

Санкт-Петербургский государственный университет
Математическое обеспечение и администрирование информационных систем
Кафедра системного программирования

Семенова Анастасия Владимировна

Разработка системы графического проектирования баз данных

Бакалаврская работа

Научный руководитель:
к. т. н., ст. преп. Брыксин Т.А.

Рецензент:
инженер Смирнов К.К.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY
Software and Administration of Information Systems
Software Engineering

Anastasiia Semenova

Development of a database modelling tool

Bachelor's Thesis

Scientific supervisor:
Senior lecturer Timofey Bryksin

Reviewer:
Engineer Kirill Smirnov

Saint-Petersburg
2016

Оглавление

Введение	4
Глава 1. Обзор	6
1.1 Предметная область	6
1.2 Обзор существующих решений	7
1.3 Платформа QReal	10
1.4 Научные работы на кафедре системного программирования СПбГУ	12
1.5 Обучение базам данных в университете	13
Глава 2. Разработка требований	14
2.1 Процесс разработки	14
2.2 Список требований.....	14
Глава 3. Реализация	17
3.1 Архитектура.....	17
3.2 Создание графических языков	18
3.3 Генерация физической схемы из логической схемы	21
3.4 Генерация SQL-кода	23
3.5 Обратная генерация	24
3.6 Создание меню для работы с элементом «Таблица».....	24
Глава 4. Подготовка продукта	27
4.1 Пользовательская документация	27
4.2 Локализация системы	27
4.3 Сборка инсталлятора	27
Глава 5. Апробация	28
5.1 Цели апробации и выбор целевой группы.....	28
5.2 Проведение апробации	28
5.3 Результаты апробации	28
5.4 Доработка по результатам апробации.....	31
Заключение	32
Список литературы	33
Приложение	36

Введение

Сегодня практически любой крупный программный проект взаимодействует с базами данных (БД), поэтому осуществление процесса быстрого и удобного создания БД — актуальная и важная задача. Одно из удачных решений — визуализация БД с помощью моделей «сущность-связь», в которых пользователь описывает сущности предметной области и отношения между ними. Изначально задаётся логическая схема: сущности, атрибуты сущностей, связи. Затем из логической схемы создается физическая схема: сущности становятся таблицами; отношения также преобразуются в соответствующие им вспомогательные таблицы; добавляются детали, специфичные для конкретной СУБД (типы данных атрибутов, дополнительные свойства атрибутов и сущностей). После этого по физической модели можно написать код, создающий базу данных.

Инструментов, позволяющих автоматизировать данный процесс, существует достаточно много. В основном это продукты, которые обладают рядом недостатков, если рассматривать их с точки зрения обучения процессу проектирования. А ведь прежде чем начать проектировать базы данных в реальных задачах, разработчики программ должны научиться делать это правильно на небольших примерах.

На кафедре системного программирования СПбГУ несколько лет разрабатывается проект QReal [14, 29] с открытым исходным кодом, позволяющий быстро создавать визуальные языки программирования и инструментальную поддержку для них. До данной работы на базе QReal не было создано полноценного инструмента для работы с базами данных. Разработка такого средства позволила бы изучить и проанализировать возможности платформы, созданного на кафедре, а также расширить его функциональность. Повторение уже существующих программных продуктов проектирования БД не является целесообразной задачей. Зато можно было бы создать легковесный и легкий в использовании инструмент, предназначенный для использования на занятиях по БД, который бы учитывал недостатки существующих решений с точки зрения процесса обучения. Так на базе QReal было решено разработать систему QReal:Databases для графического проектирования БД, предназначенную для обучающихся.

В рамках данной работы были поставлены следующие задачи:

- разработать требования к системе;
- реализовать функциональность инструмента по выработанным требованиям;

- создать локализацию системы для русского и английского языков, создать пользовательскую документацию, собрать инсталлятор;
- произвести апробацию на целевой группе и доработать приложение по результатам апробации.

Работа над проектом длилась 2 года, часть результатов была описана в курсовой работе 2015 года [28].

Глава 1. Обзор

1.1 Предметная область

База данных (БД) — это совокупность систематизированных особым образом данных, находящаяся в памяти вычислительной системы [25]. Для работы с БД используются специальные средства — системы управления базами данных (СУБД), которые обеспечивают централизованное создание и редактирование базы данных многими пользователями. На сегодняшний день большое распространение получили реляционные БД. Информация в таких базах данных хранится в таблицах из строк и столбцов.

