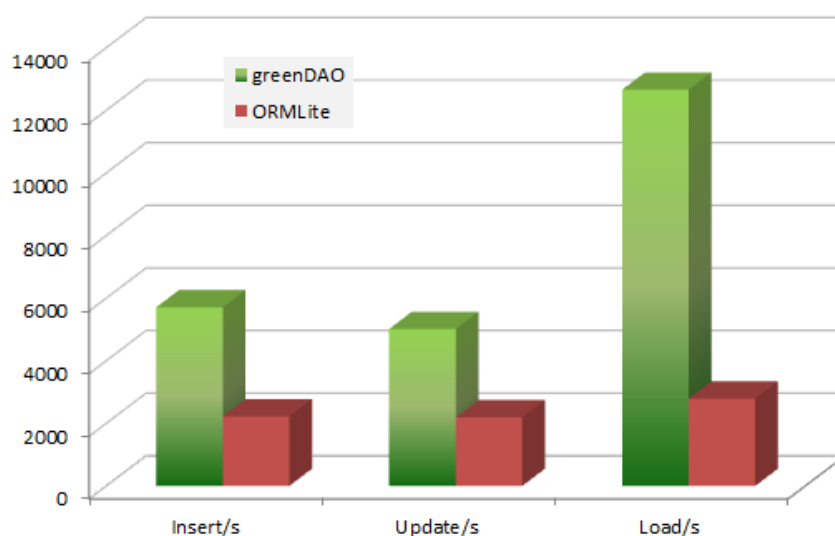


Рисунок 2 – Сравнение производительности ORM для Android разработки



Active Record представляет собой шаблон для взаимодействия приложений с БД, который применяется в web-разработке и соответствующих фреймворках ( Kohana, yii, Ruby on Rails). При таком подходе классы-сущности наследуются от базового Active Record класса и приобретают методы для сохранения, извлечения и других операций с БД. Данный шаблон хорошо подходит для тех случаев, когда приложение использует несложные запросы к базе данных. С усложнением бизнес-логики приложения начинают возникать проблемы по составлению более сложных запросов [13].

В таблице 4 приведены сведения о поддержке стратегий отображения различными средствами ОРО.

Таблица 4 – Поддерживаемые стратегии объектно-реляционного отображения

	Nhibernate	Entity Framework	LINQ to SQL	ORMLite	Green DAO	Active Record
Прямое отображение базовых типов	+	+	+	+	+	+
Таблица для каждого типа массива	–	–	–	–	–	–
Таблица для конкретного атрибута-массива	–	–	–	–	–	–
Отображение объектного атрибута	+	+	+	+	+	+
Коллекции объектов «один ко многим»	+	+	+	+	+	+
Коллекции объектов «многие ко многим»	+	+	–	–	+	+
Включение в состав главной таблицы (встроенные классы)	–	+	–	–	–	–
Таблица на иерархию классов	+	+	+	–	–	+
Горизонтальное отображение	+	+	–	–	–	–
Вертикальное отображение	+	+	–	–	–	–

Из таблицы 4 видно, что далеко не все рассматриваемые ORM-библиотеки поддерживают основные стратегии отображения.

### 1.3.1 Сравнительный обзор визуальных средств объектно-реляционного отображения

В данном обзоре представлены следующие визуальные средства ОРО:

- DeKlarit;
- MyGeneration;
- ObjectMapper.

В таблице 5 приведен список рассматриваемых в данной работе визуальных CASE-средств ОРО.

Таблица 5 – Список рассматриваемых визуальных средств ОРО

Название	Платформа, сфера применения	Тип лицензии	Поддерживаемые СУБД	Последняя версия
DeKlarit	.NET интеграция в Visual Studio	Коммерческая, есть пробная версия	Microsoft SQL Server, Oracle, DB/2, Access, MySQL	4.3
MyGeneration	.NET	Свободно распространяемая	MSSQL, Oracle, Microsoft Access, Firebird, PostgreSQL, DB2 UDB, MySQL, SQLite	1.3
ObjectMapper	.NET	Свободно распространяемая	Microsoft SQL Server	2.3.3903.0

DeKlarit представляет собой расширение для Visual Studio, которое позволяет разработчику создавать бизнес-объекты(аналоги классов), которые затем преобразуются в структуры БД. Данное средство позволяет описывать не только хранимые, но и вычисляемые атрибуты в виде несложных формул, позволяет проводить анализ бизнес-объектов и, при необходимости, их нормализацию. DeKlarit поддерживает языки C# и Visual Basic а также такие СУБД как: Microsoft SQL Server, Oracle, DB/2, Access, MySQL, PostgreSQL, MySQL [14].

Однако сами бизнес-объекты не являются аналогом классов приложения, так как для них отсутствует поддержка механизма наследования и атрибутов-коллекций. Само средство напоминает расширение CASE для создания

диаграмм «сущность связь». Кроме того DeKlarit имеет коммерческую лицензию и предназначен для интеграции в Visual Studio, отсюда следует ее привязка к платформе .NET.

MyGeneration – свободно распространяемое средство генерации программного кода классов на основе структуры БД. Данное средство имеет набор шаблонов, которые можно применять для генерации классов и конфигурационных файлов NHibernate, хранимых процедур для Microsoft SQL Server и для других технологий. Данные шаблоны можно редактировать, изменяя запрашиваемые при генерации приложением параметры или состав генерируемого кода. MyGeneration поддерживает работу со следующими СУБД: MSSQL, Oracle, Microsoft Access, Firebird, PostgreSQL, DB2 UDB, MySQL, SQLite [15].

MyGeneration хорошо справляется с генерацией кода приложения, однако данная программа не способна непосредственно отобразить объектно-ориентированную модель в структуру реляционной БД.

ObjectMapper – свободно распространяемое визуальное средство ОРО, предназначенное для работы с библиотекой NHibernate. Средство анализирует структуру БД и отображает ее в виде графической модели, на основании которой возможна генерация программного кода с классами доступа к таблицам БД и XML-файлов в которых описано само отображение. Есть поддержка работы с СУБД Microsoft SQL Server. Но средство ориентировано именно на .NET платформу и кроме того, как и MyGeneration не реализует прямое отображение элементов объектно-ориентированной модели в элементы реляционной модели.

Сравнительный обзор существующих продуктов ОРО показал, что среди основных ORM средств нет такого, который бы в равной степени сочетал в себе такие характеристики как: производительность, простота освоения и использования, настраиваемость, поддержка основных стратегий отображения и возможности автогенерации, а значит есть смысл рассматривать альтернативные варианты.

## 1.4 Требования к средствам объектно-реляционного отображения

Исходя из результатов обзора существующих средств ОРО, выделим следующие требования к средству ОРО, для того, чтобы оно обеспечивало эффективное и гибкое отображение объектно-ориентированной модели в модель реляционную:

- поддержка основных стратегий и подходов объектно-реляционного отображения;
- независимость от платформы и языка разработки;
- предоставление возможности пользователю влиять на ход отображения (настраиваемость отображения).

Каждое средство объектно-реляционного отображения поддерживает определенный для него список стратегии и шаблонов отображения. Чем больше список поддерживаемых шаблонов отображения, тем средство более универсальное и, как правило, более настраиваемое.

Как отмечалось в разделе 1.3, средства объектно-реляционного отображения подразделяются на ORM-библиотеки и визуальные CASE-средства. Часто определенные ORM-библиотеки привязаны к конкретной платформе и только к одному языку программирования. Они предназначены для внедрения их в проект разрабатываемого приложения. Визуальные средства объектно-реляционного отображения не так жестко привязаны к языку и платформе, из-за этого они могут использоваться более широким кругом лиц и не только разработчиками.

Настраиваемость системы часто зависит от ее функциональности, чем больше функций поддерживает система, тем более настраиваемой она может быть. Для средств объектно-реляционного отображения характерны настройки способа отображения классов в таблицы и настройки способов взаимодействия с сохраненными в базе объектами. Хотя большое количество настроек и несколько усложняет процесс освоения средства ОРО, но зато

предоставляет возможность организовывать более эффективное взаимодействие приложения с базой данных.

## **Выводы**

В данной главе рассматриваются предпосылки использования объектно-реляционного отображения при моделировании реляционных баз данных, а также сравниваются модели представления предметной области для проектирования баз данных и программного обеспечения. В результате сравнения с диаграммой «сущность-связь» объектно-ориентированная модель данных показала свою эффективность для описания предметной области как для программного обеспечения, так и для БД.

Сравнение основных стратегий и шаблонов объектно-реляционного отображения показало, что среди них нет универсальных, которые могли бы в равной степени сочетать производительность, гибкость запросов, оптимизацию ресурсов памяти. В результате ставится задача о выборе оптимальной стратегии отображения, которая позволит максимально эффективно организовать работу информационной системы с базой данных.

В данное время существуют средства ОРО, способные отображать таблицы, связи и другие элементы реляционной модели на объектную модель и наоборот. Производится оценка наиболее популярных средств ОРО для различных платформ.

Исследование показало, что среди рассматриваемых средств ОРО нет такого, который бы сочетал в себе все характеристики для достижения эффективного и гибкого отображения. Таким образом, есть смысл разработки средства ОРО, сочетающего в себе данные характеристики.

## **2 Модельные представления и операции отображения схем баз данных**

В рамках диссертационной работы осуществляется разработка алгоритмов и соответствующих средств ОРО, в которых основной упор ложится на обеспечение гибкого и эффективного отображения. Гибкость заключается в том, что программа может создать множество вариантов отображения для одной и той же исходной объектной модели. При этом выбирается наиболее эффективный вариант отображения, в соответствии с критериями, которые на него накладываются. Эти критерии определяются в процессе диалога с пользователем. Ему задаются вопросы об особенностях использования базы данных, о требованиях к информационной системе, о характеристиках информационной системы и т.д. На основе ответов пользователя определяются требования к производительности, к объему базы данных, к удобству администрирования, составления запросов и др.

Все это требует наличия алгоритмов для оценки и выбора подходящей стратегии отображения. Для этого требуется проанализировать как сам процесс отображения, так и модели данных, с которыми этот процесс оперирует. Для облегчения этой задачи построим математическое модельное представление для объектной модели данных и для реляционной базы данных. Модель позволяет сосредоточиться на нужных аспектах системы или объекта, отбрасывая остальные. В данном случае математическая модель служит для представления элементов объектной модели данных и схемы реляционной БД в виде математической структуры. Это нужно для того, чтобы в дальнейшем можно было выявить основные операции по отображению элементов модельного представления семантической объектной модели в элементы модельного представления реляционной базы.

### **2.1 Модельное представление объектно-ориентированных схем баз данных**

Объектно-ориентированную модель данных можно представить в виде следующей математической структуры:

$$O = \langle C, R, I, E, L, Name, Type \rangle, \quad (1)$$

где  $C$  – набор классов модели  $O$ ;

$R$  – описывает бинарные отношений между классами модели;

$I$  – содержит набор отношений прямого наследования между классами модели;

$E = E^U \cup E^D$  – множество ограничений, распространяемых на объектно-ориентированную модель данных  $O$ , включает ограничения уникальности  $E^U$  и ограничения на области значения атрибутов  $E^D$ ;

$L$  – содержит набор допустимых символов для названий элементов модели (алфавит имен);

$Name$  – содержит набор слов, составленных на алфавите  $L$ ;

$Type$  – представляет собой множество базовых типов данных.

$Type^R$  – представляет собой множество типов связей между классами;

Вышеперечисленные множества также можно записать математически и выделить их структуру.

Класс можно определить несколькими способами. В данной работе класс представляется в виде некой структуры через перечисление его характеристик, атрибутов и методов.

$c \in C$  – конкретный класс модели предметной области и он имеет следующую структуру:

$$c = \langle name, A, M \rangle,$$

где  $name \in Name$  – имя класса;

$A = A^S \cup A^C$  – множество атрибутов класса, которое включает такие подмножества как набор простых атрибутов  $A^S$  и множество атрибутов-коллекций  $A^C$ ;

$M$  – множество методов класса.

$a \in A$  – атрибут класса, который имеет следующую структуру:

$$a = \langle f, type, name \rangle$$

где  $f \in F$  – характеристика атрибута, принадлежит множеству характеристик атрибутов;

$type \in D$  – домен атрибута (тип данных), принадлежит множеству типов данных  $D = Type \cup C$ ;

$name \in Name$  – имя атрибута.

Каждый метод  $m \in M$  имеет следующую структуру:

$$m = \langle f, type, name, P \rangle,$$

где  $f \in F$  – характеристика метода;

$type \in D$  – тип данных, который возвращает метод;

$name \in Name$  – название метода;

$P$  – набор параметров метода. Элемент множества  $P$  представлен как:

$$p = \langle type, name \rangle.$$

Что касается множества связей между классами  $R$ , то каждая связь  $r \in R$  представляет собой следующую структуру:

$$r = \langle type, c_b, c_e \rangle,$$

где  $type \in Type^R$  – тип связи;

$c_b \in C$  – класс, от которого исходит связь;

$c_e \in C$  – класс, куда направлена связь.

Элемент множества  $I$  (отношение прямого наследования) представляется как  $i = \langle c_p, c_c \rangle$ , где  $c_p$  и  $c_c$  – родительский и дочерний по отношению к нему класс.

Множество ограничений  $E$  включает в себя следующие подмножества:

- ограничения уникальности  $E^U$ ;
- ограничения области значений  $E^D$ .

Ограничение уникальности  $eu \in E^U$  будет затрагивать класс и его уникальный атрибут:  $eu = \langle c, a \rangle$ , где  $c \in C, a \in A$ .

Ограничение области значений  $ed \in E^D$  ограничивает домен атрибута, к которому оно применяется, данное ограничение записывается в следующей форме:  $ed = \langle c, a, D^E \rangle$ , где  $c \in C, a \in A$ , а  $D^E \subset D$  – ограниченный домен атрибута [16].



Объектно-ориентированная модель данных представляется в виде сложной, иерархической математической структуры. На рисунке 3 показаны множества, которые фигурируют в объектно-ориентированной модели данных.

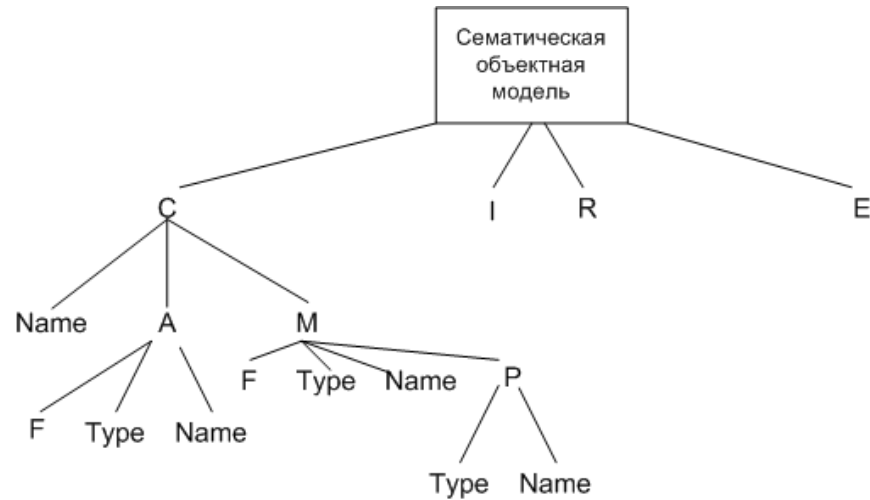


Рисунок 3 – иерархия элементов семантической объектной модели

Здесь мы видим, что класс состоит из имени, атрибутов и методов. Каждый из атрибутов состоит из имени, характеристик и типа данных, а описание метода, кроме всех перечисленных у атрибута элементов, включает в себя также и множество параметров. Также в этой системе содержатся множества различных отношений между классами и множество ограничений, распространяемых как на саму модель, так и на ее элементы.

В рамках данного математического представления были представлены основные элементы объектно-ориентированной модели, участвующие в отображении.

Простой атрибут базового типа представляется как элемент множества  $A^S$ , для него характерны такие домены как String, Integer, Float, Date, Boolean и т.д.

Атрибут перечислимого типа также является элементом множества  $A^S$ , но домен такого атрибута обычно состоит всего из нескольких значений и является подмножеством одного из доменов простых атрибутов.

Атрибут-массив представляется как элемент множества  $A^C$ , для этого атрибута характерны те же домены, что и для простых атрибутов.

Атрибут объектного типа представлен как элемент множества  $A^S$ , его доменом могут являться классы из множества  $C$ .

Коллекцию объектов можно выразить как элемент множества  $A^C$ , его доменом является класс из множества  $C$ .

Прежде чем представить иерархию наследования в данной математической модели, введем отношение наследования, которое обозначим символом  $\omega$ .

Множество атрибутов и методов класса  $c_i$  обозначено переменной

$$H_i = A_i \cup M_i.$$

Чтобы класс  $c_c$  стал наследником класса  $c_p$ , необходимо и достаточно, чтобы множество атрибутов и множество методов класса  $c_p$  являлось собственным подмножеством множества атрибутов и методов класса  $c_c$ :

$$c_c \omega c_p \Leftrightarrow H_p \subset H_c.$$

Тогда прямое наследование можно сформулировать так: если класс  $c_c$  прямо наследуется от класса  $c_p$ , то он наследуется от  $c_p$ , и не существует такого класса  $c_k$ , который наследуется от  $c_p$  и от которого наследуется  $c_c$ :

$$c_c I c_p \Rightarrow \nexists c_k, \text{ что } c_c \omega c_k \text{ и } c_k \omega c_p.$$

Таким образом, иерархия наследования  $C^I \subseteq C$  с корневым классом  $c^p$  - это такое подмножество  $C$ , которое содержит корневой элемент и все унаследованные от него классы.

$$C^I \subseteq C, c^p \in C^I, (\forall c \in C^I, c \neq c^p) c \omega c^p.$$

Также в иерархию наследования следует включить и отношения прямого наследования, которые установлены между классами из множества  $C^I$ . Таким образом, иерархию наследования можно представить в виде следующей структуры:

$$IH = \langle C^I, I^C \rangle,$$

где  $C^I$  - множество классов иерархии наследования;

$I^C$  - набор отношения наследования, которые связывают классы из множества  $C^I$ .

## 2.2 Модельное представление реляционных схем баз данных

Реляционные базы данных построены на основе реляционной модели, которая основана на достаточно «отточенной» математической теории. В качестве основных структур данных в реляционной модели (таблицы, результаты запросов) используются только  $n$ -арные отношения, над которыми могут производиться различные операции, предусмотренные реляционной алгеброй и реляционным исчислением [5]. Но современная реляционная база данных состоит не только из набора взаимосвязанных отношений. Кроме таблиц база данных часто содержит набор хранимых процедур, триггеры, схемы, ограничения и т.д.

Среди основных компонентов реляционной базы данных можно выделить следующие множества:  $T$  - таблицы,  $A$  - атрибуты,  $Type$  - типы атрибутов,  $P$  - хранимые процедуры, входные и выходные параметры хранимых процедур  $P_i$  и  $P_o$  соответственно,  $H$  - триггеры,  $R$  - связи между таблицами,  $O$  - ограничения и множество допустимых имен  $Name$ .

На рисунке 4, по аналогии с рисунком 3, изображена схема иерархии элементов реляционной БД.

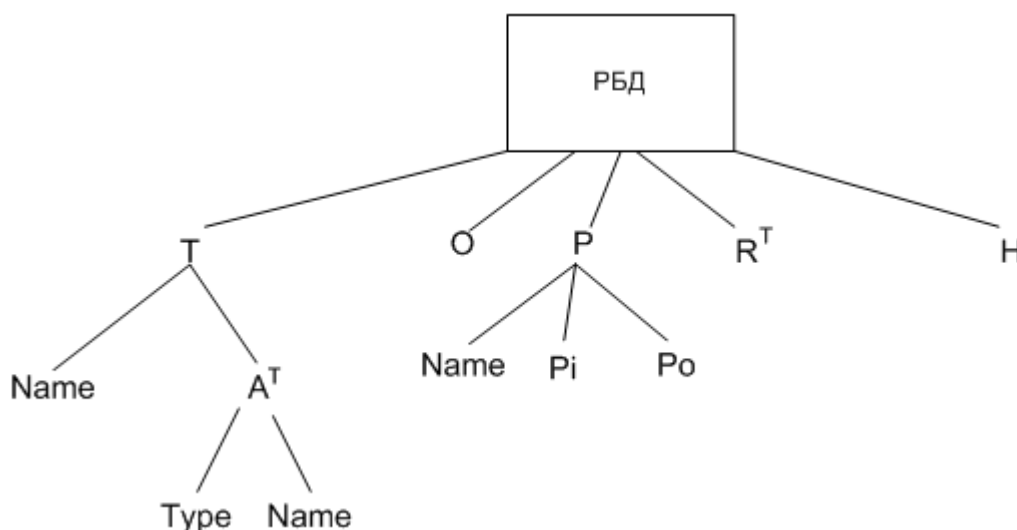


Рисунок 4 – Иерархия элементов реляционной БД

Таким образом, схему реляционной БД можно представить в виде следующей записи:

$$B = \langle T, P, R^T, H, O \rangle, \quad (2)$$

где  $T$  – представляет собой набор таблиц базы данных;

$P$  – содержит набор хранимых процедур;

$R^T$  – описывает бинарные реляционные связи между таблицами;

$H$  – содержит набор триггеров базы данных;

$O$  – множество ограничений на элементы базы.

Множество таблиц состоит из элементов  $t \in T$ , которые можно рассматривать как:

$$t = \langle name, A^T \rangle,$$

где  $name \in Name$  – название таблицы;

$A^T$  – множество атрибутов (столбцов) таблицы.

Каждый атрибут  $a \in A^T$  можно представить как:

$$a = \langle name, type \rangle,$$

где  $name \in Name$  – имя атрибута, принадлежит множеству допустимых имен;

$type \in Type$  – тип данных (домен) атрибута, принадлежит множеству доменов базы данных.

Множество ключей можно выразить с помощью элементов  $k \in K$  со следующей структурой:

$$k = \langle type, t, a \rangle,$$

где  $type \in \{PK, FK\}$  – тип ключа (первичный или внешний);

$t \in T$  – таблица, к которой относится данный ключ;

$a \in A^T$  – сам ключевой столбец таблицы.

Связи  $r \in R^T$  между таблицами можно представить как:

$$r = \langle t_1, a_1, cord_1, t_2, a_2, cord_2 \rangle,$$

где  $t_1, t_2 \in T$  – таблицы, на которые распространяется связь;

$a_1, a_2 \in A^T$  – атрибуты таблиц, на которые ссылается связь;

$cord_1, cord_2 \in \{0, 1, *\}$  – кардинальности связи для обеих таблиц, может принимать значения: 0, 1 и \* (много).

Столбец базового типа представляется элементом множества  $A$ , для него характерны такие домены как STRING, INTEGER, FLOAT, DOUBLE, DATE, BIT и т.д.

Столбец бинарного типа также представлен элементом из множества  $A$ , для таких столбцов характерны домены `VARBINARY` или `LONGVARBINARY`.

Таблица базы данных представляется элементом множества  $T$ .

Связь типа «один к одному» является элементом множества  $R$ . Кардинальность такой связи для обеих таблиц равна одному.

Связь типа «один ко многим» представляет собой элемент множества  $R$ . Кардинальность для одной таблицы равна единице, для другой равна  $*$  (много).

Связь типа «многие ко многим» также характеризуется элементом множества  $R$ . При этом кардинальности для обеих таблиц равны  $*$ .

### 2.3 Операции отображения объектно-ориентированных схем в реляционные

Посмотрев на структуру объектной модели данных и модели реляционной можно обнаружить аналоги множеств одной модели в другой. Они похожи и по содержанию и по названию, и между ними можно провести соответствие.

Запишем отображения множеств из математического представления объектной модели во множества математического представления РБД, для этого выделим следующие семейства отображений:

- отображения классов;
- отображения атрибутов и связей между классами.

К семейству отображений классов можно отнести следующие отображения:

- отображение одного класса, обозначим его как  $st: C \rightarrow T$ , здесь элемент множества классов отображается в элемент множества таблиц БД;
- отображение иерархии наследования, в общем случае его можно обозначить как  $ihm: C^I, I^C \rightarrow T, R^T$ , здесь все классы, входящие в иерархию, отображаются в одну или несколько таблиц.

При этом, можно выделить 3 альтернативных вида отображения наследования:

- $fm: C^I \rightarrow T$  – отображение с фильтрацией, при котором вся иерархия отображается в одну таблицу БД;
- вертикальное и горизонтальное отображение, при котором каждый класс отображается в отдельную таблицу ( $sim: C^I \rightarrow T$ ), а отношение наследования в реляционную связь ( $im: I^C \rightarrow R^T$ ).

К семейству отображения атрибутов и связей можно отнести следующие отображения:

- $dm: A^S \rightarrow A^T$  – прямое отображение атрибута в столбец таблицы;
- $am: A^C \rightarrow T$  – отображение атрибута-массива в отдельную таблицу базы данных;
- $om: A^S \rightarrow R^T$  – отображение объектного атрибута в реляционную связь типа «один к одному»;
- $bm: A^S \rightarrow A^T$  – отображение объектного атрибута в набор столбцов таблицы;
- $cam: A^C \rightarrow R^T$  – отображение коллекции объектов в реляционную связь типа «один ко многим»;
- $A^C \rightarrow T, A^C \rightarrow R^T$  – отображение коллекции объектов в связь типа «многие ко многим».

Также можно выделить следующие виды отображений:

- $mm: M \rightarrow P$  – отображение методов классов в хранимые процедуры базы данных;
- $edm: E^D \rightarrow H$  – отображение ограничений на область значений атрибута в триггеры БД;
- $eum: E^U \rightarrow K$  – отображение ограничений на уникальность атрибутов во множество ключей БД.

В каждом выражении слева находится множество из объектной модели, а справа множество из РБД. Если при отображении одного множества получаются несколько, то они указываются через запятую.

Такое модельное представление позволяет выделить основные структурные элементы объектной модели и схемы реляционной базы данных и хорошо подходит для составления операций по отображению элементов

данных модельных представлений. Это облегчит процесс составления алгоритмов для выбора стратегий ОРО и самих операций, осуществляющих отображение.

## **2.4 Обобщенный алгоритм отображения объектно-ориентированных схем баз данных в реляционные**

Процесс отображения объектно-ориентированной схемы базы данных в реляционную состоит из следующих этапов:

- анализ объектно-ориентированной модели и выделение из нее элементов для отображения (на уровне классов и на уровне атрибутов с методами);
- анализ пользовательских требований к отображению;
- для каждого элемента отображения выявление паттерна отображения и дальнейшее его отображение.

Процесс анализа объектно-ориентированной модели данных разбит на два этапа:

- анализ на уровне классов;
- анализ на уровне атрибутов и методов.

На каждом из этих этапов выявляются определенные элементы модели для отображения, на уровне классов – это классы и иерархии наследования, а на втором этапе выявляются такие элементы отображения как атрибуты классов, методы, связи между классами и т.д. Для описания алгоритмов предполагается использовать естественный язык.

Целью анализа объектно-ориентированной модели на уровне классов является выявление в модели групп классов, связанных между собой отношениями наследования, иначе говоря, выявление иерархий наследования. Такая иерархия представляет собой отдельный элемент отображения и поэтому требует обособления от остальных элементов отображения, которыми являются другие классы и иерархии наследования.

Анализ начинается с обхода классов модели и в результате причисления каждого класса к одной из 2 групп:

- класс, состоящий в иерархии наследования;
- обособленный класс, который не состоит в иерархии наследования.