При моделировании БД используются различные уровни абстракции. Чаще всего выделяют концептуальный, логический и физический уровни, определения которых не являются строго фиксированными. Например, достаточно часто объединяют концептуальную и логическую схему. Под термином «физическая схема» может подразумеваться описание деталей хранения объектов на физическом носителе [22, 30] или просто расширение логической схемы возможностями конкретной СУБД [27].

Как правило, определение уровня абстракции вводится в каждом источнике перед использованием, поэтому читатель осведомлен, какой смысл вкладывает автор в конкретный термин. Ниже представлены определения, которые выбраны для использования в QReal:Databases.

Логическая модель связана с формализованным описанием предметной области. Детализация описания зависит от специфики области и целей создания системы [25, 27]. Наиболее популярным способом описания логической схемы БД является ER-модель (модель «Сущность-связь»), впервые описанная Питером Ченом в 1976 году [11, 21, 23]. Для описания логической схемы в ER-модели используются следующие элементы:

- entity (сущность) — объект, отличающийся от других предметов по какому-нибудь признаку;
- attribute (атрибут) — свойство сущности;
- relationship (связь, отношение) — ассоциация между объектами.

Физическая модель БД — это схема, содержащая объекты, которые реально будут созданы в СУБД. В отличие от логической схемы, физическая содержит информацию, специфичную для конкретной системы управления. Сущности предметной области преобразуются в конкретные таблицы, атрибуты — в столбцы, связи преобразуются в соответствующие объекты [27].

Для работы с реляционными БД чаще всего используется структурированный непроцедурный язык запросов SQL [7]. Различные СУБД используют свои SQL-диалекты, расширяющие исходный язык запросов [25, 30].

1.2 Обзор существующих решений

1.2.1 Выбор продуктов для обзора

Существует достаточно много инструментов графического проектирования баз данных, их количество оценивается десятками. Для обзора выбраны 12 инструментов:

1. CA ERwin DataModelling [6]
2. Deziq for Databases [4]
3. PowerDesigner [12]
4. Enterprise Architect [5]
5. Toad Data Modeler [18]
6. Database Workbench [2]
7. Software Ideas Modeler [16].
8. Oracle SQL Developer Data Modeler [10]
9. MySQL Workbench [8]
10. Open ModelSphere [9]
11. SQL Power Architect [17]
12. DB-Main [3]

Часть из них включена ввиду широкой известности, часть выбрана из популярных бесплатных продуктов. Каждому продукту для удобства дан номер, который впоследствии будет использован в обзоре.

1.2.2 Функциональность инструментов

В результате анализа выявлены 5 элементов функциональности, которые присутствуют практически во всех инструментах проектирования БД и могут быть полезны при обучении: создание логической и физической схем БД, преобразование одной в другую, обратное проектирование (восстановление моделей по целевому коду), генерация для создания таблиц и генерация скриптов для редактирования структуры уже существующих таблиц.

Таблица 1 содержит информацию о количестве поддерживаемых инструментами СУБД и о наличии функциональности в конкретном инструменте. Зелёная ячейка — это наличие соответствующей функциональности, красная — отсутствие.

Самые популярные СУБД, поддерживаемые инструментами: MySQL Server, Oracle, DB2, Informix, PostgreSQL, MySQL, MS Access, SQLite, MS SQL Server.

Таблица 1. Сравнение функциональности инструментов.

Критерии сравнения:

1 — число поддерживаемых СУБД

2 — средства создания и логических, и физических схем

3 — средства преобразования логических схем в физические

4 — средства обратного проектирования по коду или по файлу с БД

5 — средства генерации скриптов для создания таблиц

6 — средства генерации скриптов для редактирования структуры таблиц

Продукт/Критерий	1	2	3	4	5	6
1. CA ERwin DataModelling	6-11	Green	Green	Green	Green	Green
2. Dezipn for Databases	>15	Green	Green	Green	Green	Green
3. PowerDesigner	>70	Green	Green	Green	Green	Green
4. Enterprise Architect	2	Green	Green	Green	Green	Green
5. Toad Data Modeler	>10	Green	Green	Green	Green	Green
6. Database WorkBench	8	Green	Green	Green	Green	Green
7. Software Ideas Modeler	11	Red	Red	Green	Green	Red
8. Oracle SQL Developer Data Modeler	7	Green	Green	Green	Green	Green
9. MySQL Workbench	1	Red	Red	Green	Green	Green
10. Open ModelSphere	5	Green	Green	Green	Green	Green
11. SQL Power Architect	4	Red	Red	Green	Green	Green
12. DB-Main	5	Green	Green	Green	Green	Green