При первом обходе в первую группу попадают те классы, которые унаследованы от других классов. При этом в группу добавляется не только унаследованный класс, но и все классы, от которых он унаследован, вплоть до корневого элемента иерархии. То есть в группу добавляется не отдельный класс, а сразу упорядоченный список классов, связанных отношением наследования. Назовем такой список ветвью наследования. Если при добавлении в группу ветвь наследования А покрывает другую ветвь В, то ветвь В заменяется ветвью А.

Для примера возьмем классы со следующими названиями: А, В, С, D, Е, F, G, H, I, и J. От класса А наследуются классы В, Е и D, от класса D наследуются классы J и G. От класса С наследуются классы F и I, а от класса I наследуется класс H. Данные классы и отношения между ними показаны на рисунке 5.

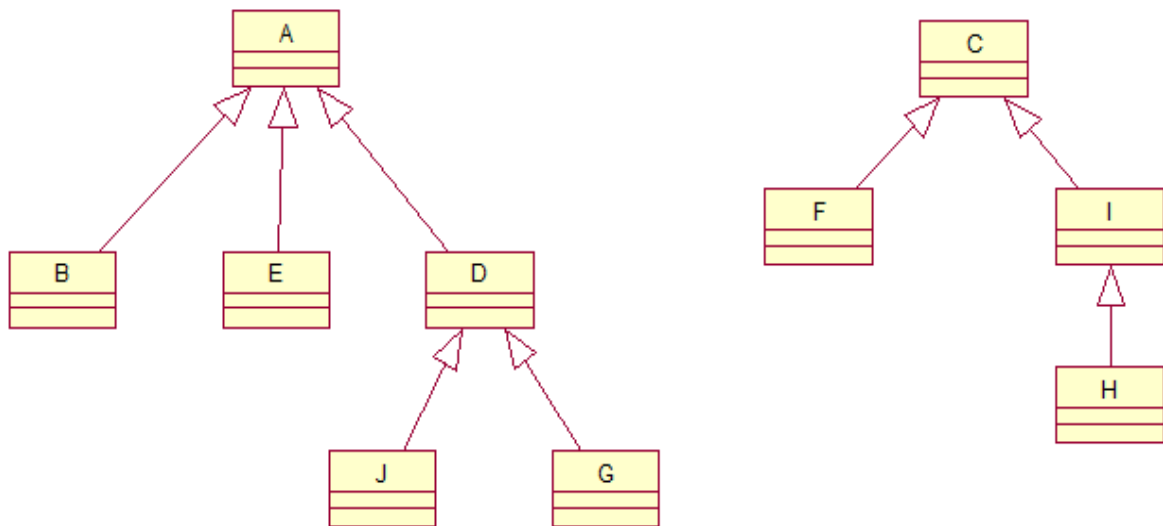


Рисунок 5 – иерархия наследование классов

После добавления всех ветвей в группу получится следующие содержимое (рисунок 6).



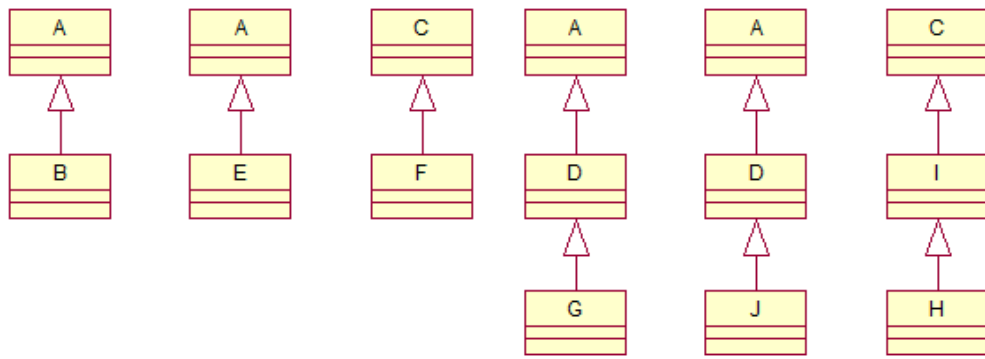


Рисунок 6 – Ветви наследования

Затем происходит объединение ветвей наследования в иерархии. Это происходит путем присоединения к одной ветви иерархии всех остальных ветвей, при этом дублирующие классы убираются.

Выявление элементов отображения внутри отдельного класса осуществляется с учетом следующих принципов:

- каждый атрибут класса представляет собой элемент отображения;
- каждый метод класса является элементом отображения.

Анализ пользовательских требований к отображению предполагает опрос пользователя с целью определения основных требований, выдвигаемых к результирующей схеме БД. Требования выдвигаются по следующим видам критериев:

- общие критерии выбора стратегии отображения;
- критерии, связанные с характеристиками элементов отображения;

Были выделены следующие свойства, относящиеся к критериям отображения:

- название критерия;
- область значений (домен);
- приоритет критерия, его важность.

Приоритет учитывается при наличии взаимоисключающих критериев, которые влияют на выбор паттерна отображения. В таких ситуациях предпочтение отдается более приоритетному критерию.

С общими критериями связаны следующие характеристики реляционных БД, которые варьируются от выбора той или иной стратегии отображения, и на которые накладываются пользовательские требования:

- цель использования БД (накладывает требования на производительность запросов);
- сложность результирующей схемы БД;
- размер таблиц БД;
- удобство модификации схемы;
- степень неиспользуемой памяти.

На производительность запроса влияет объем запрашиваемой таблицы, а также количество операций соединения, необходимое для реализации запроса. Критерий производительности может выражаться следующими значениями: низкая, средняя, максимальная.

Удобство сопровождения и администрирования схемы БД прежде всего зависит от сложности самой схемы (количества таблиц и связей между ними). Чем больше таблиц и связей между ними в базе, тем сложнее воспринимать данную схему и работать с ней. Критерий сложности схемы БД может принимать следующие значения: минимальная, умеренная, сложная.

Удобство модификации схемы во многом зависит от появления аномалий модификации структуры таблиц, например изменение атрибута в одной таблице требует такого же изменения в других таблицах, где этот атрибут встречается.

Степень неиспользуемой памяти таблицы определяется количеством столбцов, которые заполнены частично. И чем больше создается пустых значений, для которых выделена память, тем выше степень неиспользуемой памяти. Данный критерий характеризуется следующими значениями: минимальная, средняя, высокая.

К критериям связанным с характеристиками элементов отображения относятся:

- размер домена атрибута;
- объем занимаемой памяти атрибута;

- глубина иерархии наследования.

В зависимости от размера домена, атрибуты базовых типов могут подразделяться на перечислимые и не перечислимые типы. Атрибуты перечислимых типов (перечисления) отличаются ограниченной областью значений, это могут быть такие атрибуты как пол, страна, тип кузова и т.д.

Для большинства доменов характерны небольшие по объему занимаемой памяти атрибуты, большие по объему атрибуты характерны для таких структур данных как массивы, файлы, изображения, крупный текст и т.д.

В качестве альтернативных вариантов выбора выступают паттерны отображения, их можно подразделить на следующие группы, в зависимости от типов элементов отображения, для которых эти паттерны применяются:

- паттерны отображения простых атрибутов;
- паттерны отображения атрибутов-массивов;
- паттерны отображения объектных атрибутов;
- паттерны отображения коллекций объектов;
- паттерны отображения иерархий наследования.

В таблице 6 перечислены паттерны отображения по группам, при отображении из каждой группы выбирается только один подходящий паттерн.

Таблица 6 – Список альтернативных паттернов отображения

№	Название группы	Паттерн отображения
1	Паттерны отображения простых атрибутов	Прямое отображение
		Отображение в бинарный поток
		Отображение перечислений
2	Паттерны отображения атрибутов-массивов	Отображение в бинарный поток
		Таблица для каждого типа массива
		Таблица для конкретного атрибута-массива
3	Паттерны отображения объектных атрибутов	Включение в состав главной таблицы (отображение встроенного класса)
		Отображение «один к одному»

## Продолжение таблицы 6

№	Название группы	Паттерн отображения
4	Паттерны отображения коллекций объектов	Отображение «один ко многим»
		Отображение «многие ко многим»
5	Паттерны отображения иерархий наследования	Одна таблица на иерархию наследования (Filtered mapping)
		Горизонтальное отображение (Horizontal mapping)
		Вертикальное отображение (Vertical mapping)

В таблицах 7-11 каждому паттерну отображения перечислены характерные для него значения критериев.

Таблица 7 – значения критериев для паттернов отображения простых атрибутов

Название паттерна	Производительность запросов	Размер домена атрибута	Объем занимаемой атрибутом памяти
Прямое отображение	Максимальная	Неограниченный	Небольшой
Отображение перечислений	Максимальная	Ограниченный	Минимальный
Отображение в бинарный поток	Средняя	Неограниченный	Большой

Таблица 8 – значения критериев для паттернов отображения атрибутов-массивов

Название паттерна	Производительность запросов	Сложность схемы БД	Размер таблиц
Отображение в бинарный поток	Средняя	Минимальная	-
Таблица для типа массива	Средняя	Умеренная	Большой
Таблица для каждого атрибута массива	Максимальная	Сложная	Средний

Таблица 9 – значения критериев для паттернов отображения объектных атрибутов

Название паттерна	Производительность запросов	Сложность схемы БД
Отображение встроенного класса	Максимальная	Минимальная
Отображение «один к одному»	Средняя	Средняя

Таблица 10 – значения критериев для паттернов отображения коллекций объектов

Название паттерна	Производительность запросов	Сложность схемы БД	Кардинальность связи
Отображение «один ко многим»	Максимальная	Минимальная	1..М
Отображение «многие ко многим»	Средняя	Средняя	М..М

Таблица 11 – значения критериев для паттернов отображения иерархий наследования

Название паттерна	Производительность запросов	Сложность схемы БД	Степень неиспользуемой памяти	Сложность модификации схемы
Одна таблица на иерархию	Максимальная	Минимальная	Высокая	Минимальная
Горизонтальное отображение	Средняя	Средняя	Минимальная	Средняя
Вертикальное отображение	Средняя	Средняя	Минимальная	Минимальная

С целью выявления значений критериев отображения организуется диалог между пользователем и средством отображения. Диалог организован в форме вопрос-ответ и предполагает ответ пользователя на следующие вопросы:

- функциональное назначение проектируемой базы данных;
- область применения проектируемой базы данных;

- количество пользователей базы данных;
- потребность в администрировании базы данных;
- частота модификации схемы базы данных [17].

По функциональному назначению базы данных подразделяются на OLTP и OLAP системы. Для OLTP систем важна высокая производительность запросов, а для OLAP систем приоритетнее эффективное использование памяти, выделяемое под таблицы [18].

Реляционные базы данных могут применяться в следующих сферах:

- в организациях для работы учетных систем и ведения электронного документооборота;
- в области web для хранения данных о пользователях, текстов сообщений и прочих сущностей;
- в мобильных приложениях для долговременного хранения используемых данных или настроек самого приложения [19].

Базы данных подразделяются на однопользовательские и многопользовательские, при увеличении количества пользователей нагрузка на базу возрастает пропорционально.

Почти все реальные базы данных нуждаются в администрировании. И удобство администрирования во многом зависит от сложности самой схемы БД. С увеличением сложности схемы, базу данных сложнее воспринимать и работать с ней.

Некоторые базы данных в процессе работы с ними подвергаются изменению ее схемы. И чем меньше проблем возникает при модификации схемы БД, тем ее легче администрировать и расширять.

На основании вышеизложенного материала был составлен алгоритм выбора паттерна отображения из каждой группы. Алгоритм представлен в форме дерева решений (рисунки 7 – 9).

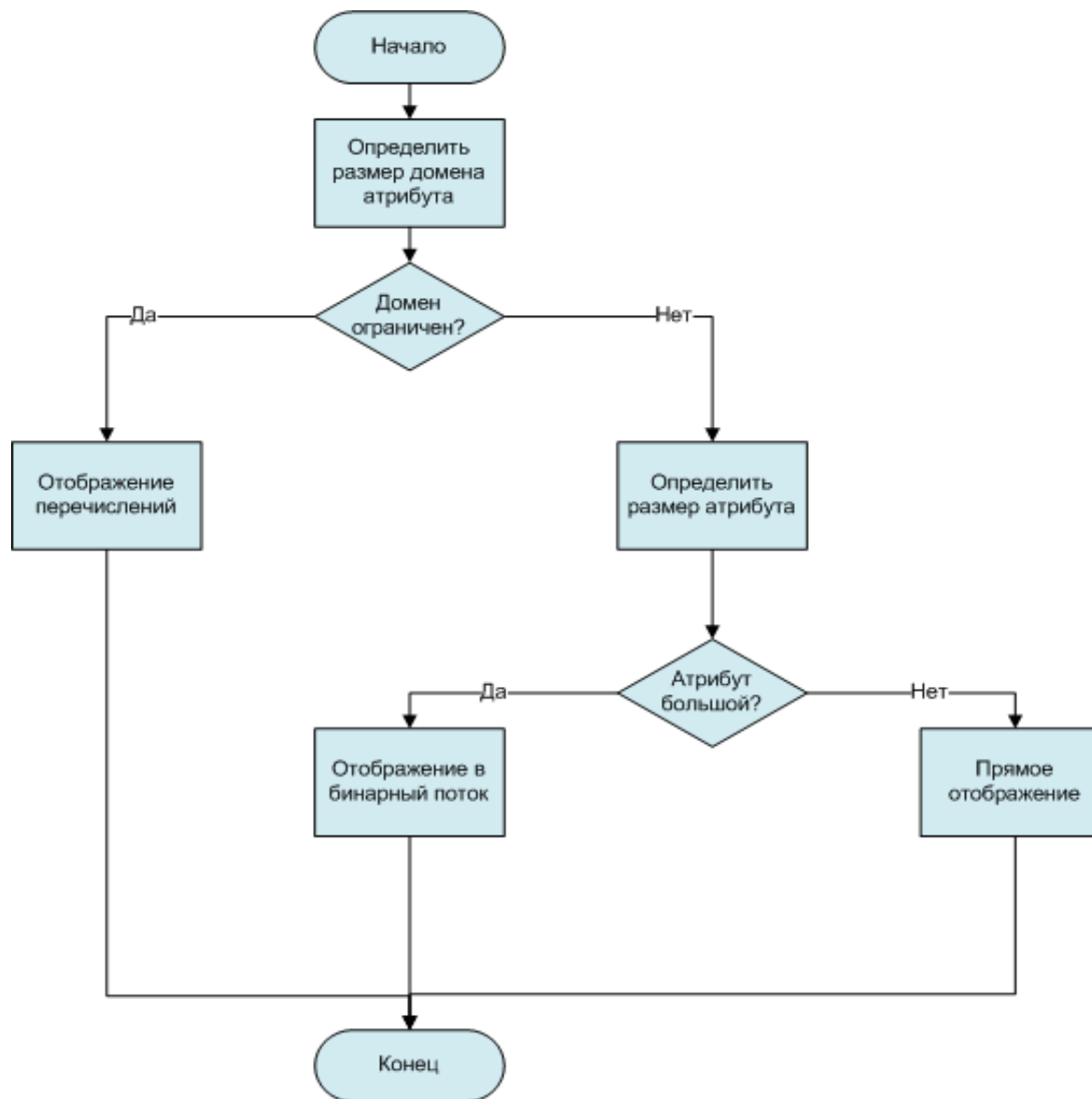


Рисунок 7 – Алгоритм выбора паттерна отображения простых атрибутов

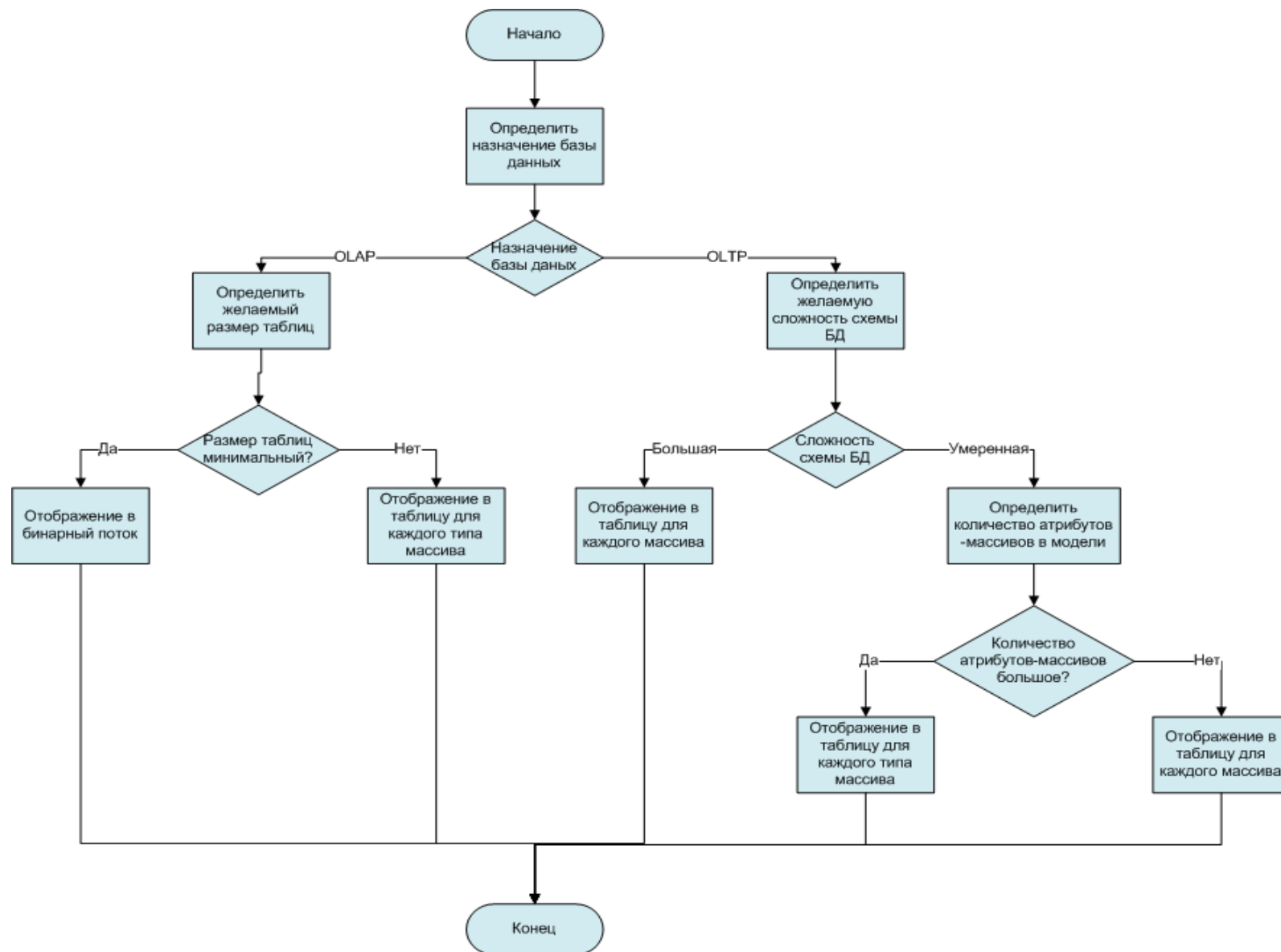


Рисунок 8 – Алгоритм выбора паттерна отображения атрибутов-массивов



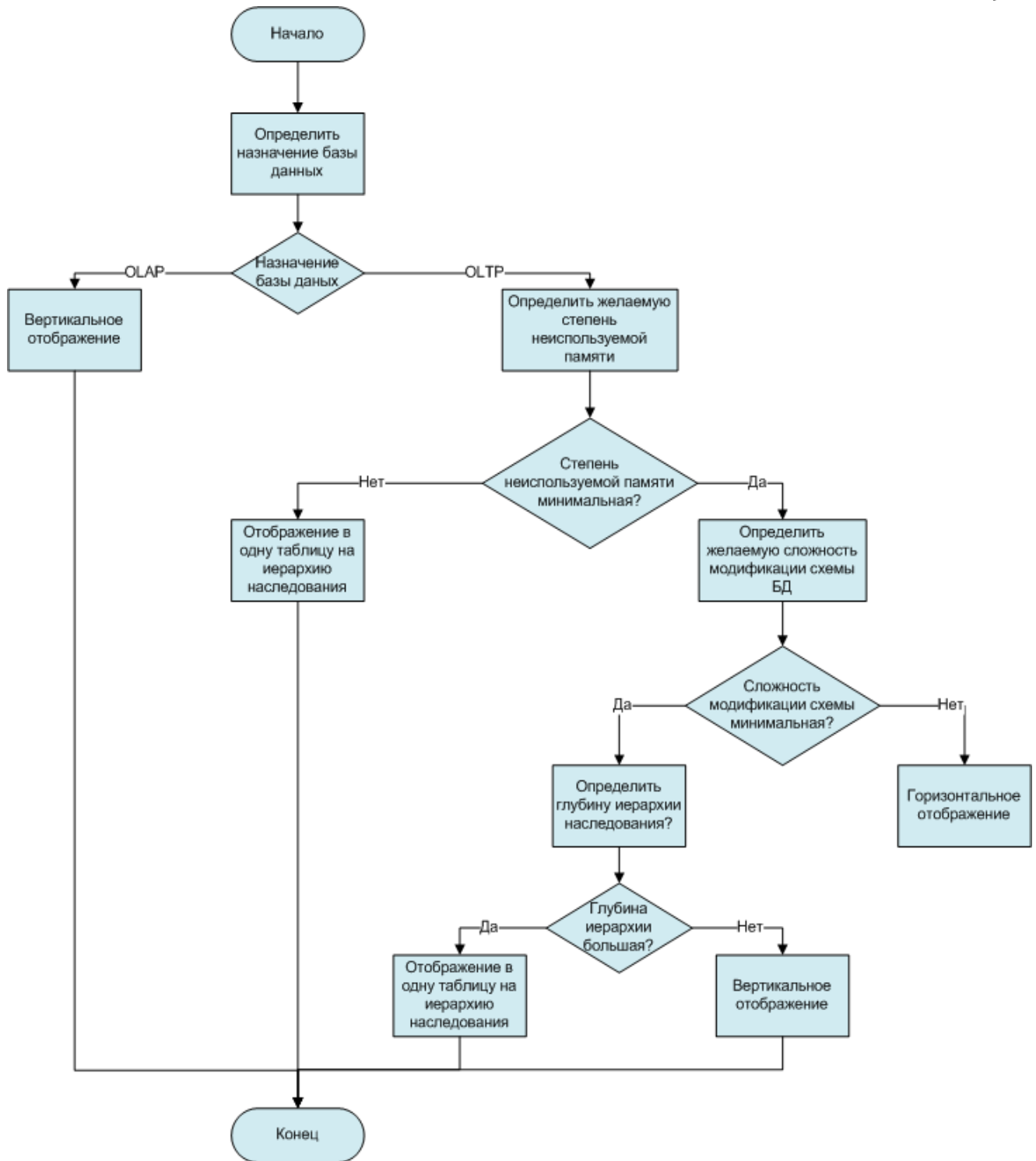


Рисунок 9 – Алгоритм выбора паттерна отображения иерархии наследования

Для отображения объектных атрибутов рекомендуется применять паттерн «Отображение встроенного класса». При отображении коллекции объектов выбор паттерна зависит от кардинальности возникающей между классами связи. В случае связи «один ко многим» выбирается паттерн «Отображение один ко многим», в случае связи «многие ко многим» выбирается паттерн «Отображение многие ко многим».

## Выводы

В данной главе рассмотрено математическое представление для описания объектно-ориентированной и реляционной схемы базы данных. Для описания объектно-ориентированной схемы базы данных и его основных элементов используется формула (1). Для описания структуры реляционной схемы базы данных применяется формула (2).

В каждом математическом представлении описаны основные элементы схем БД и взаимосвязи между ними. Это облегчит анализ процесса объектно-реляционного отображения и поможет разбить схемы на элементы отображения.

Также, с целью дальнейшего использования в алгоритмах, данной главе рассматривались математические операции отображения элементов объектно-ориентированных схем в элементы схем реляционных.

Основываясь на математическом представлении схем БД, был составлен обобщенный алгоритм отображения объектно-ориентированных схем БД в реляционные схемы. В рамках данного алгоритма были выявлены основные критерии, влияющие на выбор стратегии отображения. Были составлены алгоритмы, которые с учетом критерий выбирают паттерн для отображения каждого элемента объектно-ориентированной схемы БД.

### **3 Программная реализация и экспериментальная оценка эффективности разработанных алгоритмов**

#### **3.1 Назначение и основные возможности программных средств объектно-реляционного отображения**

Процесс объектно-реляционного отображения не тривиален. В рамках него осуществляется анализ входной объектно-ориентированной модели данных, в ходе которого выявляются наборы элементов модели, которые можно отобразить. Затем, учитывая особенности самой объектно-ориентированной модели, а также пользовательских требований к результатам отображения, происходит выбор шаблона (паттерна) отображения для каждого выявленного элемента отображения. Сам паттерн отображения также представляет собой алгоритм, на вход которого поступает элемент объектно-ориентированной схемы (модели), и в результате создается соответствующий элемент схемы реляционной базы данных.

Для решения этих задач целесообразно использовать программные средства, предназначенные для выполнения объектно-реляционного отображения. Данное программное средство разрабатывается в рамках данной работы. Его основное назначение – обеспечение эффективного и гибкого отображения объектно-ориентированных схем баз данных с поддержкой основных шаблонов отображения и с учетом пользовательских требований к результирующей реляционной схеме базы данных. Далее описаны основные возможности разрабатываемого программного средства.

Создание, редактирование и удаление объектно-ориентированной модели и ее элементов. Средство позволяет добавлять классы, редактировать их содержание и устанавливает зависимости между классами при их наличии.

Создание наглядной диаграммы объектно-ориентированной модели. Диаграмма показывает все классы модели, атрибуты и методы классов, связи ассоциации и наследования между классами генерируются автоматически.

Сохранение созданного проекта и ее дальнейшей загрузки из XML-файла или же из базы данных проектов.

Выполнение анализа объектно-ориентированной модели и ее дальнейшее отображение в схему реляционной базы данных. В ходе этого процесса также происходит диалог с пользователем системы, в результате которого определяются значения критериев, влияющих на ход отображения.

Анализ результатов объектно-реляционного отображения. Этот процесс предполагает сравнение полученных схем реляционных БД по основным критериям.

На рисунке 10 представлены варианты использования к разрабатываемому средству ОРО.