1.2.3 Нефункциональные особенности инструментов

Анализ нефункциональных особенностей инструментов всегда вызывает сложности, связанные, например, с правильной расстановкой приоритетов и оценкой удобства интерфейса. Для достижения объективности было решено отталкиваться от того, что кажется наиболее подходящим для инструмента для обучающихся, а затем смотреть на то, как это выполнено в других инструментах.

Таблица 2 объединяет главные аспекты обзора:

1. Бесплатность (продукт должен быть бесплатным, чтобы им без труда можно было пользоваться на занятиях). Зеленая ячейка в таблице – продукт или одна из его версий бесплатна для некоммерческого использования; красная – продукт платный (возможно присутствие тридцатидневной пробной версии).

Для студентов ВУЗов данный пункт может быть несущественным, так как многие компании согласны предоставлять инструменты учебным заведениям бесплатно для некоммерческого использования.

2. Кроссплатформенность (должна быть возможность запустить приложение под операционной системой, которую используют на занятиях). Зеленый цвет – продукт поддерживает операционные системы Windows, Linux и Mac OS (3 наиболее популярных семейства ОС); красный – поддержка только одной ОС.
3. Продукт легковесен (обилие функциональности не должно мешать обучающимся осваивать предмет – они должны достаточно легко понимать, что и как им нужно сделать для достижения целей). Зеленый цвет – продукт легковесен; желтый – продукт обладает обширной функциональностью, связанной с базами данных (для промышленных продуктов желтый – это лучший вариант); красный – продукт обладает обширной функциональностью, не связанной с базами данных.
4. Редактирование элементов возможно из рабочей области программы. Этот пункт обоснован следующей проблемой. Во многих инструментах редактирование элементов (даже если это просто имя объекта) происходит следующим образом: нужно нажать на сущность/таблицу, открыть большое по размеру меню сущности/таблицы, которое закроет всю диаграмму, затем в меню можно работать со свойствами. При этом если пользователь забыл какую-то информацию о схеме, ему придется закрывать меню и открывать его заново, чтобы увидеть модель. Возможность вносить изменения в свойства объектов из главного окна решает проблему. Зеленый цвет – редактирование свойств объектов из главного окна возможно; красный – нет.
5. Структурированная справка, сопровождаемая большим количеством снимков экрана, доступная оффлайн (ученик должен иметь возможность быстро и наглядно получить ответ по использованию приложения даже при отсутствии доступа в Интернет). Зеленый цвет – все ограничения соблюдены; красный – справка слабо структурирована, отсутствуют снимки приложения или справка доступна только онлайн.

Во всех инструментах связь «один-к-одному» либо отсутствует в принципе, либо является вырожденным случаем связи «один-ко-многим», не имея при этом собственного графического представления на диаграммах. Для промышленных средств это обоснованное решение: отношение «один-к-одному» используется не так часто и вызывает проблемы в процессе генерации физической модели. Но если говорить об

инструменте для обучения, поддержка этой связи имеет смысл, так как можно продемонстрировать пользователям существование и такого отношения, а также его влияние на физическую схему.

Таблица 2. Нефункциональные особенности инструментов.