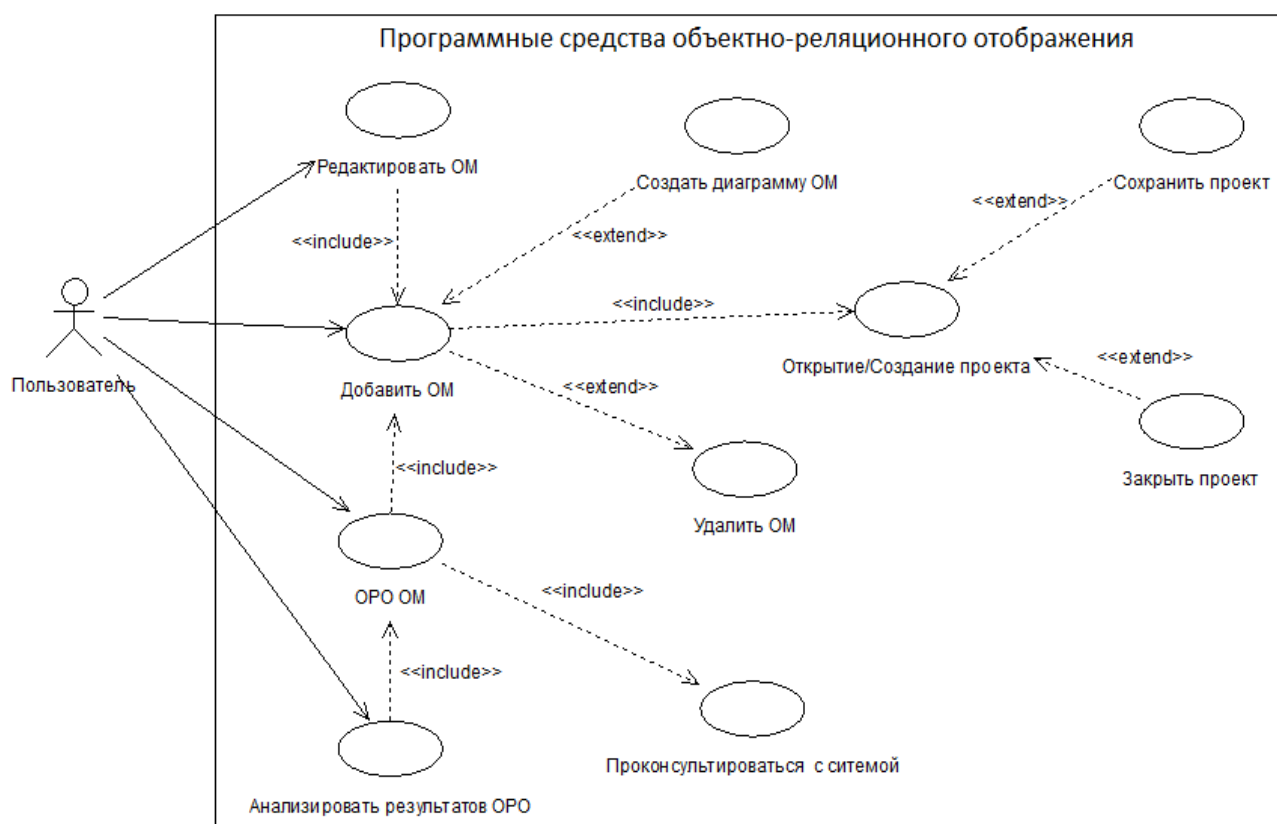


Рисунок 10 – Диаграмма вариантов использования программных средств объектно-реляционного отображения

Основным прецедентом для данной системы является выполнение объектно-реляционного отображения, в ходе этого процесса вызывается прецедент консультации пользователя с системой. Все функции системы связанные с отображением и редактированием проекта распределены между следующими компонентами:

- пользовательский интерфейс;

- модуль экспорта и загрузки проекта;
- модель метаданных;
- система консультации и опроса пользователя;
- модуль, выполняющий объектно-реляционное отображение;
- модуль оценки результатов отображения;

Для детализации прецедентов консультации пользователя с системой и объектно-реляционного отображения были составлены диаграммы последовательности (рисунки 11 и 12 соответственно), в которых участвуют вышеперечисленные компоненты системы.

В ходе диалога с пользователем система опрашивает пользователя по таким вопросам как:

- функциональное назначение проектируемой базы данных;
- сфера применения базы данных;
- планируемое количество пользователей базы;
- частота внесения изменений в схему базы данных.

По функциональному назначению базы данных могут проектироваться для систем делового анализа и хранилища данных (OLAP) или для систем оперативной обработки транзакций (OLTP).

OLTP-системы применяются для оперативного ежедневного учета различных хозяйственных операций предприятий, в web-сайтах и т.д. В общем случае, OLTP-системы предназначены для обработки несложных, но часто поступающих запросов пользователей базы данных. Для базы данных оперативной обработки транзакций важна высокая производительность запросов, для того чтобы обеспечить минимальное время отклика на запрос. Размеры таблиц часто варьируется в пределах тысяч и десятков тысяч строк. Таким образом, при выборе пользователя использовать базу для OLTP системы, средство ОРО накладывает приоритет на производительность запросов допуская не самое эффективное использование памяти.

OLAP-системы используются для принятия решений путем анализа большого объема данных, которые были накоплены за относительно большой промежуток времени. Для базы данных делового анализа характерен большой

объем таблиц, который исчисляется миллионами строк, и не такие высокие требования к производительности запросов, как в OLTP-системах.

Реляционные базы данных могут применяться в учетных системах организаций, в web-приложениях и сервисах, а также в качестве локальных баз данных для «десктопных» и мобильных приложений.

Чаще всего реляционные базы данных – многопользовательские, и от количества пользователей пропорционально зависит уровень нагрузки на базу данных и часто объем самой базы.

Частота изменения схемы базы данных накладывает требования на сложность модификации и сложность самой схемы БД.

Учитывая все вышеперечисленные требования, средство ОРО по каждому критерию отображения определяет его значение и выделяет приоритетные критерии. Но для учета всех критериев отображения также необходимо провести анализ самой объектно-ориентированной схемы. В результате анализа схема разбивается на элементы отображения (классы, иерархии наследования, различные виды атрибутов). В зависимости от количества элементов отображения каждого вида, от глубины иерархий наследования, от размерности массивов и сложности самой схемы, система может предложить различные варианты отображения объектно-ориентированной схемы БД.

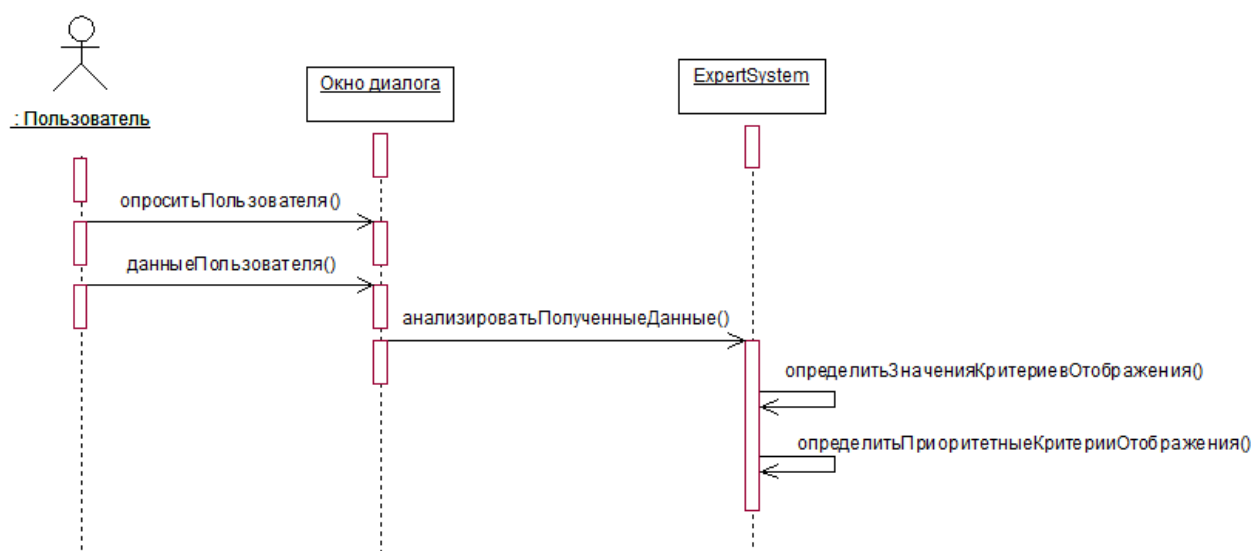


Рисунок 11 – Диаграмма последовательности для прецедента «Проконсультироваться с системой»

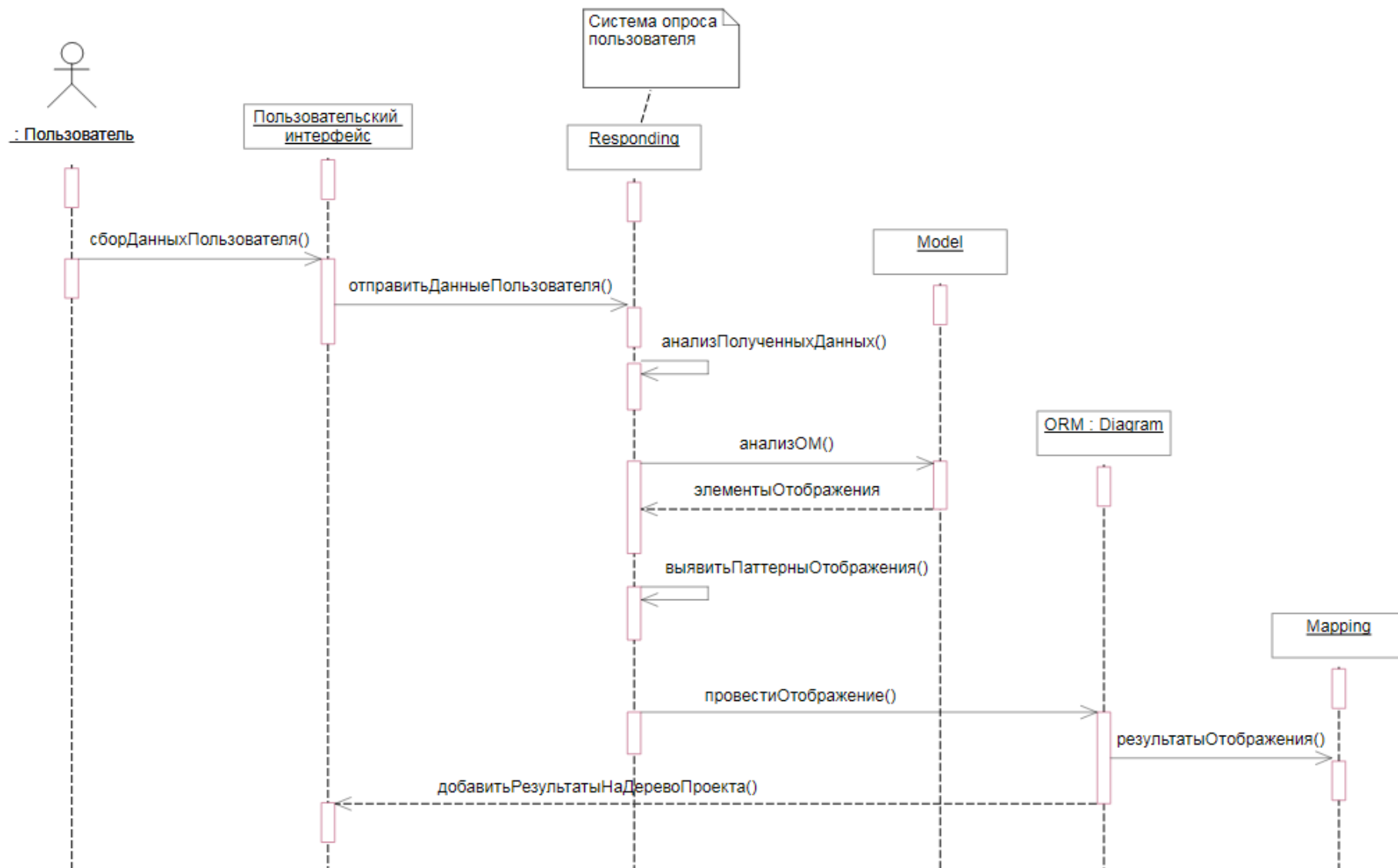


Рисунок 12 – Диаграмма последовательности для прецедента «Объектно-реляционное отображение»

В конечном итоге для каждого элемента отображения применяется подходящий для него паттерн отображения по алгоритмам, описанным в разделе 2.4.

### 3.2 Архитектура программных средств объектно-реляционного отображения

В качестве хранилища созданных проектов средство объектно-реляционного отображения использует базу метаданных, которая хранится на соответствующем сервере. К данной базе могут подключаться другие клиенты, которые могут производить другие операции, как с объектно-ориентированной схемой, так и с реляционной схемой базы данных. Задача программного средства ОРО – достать данные проекта из базы метаданных, провести операцию объектно-реляционного отображения и загрузить проект обратно в базу метаданных.

Диаграмма развертывания средства объектно-реляционного отображения и базы метаданных представлена на рисунке 13.

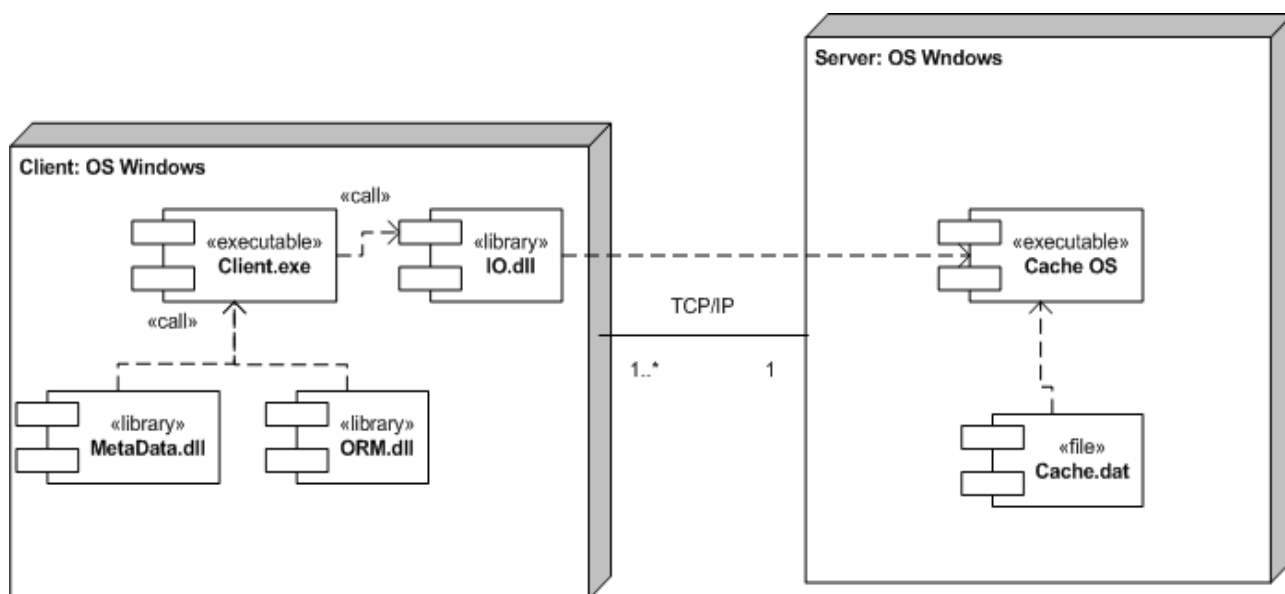


Рисунок 13 – Общая архитектура программных средств ОРО

Клиентское ПО ОРО представляет собой разрабатываемое в рамках данной работы программное средство. Диаграмма компонентов приложения приведена на рисунке 14.



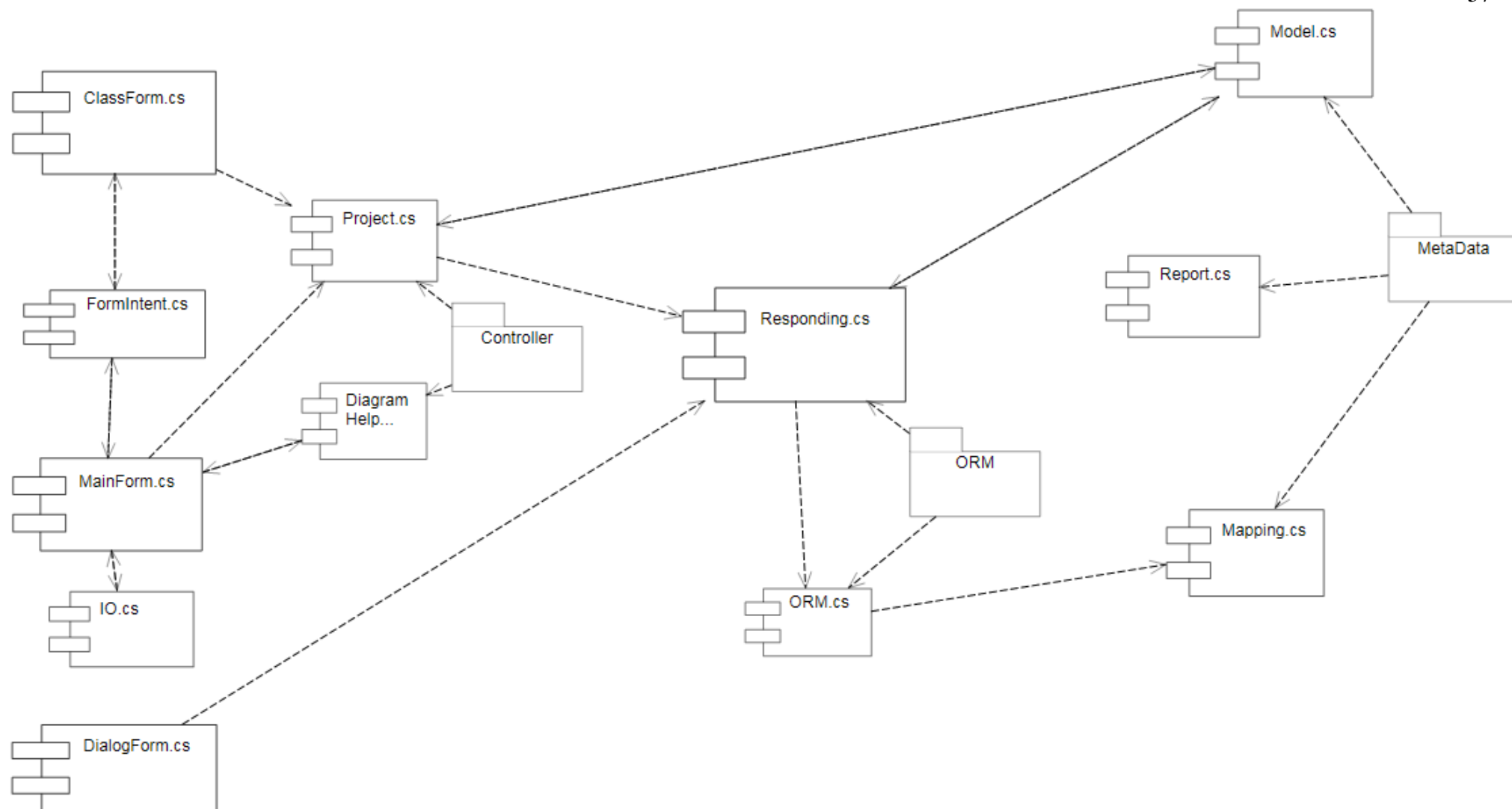


Рисунок 14 – Диаграмма компонентов проекта программных средств ОРО

Пользовательский интерфейс программы обеспечивается при помощи форм MainForm.cs, ClassForm.cs и DialogForm.cs.

MainForm.cs представляет собой окно проекта программы, в ней содержится основное меню программы, дерево проекта и область рисования диаграмм. ClassForm.cs представляет собой окно редактирования свойств класса. В данной форме можно изменить имя класса, редактировать список его атрибутов и свойства самих атрибутов. DialogForm.cs организует опрос пользователя на выявление значений основных критериев отображения. Класс FormIntent.cs служит посредником для обмена данными и командами между формами приложения.

Пакет MetaData содержит следующие классы, описывающие элементы проекта:

- Model.cs – класс, описывающий объектно-ориентированную модель и содержащий методы для анализа соответствующей модели;
- Mapping.cs – класс, описывающий отображение объектно-ориентированной модели в схему реляционной базы данных;
- Report.cs – класс, описывающий результаты оценок схем реляционных БД.

Пакет Controller содержит компоненты, которые выступают посредниками между классами пользовательского интерфейса и классами из пакета MetaData. Это классы Project.cs и DiagramHelper.cs.

Компонент Project.cs является классом и содержит методы для добавления, изменения и удаления списка моделей и классов моделей. Данный класс организует взаимодействие дерева проекта из MainForm.cs с классами из пакета MetaData.

Компонент DiagramHelper.cs является классом, содержащим методы для расстановки элементов диаграммы, для отрисовки элементов объектно-ориентированной модели. Также этот класс содержит методы, облегчающие организацию взаимодействия пользователя с элементами диаграммы.

Пакет ORM содержит компоненты, непосредственно отвечающие за объектно-реляционное отображение, это классы Responding.cs и ORM.cs.

Класс `Responding.cs` обрабатывает результаты опроса пользователя, поступающие из компонента `DalogForm.cs`. Класс `Responding.cs` определяет значения основных критериев отображения, и определяет приоритетные критерии. На основе этих данных выбирается паттерн отображения для каждого элемента отображения.

Класс `ORM.cs` содержит в себе методы для выполнения объектно-реляционного отображения. При этом отображение происходит поэлементно, каждый элемент отображается при помощи подходящего для него паттерна.

### **3.3 Спецификации программных средств объектно-реляционного отображения**

В пакете `MetaData` содержатся классы `Model`, `ModelClass`, `Attribute`, `ModelRelation`, `Mapping`, `Table`, `TableAttribute`, `Relationship` и `ModelElement`. Эти классы соответствуют элементам объектно-ориентированной и реляционной схемы базы данных.

Класс `Model` содержит в себе список классов объектно-ориентированной схемы и список связей между классами. Также этот класс содержит функции по анализу объектно-ориентированной схемы. Класс `ModelClass` соответствует классу объектно-ориентированной схемы и содержит в себе список атрибутов. Класс `Attribute` содержит свойства атрибута класса. Класс `ModelRelationship` соответствует связям ассоциации между классами, когда один класс содержит в себе атрибут, являющийся объектом другого класса объектно-ориентированной схемы. Класс `ModelElement` предназначен для хранения необходимых данных об элементе отображения, может хранить в себе класс, иерархию наследования, связь или атрибут.

В пакете `Controller` содержатся такие классы как `Project` и `DiagramHelper`. Класс `Project` предназначен для организации обмена данными между пользовательским интерфейсом (деревом проекта, реализованным с помощью элемента управления `TreeView`) и классами `Model`, `ModelClass`, `Attribute` и `Mapping`. Класс `DiagramHelper` предназначен для облегчения процесса отображения диаграммы классов в приложении. Класс содержит методы по

расстановке элементов диаграммы и их прорисовки в пользовательском окне программы.

В пакете ORM содержатся классы, реализующие процесс объектно-реляционного отображения. Это классы Responding, Question и ORM. Класс Responding предназначен для обработки результатов опроса пользователя, эти результаты преобразуются в критерии отображения. Класс Question содержит данные о вопросе, задаваемом пользователю.

Спецификации классов из пакета MetaData показаны в таблицах 12 – 19.

Таблица 12 – Спецификация класса Model

Класс Model	
Наименование	Назначение
Model	Класс, соответствующий объектно-ориентированной схеме, реализует методы анализа данной схемы
Атрибуты	
Name	Название схемы
ModelClasses	Список классов
ModelRelationships	Список связей ассоциации между классами
Mappings	Список отображений схемы
ModelElements	Список элементов отображения
Методы	
XElement toXML()	Сохраняет объектно-ориентированную схему в виде XML элемента
void analyze()	Производит анализ объектно-ориентированной схемы для выявления дополнительных критериев отображения
void getModelElements()	Производит выявление элементов отображения
void draw()	Прорисовывает все элементы объектно-ориентированной схемы

Таблица 13 – Спецификация класса ModelClass

Класс ModelClass	
Наименование	Назначение
ModelClass	Соответствует классу объектно-ориентированной схемы и содержит в себе список атрибутов
Атрибуты	
Name	Имя класса
Parent	Ссылка на родительский класс
Model	Ссылка на модель, в котором содержится класс
Attributes	Список атрибутов класса
X, Y	Координаты класса на диаграмме

## Продолжение таблицы 13

Методы	
XElement toXML()	Сохраняет содержимое класса в виде XML элемента
void draw()	Прорисовывает класс на диаграмме

Таблица 14 – Спецификация класса Attribute

Класс Attribute	
Наименование	Назначение
Attribute	Содержит свойства атрибута класса
Атрибуты	
Name	Имя атрибута
Type	Тип атрибута
ObjectType	Объектный тип атрибута
Array	Является ли атрибут массивом

Таблица 15 – Спецификация класса ModelRelationship

Класс ModelRelationship	
Наименование	Назначение
ModelRelationship	Содержит данные о связи ассоциации между классами
Атрибуты	
Type	Тип ассоциации
ClassA	Первый класс
ClassB	Второй класс

Таблица 16 – Спецификация класса Mapping

Класс Mapping	
Наименование	Назначение
Mapping	Класс, соответствующий отображению объектно-ориентированной схемы в реляционную схему
Атрибуты	
Model	Ссылка на объектно-ориентированную схему
Tables	Список таблиц отображения
Relationships	Список связей между таблицами
Методы	
XElement toXML()	Сохраняет реляционную схему в виде XML элемента

Таблица 17 – Спецификация класса Table

Класс Table	
Наименование	Назначение
Table	Соответствует таблице реляционной схемы и содержит в себе список столбцов таблицы

## Продолжение таблицы 17

Атрибуты	
Name	Название таблицы
Attributes	Список столбцов таблицы
Методы	
XElement toXML()	Сохраняет структуру таблицы в виде XML элемента

Таблица 18 – Спецификация класса TableAttribute

Класс TableAttribute	
Наименование	Назначение
TableAttribute	Содержит свойства столбца таблицы
Атрибуты	
Name	Имя столбца
Type	Тип столбца

Таблица 19 – Спецификация класса ModelElement

Класс ModelElement	
Наименование	Назначение
ModelElement	Содержит данные об элементе отображения
Атрибуты	
Type	Тип элемента
Content	Содержимое элемента

Спецификации классов из пакета Controller показаны в таблицах 20 – 21.

Таблица 20 – Спецификация класса Project

Класс Project	
Наименование	Назначение
Project	Содержит в себе классы Model и предназначен для взаимодействия их с пользовательским интерфейсом
Атрибуты	
Models	Список объектно-ориентированных моделей проекта
Reports	Список результатов оценок схем
Методы	
XElement toXML()	Сохраняет структуру проекта в виде XML элемента
void insertModel()	Добавляет объектно-ориентированную модель
void insertModelClass()	Добавляет класс
void getProjectElement()	Возвращает элемент проекта, соответствующий вершине дерева проекта TreeView
void bindElement()	Связывает элемент проекта с вершиной дерева проекта

Таблица 21 – Спецификация класса DiagramHelper

Класс DiagramHelper	
Наименование	Назначение
DiagramHelper	Класс предназначен для облегчения процесса создания диаграммы классов в приложении
Атрибуты	
CurrentModel	Содержит ссылку на изображаемую объектно-ориентированную схему
Методы	
ModelClass getCheckedClass()	Возвращает класс объектно-ориентированной схемы по ее координатам на диаграмме
paceClasses()	Расставляет определенным образом классы на диаграмме, вызывается при создании диаграммы

Спецификации классов из пакета ORM представлены в таблицах 22 – 24.