Критерии сравнения:

1 — бесплатность

2 — кроссплатформенность

3 — легковесность

4 — возможность редактирования из главного окна

5 — удобная справка

Продукт/Критерий	1	2	3	4	5
1. CA ERwin DataModelling	Green	Windows	Yellow	Red	Green
2. Dezhign for Databases	Red	Windows	Yellow	Red	Green
3. PowerDesigner	Red	Windows	Red	Red	Red
4. Enterprise Architect	Red	Windows, Linux, Mac	Red	Red	Green
5. Toad Data Modeler	Red	Windows	Yellow	Red	Green
6. Database WorkBench	Red	Windows, Linux and FreeBSB	Yellow	Red	Green
7. Software Ideas Modeler	Green	Windows, Linux	Red	Green	Red
8. Oracle SQL Developer Data Modeler	Green	Green	Yellow	Red	Red
9. MySQL Workbench	Green	Green	Yellow	Green	Red
10. Open ModelSphere	Green	Windows, Linux, Mac	Red	Green	Green
11. SQL Power Architect	Green	Поддерживающие JRE выше 6.0	Yellow	Red	Red
12. DB-Main	Green	Windows, Linux, Mac	Red	Red	Green

Полученные результаты обзора необходимо учитывать при разработке требований к QReal:Databases (глава 2).

1.3 Платформа QReal

На кафедре системного программирования несколько лет разрабатывается инструмент QReal, позволяющий создавать языки визуального моделирования.

Архитектуру QReal в целом можно разделить на несколько частей (см. рис. 1): ядро системы (общая часть), репозиторий с моделями и набор плагинов [29].

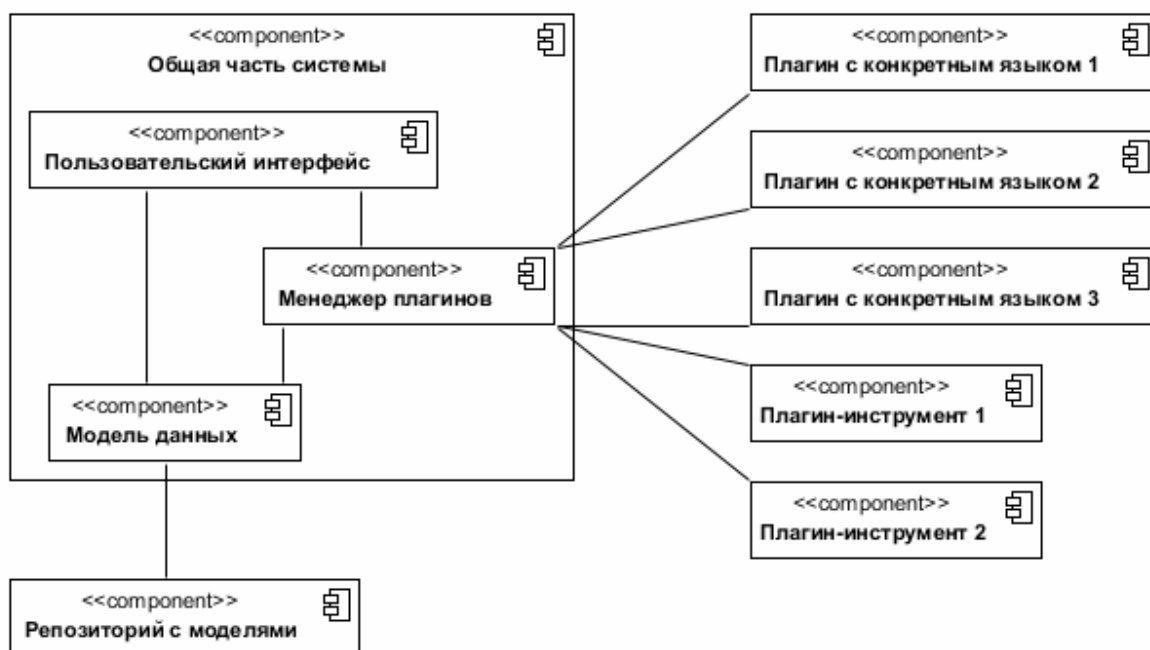


Рисунок 1. Общая архитектура QReal

В ядре системы содержится вся абстрактная функциональность инструмента, которая может быть использована в любом визуальном языке. На данный момент ядро системы поддерживает широкий спектр возможностей, начиная с базовой функциональности (возможность создания элементов и изменения через редактор свойств; соединение элементов связями; хранение объектов в репозитории и пр.) и заканчивая функциональностью, делающей взаимодействие пользователей с приложением более удобным и простым (использование жестов мышью для создания элементов; кружки элементов, ассоциированные со связями, используемые для «вытягивания» связи, и пр.).

Репозиторий QReal — объектно-ориентированная БД, предназначенная для хранения моделей. Репозиторий является динамически подгружаемой библиотекой, предоставляющей интерфейс для использования сторонними программами.