Таблица 22 – Спецификация класса Responding

Класс Responding	
Наименование	Назначение
Responding	Класс предназначен для обработки результатов опроса пользователя
Атрибуты	
Questions	Содержит список вопросов, задаваемых пользователю
Методы	
Question[] createPoll()	Создать опрос пользователя, возвращает список вопросов
void setAnswers()	Отправить список вопросов с ответами
void analyze()	Анализ полученных из формы данных
void selectMappingPatterns()	Выявить паттерны отображения
void selectElementPattern()	Определить шаблон отображения для элемента отображения

Таблица 23 – Спецификация класса Question

Класс Question	
Наименование	Назначение
Question	Содержит данные о вопросе, задаваемом пользователю
Атрибуты	
Name	Имя вопроса, его идентификатор
Text	Текст вопроса
SelectedAnswer	Выбранный вариант ответа
Answers	Варианты ответа вопроса

Таблица 24 – Спецификация класса ORM

Класс ORM	
Наименование	Назначение
ORM	Предназначен для отображения элементов отображения в элементы схемы реляционной базы данных
Методы	
void map()	Выполнить отображение элемента отображения

Все вышеперечисленные классы системы представлены на диаграмме классов программных средств объектно-реляционного отображения (рисунок 15).

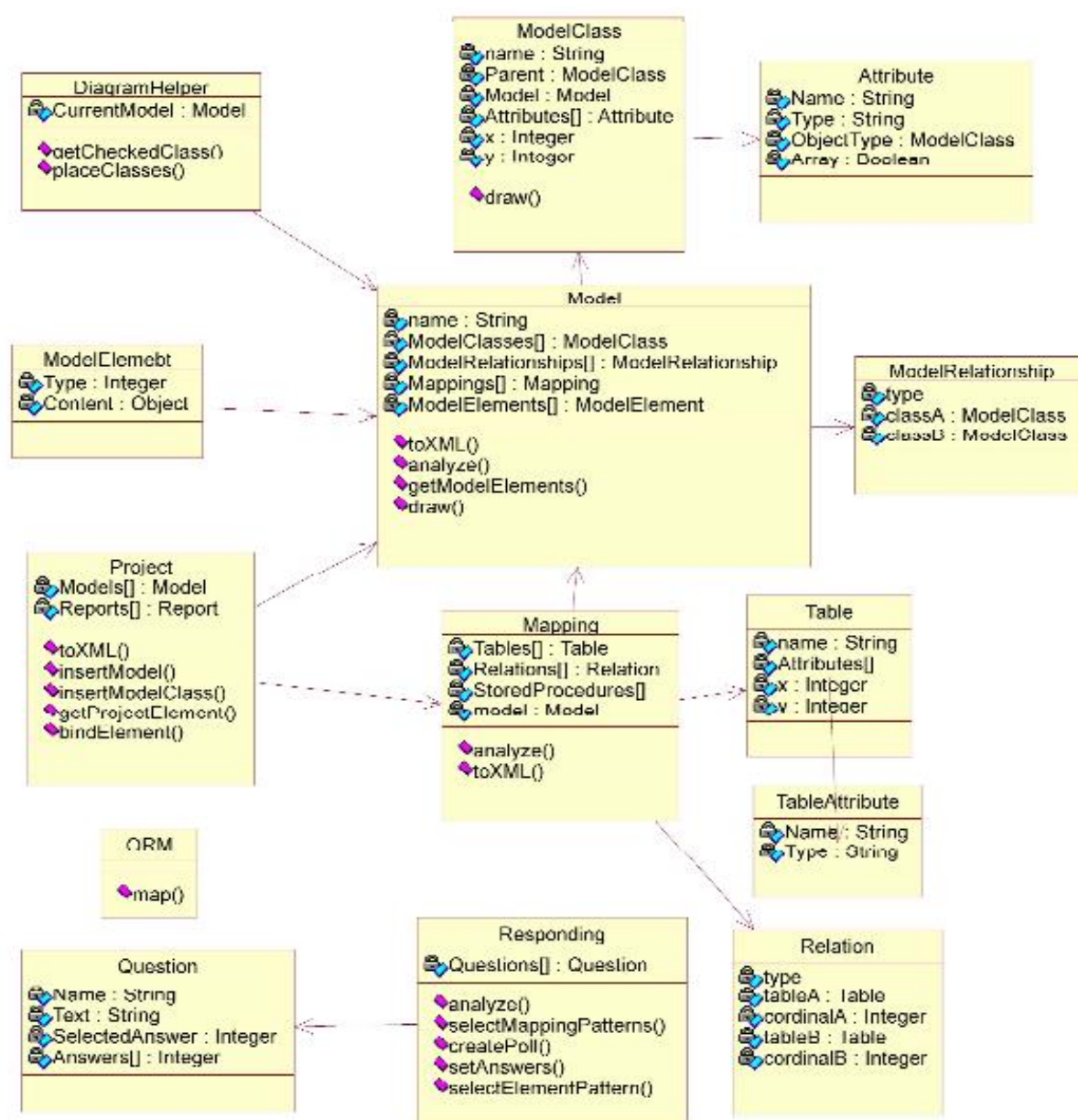


Рисунок 15 – Диаграмма классов программных средств объектно-реляционного отображения



Классы из пакетов MetaData и ORM неявно подключаются к коду основной программы приложения из соответствующих динамических библиотек. Остальные классы содержатся в коде основной программы.

### 3.4 Экспериментальная оценка применения программных средств объектно-реляционного отображения

Для эффективного решения задачи отображения необходимо учитывать как особенности самой объектно-ориентированной схемы, так и характеристики результирующей реляционной базы данных. В разрабатываемых программных средствах объектно-реляционного отображения большое внимание уделяется обеспечению эффективного и гибкого отображения. Гибкость заключается в том, что программные средства могут создать множество вариантов отображения для одной и той же исходной объектной модели. При создании выбирается наиболее эффективный в данном случае вариант отображения.

После запуска программы пользователю предлагается добавить в дерево проекта объектно-ориентированную модель предметной области (Рисунок 16)

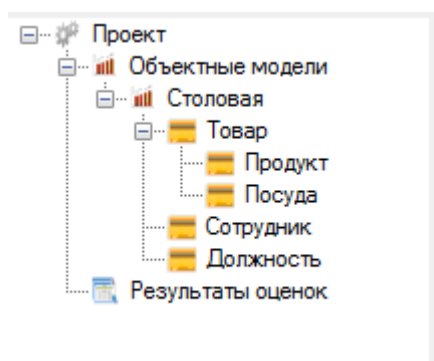


Рисунок 16 – Структура дерева проекта

На данном рисунке изображена объектно-ориентированная схема «Столовая». Классы Продукт и Посуда наследуются от класса Товар. На рисунке 17 показана диаграмма созданной объектно-ориентированной схемы базы данных.

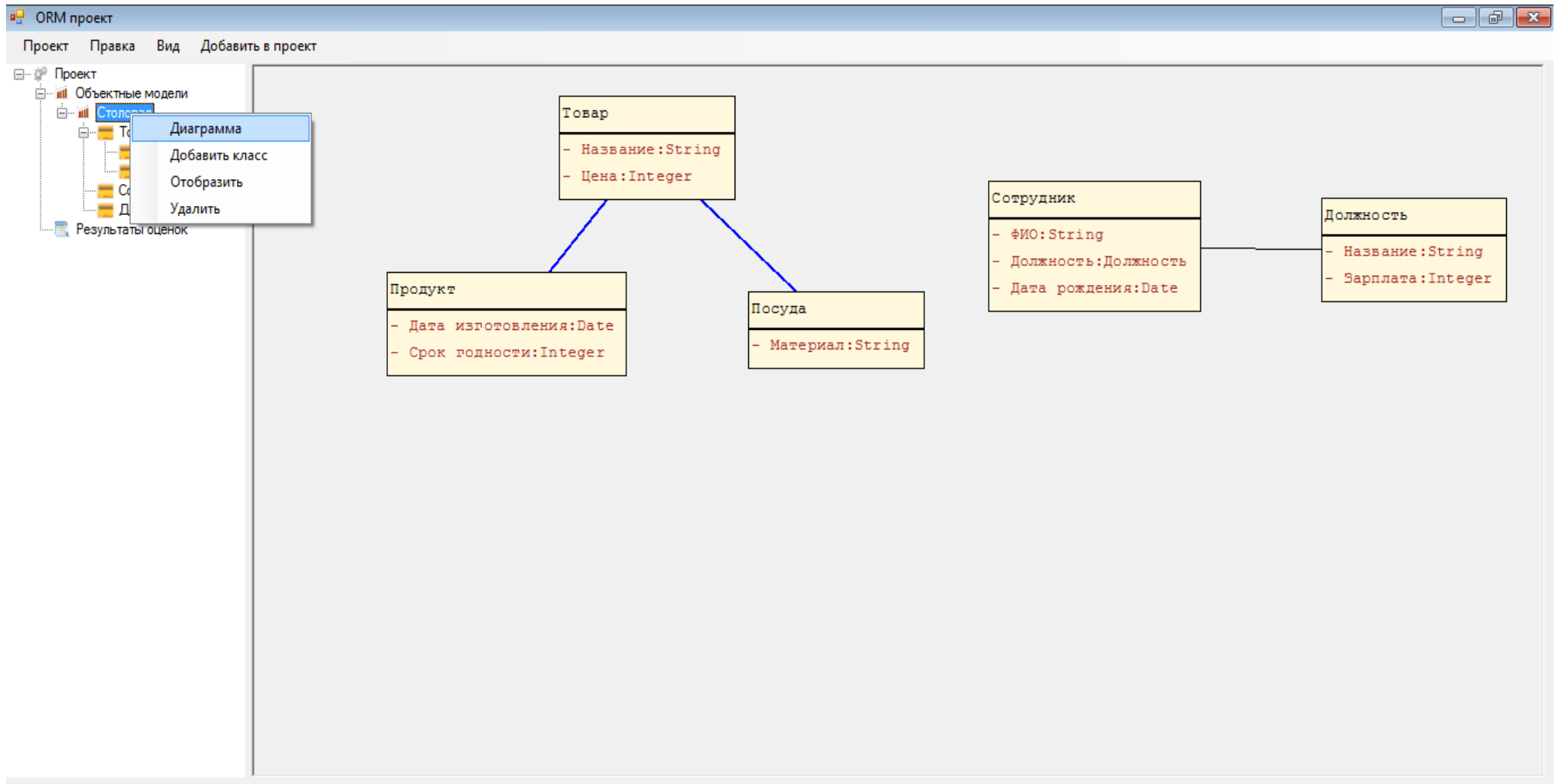
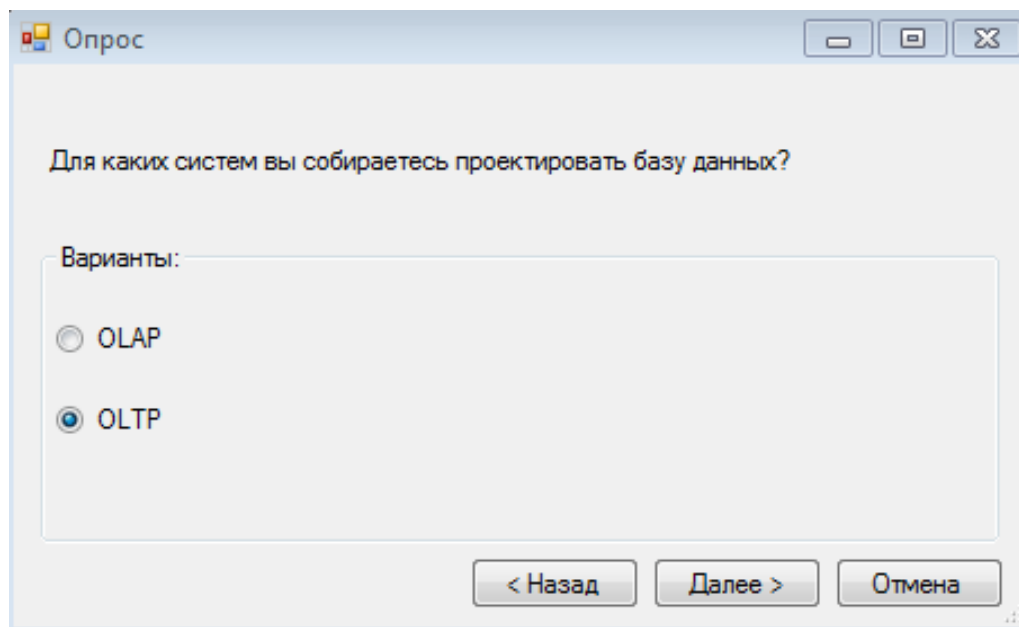


Рисунок 17 – диаграмма объектно-ориентированной схемы базы данных

В начале процесса отображения происходит опрос пользователя, ему задаются вопросы касаются основных характеристик и назначения проектируемой базы данных. На рисунках 18 – 20 показаны отвеченные пользователем вопросы.



Опрос

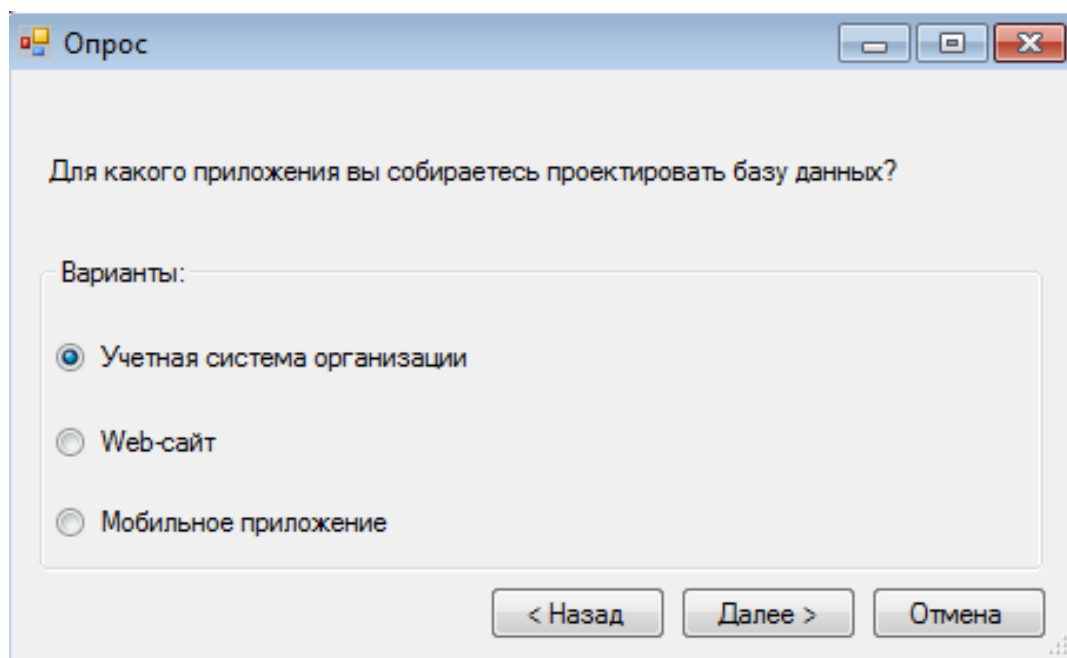
Для каких систем вы собираетесь проектировать базу данных?

Варианты:

- ☐ OLAP
- ☒ OLTP

< Назад    Далее >    Отмена

Рисунок 18 – Ответ на вопрос о назначении проектируемой базы данных



Опрос

Для какого приложения вы собираетесь проектировать базу данных?

Варианты:

- ☒ Учетная система организации
- ☐ Web-сайт
- ☐ Мобильное приложение

< Назад    Далее >    Отмена

Рисунок 19 – Ответ на вопрос о сфере применения проектируемой базы данных

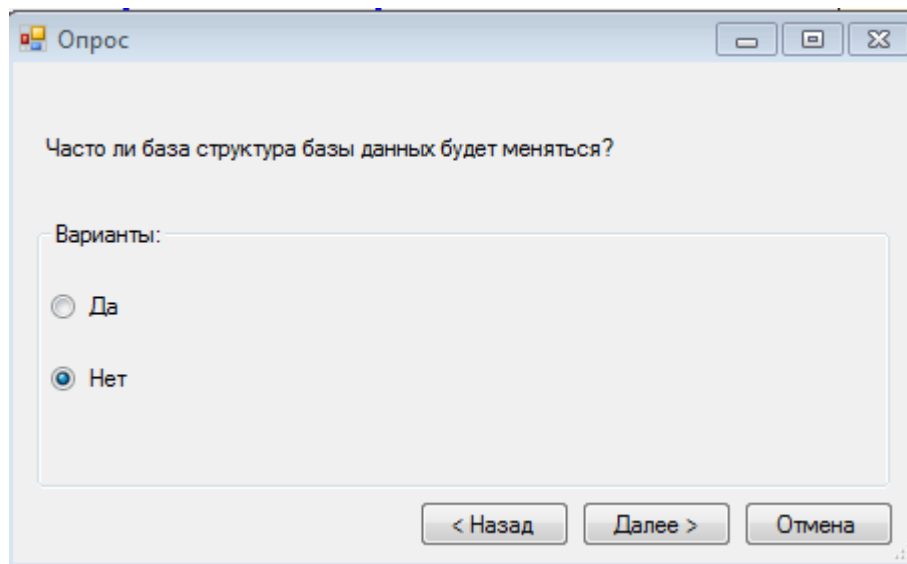


Рисунок 20 – Ответ на вопрос о частоте изменения структуры базы данных

После сбора необходимых данных программными средствами была проанализирована объектно-ориентированная схема базы данных и учтены результаты опроса. В результате объектно-реляционного отображения на выходе образовалась реляционная схема базы данных, которая затем была добавлена в структуру проекта.

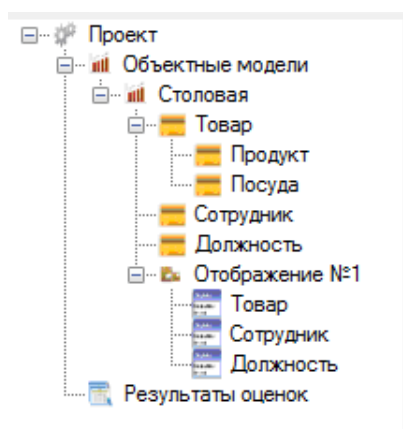


Рисунок 21 – Обновленная структура дерева проекта

Реляционная схема базы данных содержит таблицы Товар, Сотрудник и Должность. Все атрибуты классов Продукт и Посуда вошли в состав столбцов таблицы Товар (рисунок 22), таким образом, при отображении иерархии наследования применен шаблон «Одна таблица на иерархию классов». Такая схема показывает высокую производительность запросов, что играет важную роль для OLTP-систем.

Таблица: Товар

Название: Товар

	Имя столбца	Тип столбца
	Id	NUMBER(10) ▼
	Название	NVARCHAR(50) ▼
	Цена	NUMBER(10) ▼
	Дата_изготовления	DATETIME ▼
	Срок_годности	NUMBER(10) ▼
	Материал	NVARCHAR(50) ▼
	Discriminator	NVARCHAR(50) ▼
▶*		▼

Рисунок 22 – Столбцы таблицы Товар

Если ответить на некоторые вопросы иначе (рисунок 23), то на выходе, скорее всего, получится другой вариант отображения (рисунок 24).

Опрос

Для каких систем вы собираетесь проектировать базу данных?

Варианты:

☒ OLAP

☐ OLTP

< Назад    Далее >    Отмена

Рисунок 23 – Окно опроса пользователя

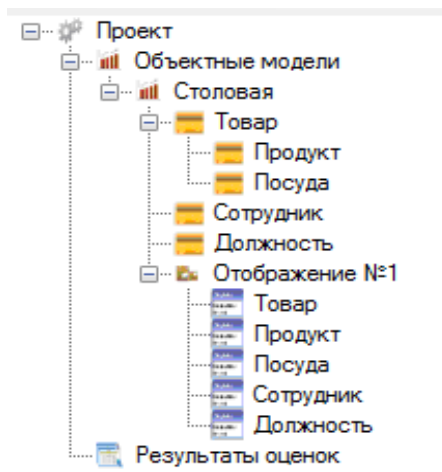


Рисунок 24 – Структура дерева проекта

Сами таблицы будут содержать следующие столбцы, показанные на рисунках 25 – 27.

Таблица: Товар

Название:

	Имя столбца	Тип столбца
	Id	NUMBER(10) ▼
	Название	NVARCHAR(50) ▼
	Цена	NUMBER(10) ▼
▶▶		▼

Рисунок 25 – Столбцы таблицы Товар после повторного отображения

Таблица: Продукт

Название:

	Имя столбца	Тип столбца
	Id	NUMBER(10) ▼
	Дата_изготовления	DATETIME ▼
	Срок_годности	NUMBER(10) ▼
	Id_Товар	NUMBER(10) ▼
▶*		▼

Рисунок 26 – Столбцы таблицы Продукт

Таблица: Посуда

Название:

	Имя столбца	Тип столбца
	Id	NUMBER(10) ▼
	Материал	NVARCHAR(50) ▼
	Id_Товар	NUMBER(10) ▼
▶*		▼

Рисунок 27 – Столбцы таблицы Посуда

В данном случае, при отображении иерархии наследования применен шаблон «Вертикальное отображение». Такая реляционная схема позволяет эффективно использовать выделяемую для хранения ячеек таблиц память.

Разрабатываемые программные средства объектно-ориентированного отображения на примере показали гибкость и эффективность проведения отображения.

## **Выводы**

В данной главе рассматриваются основные возможности, назначение, а также архитектура и спецификации программных средств объектно-реляционного отображения.

Приводится список и описание вариантов использования и компонентов программных средств объектно-реляционного отображения. Для детализации вариантов использования используются диаграммы последовательности.

Приводятся спецификации классов для работы с моделью метаданных, классов, реализующих алгоритмы объектно-реляционного отображения и классов, организующих взаимодействие элементов пользовательского интерфейса с классами модели метаданных.

Реализованные в классах алгоритмы анализа и объектно-реляционного отображения разбивают объектно-ориентированную схему на элементы, позволяют учитывать пользовательские требования к результатам отображения, также учитываются особенности самой объектно-ориентированной схемы базы данных. Далее каждый элемент отображается в соответствии с выбранным для него шаблоном отображения.

По результатам экспериментального анализа можно сделать вывод об эффективности работы разработанных программных средств.



## Заключение

В ходе научно-исследовательской работы были решены следующие задачи:

- исследованы методики проектирования реляционных баз данных;
- исследованы различные стратегии и шаблоны объектно-ориентированного отображения и проведена их оценка;
- были описаны модельные представления и операции отображения схем баз данных;
- были разработаны алгоритмы отображения объектно-ориентированных схем баз данных в реляционные схемы;
- разработаны программные средства объектно-реляционного отображения.

В данной диссертационной работе были исследованы методики проектирования баз данных. В результате данного исследования выяснилось, что в плане описания логической модели предметной области, объектно-ориентированная модель обладает большими возможностями, чем модель «сущность-связь».

Учитывая доминирующее положение реляционных СУБД на рынке и большую популярность объектно-ориентированного подхода в программировании, в целях решения проблемы объектно-реляционного импеданса был предложен подход моделирования базы данных на основе объектно-ориентированной схемы с ее дальнейшим преобразованием в реляционную схему базы данных.

Были исследованы основные стратегии и шаблоны объектно-реляционного отображения. Исследование стратегий и шаблонов отображения показало, что все они находят применение в определенных ситуациях, однако они не универсальны. Выбор оптимальной стратегии отображения позволит максимально эффективно организовать работу информационной системы с базой данных.

Обеспечение эффективного и гибкого объектно-реляционного отображения требует наличия алгоритмов для оценки и выбора подходящей

стратегии отображения. Для этого требуется проанализировать как сам процесс отображения, так и модели данных, с которыми этот процесс оперирует. Для облегчения этой задачи была построена математическая модельное представление объектно-ориентированной и реляционной схемы базы данных. На основании модельных представлений, был составлен обобщенный алгоритм отображения объектно-ориентированных схем БД в реляционные схемы.

Были разработаны программные средства объектно-реляционного отображения, основное назначение которых – обеспечение эффективного и гибкого отображения объектно-ориентированных схем баз данных с поддержкой основных шаблонов отображения и с учетом пользовательских требований к результирующей реляционной схеме базы данных.

Результаты экспериментальных оценок дают основания утверждать, что разработанные алгоритмы и программные средства работают корректно и полностью соответствуют заданным спецификациям.

### Список использованных источников

1. Крёнке, Д. Теория и практика построения баз данных / Д. Крёнке. – М.: Питер, 2003. – 799 с.
2. Мюллер, Дж. Базы данных и UML. Проектирование / Дж. Мюллер. – М.: Лори, 2002. – 356 с.
3. Представление данных с помощью модели «сущность связь» [Электронный ресурс]. URL: [http://www.mstu.edu.ru/study/materials/zelenkov/ch\\_2\\_1.html](http://www.mstu.edu.ru/study/materials/zelenkov/ch_2_1.html) (дата обращения: 15.06.2017).
4. Технологическая архитектура, стандарты и шаблоны [Электронный ресурс]. URL: <http://www.intuit.ru/studies/courses/995/152/lecture/4234?page=8> (дата обращения: 15.06.2017).
5. Модели данных [Электронный ресурс]. URL: <http://www.studfiles.ru/preview/5150865/page:10/> (дата обращения: 16.06.2017).
6. Семенов В.А., Морозов С.В., Порох С.А. Стратегии объектно-реляционного отображения: систематизация и анализ на основе паттернов [Электронный ресурс]. URL: <http://citforum.ru/database/articles/strategy/> (дата обращения: 16.06.2017).
7. Рубанов, В.В. Способы отображения объектов в реляционных базах данных / В.В. Рубанов // Труды института системного программирования РАН. – 2002. – №3. – С. 139 – 164.
8. Цуканов П. ORM технологии в .NET [Электронный ресурс]. URL: <https://www.slideshare.net/ptsukanov/orm-net-nhibernate-linq-to-sql-entity-framework> (дата обращения: 16.06.2017).
9. Литвинюк А. Введение в Hibernate и основы ORM [Электронный ресурс]. URL: <http://www.nestor.minsk.by/kg/2005/32/kg53209.html> (дата обращения: 16.06.2017).
10. Введение в Entity Framework [Электронный ресурс]. URL: <http://metanit.com/sharp/entityframework/1.1.php> (дата обращения: 16.06.2017).

11. LINQ to SQL [Электронный ресурс]. URL: <https://msdn.microsoft.com/ru-ru/en-en/library/bb386976.aspx> (дата обращения: 16.06.2017).
12. 5 лучших ORM для Android [Электронный ресурс]. URL: <http://java-help.ru/5-best-android-orms/> (дата обращения: 16.06.2017).
13. Active Record: ORM и типы данных [Электронный ресурс]. URL: <http://savepearlharbor.com/?p=221499> (дата обращения: 17.06.2017).
14. Современные технологии объектно-реляционного отображения [Электронный ресурс]. URL: [http://lib.custis.ru/Современные\\_технологии\\_объектно-реляционного\\_отображения](http://lib.custis.ru/Современные_технологии_объектно-реляционного_отображения) (дата обращения: 17.06.2017).
15. MyGeneration.NET Code Generator [Электронный ресурс]. URL: <https://www.codeproject.com/Articles/6204/MyGeneration-NET-Code-Generator> (дата обращения: 17.06.2017).
16. Основы теории множеств [Электронный ресурс]. URL: <http://www.studfiles.ru/preview/3676428/page:2/> (дата обращения: 17.06.2017).
17. Методология проектирования баз данных [Электронный ресурс]. URL: <http://www.intuit.ru/studies/courses/3439/681/lecture/14021?page=2> (дата обращения: 17.06.2017).
18. Мировые информационные ресурсы и их использование [Электронный ресурс]. URL: <http://univer-07.narod.ru/tema3.htm> (дата обращения: 17.06.2017).
19. Применения реляционных баз данных [Электронный ресурс]. URL: <http://mech.math.msu.su/~shvetz/54/inf/databases/chApplications.xhtml> (дата обращения: 17.06.2017).