Функциональность, специфичная для какой-либо конкретной области, выносится в отдельный модуль [13].

QReal позволяет создавать плагины-редакторы, описывающие язык предметной области. Язык можно создавать с помощью метамодели, задаваемой либо в виде XML-файла определенной структуры, либо графически в метаредакторе. Кроме простого описания синтаксиса языка можно задавать отношения вложенности, подробно описывать внешнее представление элементов и связей (в том числе и в зависимости от некоторых условий), а также использовать другие дополнительные возможности, поддерживаемые ядром системы.

Помимо модулей редакторов существует возможность создания инструментальных подключаемых модулей, реализующих специфичные функциональные возможности (генерация, интерпретация, проверка корректности и пр.).

Таким образом, для создания решения на базе платформы QReal необходимо описать метамодель новых языков одним из описанных способов и при необходимости расширить функциональность платформы с помощью подключаемых модулей.

1.4 Научные работы на кафедре системного программирования СПбГУ

На тему проектирования баз данных на кафедре системного программирования математико-механического факультета СПбГУ было выполнено немало научных работ. В основном разрабатываемые инструменты в качестве ядра использовали CASE-платформу REAL. Совокупность методик работы с пакетом REAL получила название «Технология REAL-IT».

Наиболее близкими к теме данной работы являются разработки А.Н. Иванова [24] и Н.С. Нестерова [26].

В процессе работы на диссертацией «Автоматизированная генерация информационных систем, ориентированных на данные» А.Н. Иванов достиг следующих результатов: реализована модель пользовательского интерфейса информационной системы, ориентированного на взаимодействие с данными; создана методика построения модели и реализована генерация кода по ней; модель классов UML расширена с целью поддержания возможности проектирования БД; разработан диаграммный язык для описания ограничений на классы; приложение было протестировано в нескольких промышленных продуктах [24].

В процессе работы над дипломной работой «Перенос технологии REAL-IT на платформу Microsoft .Net» А.С. Нестеров достиг следующих результатов: реализована генерация графического пользовательского интерфейса для технологии REAL-IT/.Net; создан механизм редактирования кода форм, созданного автоматически в процессе генерации; реализовано приложения для демонстрации функциональности [26].

Результаты, полученные разработчиками кафедры, могут быть использованы при дальнейшем развитии проекта QReal:Databases: добавление возможности добавления и редактирования данных в БД, генерация интерфейса для взаимодействия с БД, обеспечение работы с ограничениями. На данный момент было решено включить в требования к проекту (см. гл. 2) функциональность из современных инструментов, так как обзор (см. гл.1, раздел 2) позволил выделить наиболее актуальные и популярные возможности продуктов.

В проекте REAL-IT было накоплено много ошибок, и технологии, используемые при разработке, с течением времени устарели. В связи с этим планировалось переписать части проекта с использованием Qt, но разработчики пришли к выводу, что более простым решением будет создание новой технологии. Так создан проект QReal, который, в отличие от REAL, является не конкретным CASE-инструментом, а платформой для разработки визуальных языков и инструментальных средств для них.

К моменту начала работы над QReal:Databases в QReal было несколько наработок: реализована генерация SQL-кода из физической схемы и генерация физической схемы из SQL-кода. Полноценного о проекта, посвященного приложению для работы с базами данных, в проекте еще не существовало.

1.5 Обучение базам данных в университете

В каждом университете для любого предмета составляется учебная программа, содержащая информацию о темах курса. Программы одного и того же предмета могут отличаться в разных странах и университетах, однако существуют общие рекомендации к курсам, которым необходимо следовать для достижения актуальности и полноты тем.

Для компьютерных наук существует стандарт, составляемый сообществами ACM и IEEE. Для обзора в данной работе выбран курс CS430 «Database System» описанный в стандарте 2013 года [1].

Курс «Database Systems» посвящен следующим темам:

- введение в концепцию СУБД;
- моделирование и проектирование БД;
- проектирование реляционных БД;
- языки запросов;
- хранение и индексирование;
- обработка транзакций;
- восстановление данных.

Помимо описания тем предмета стандарт предоставляет список разделов курса с суммарным временем занятий и более подробным описанием. В главе 2 представлены темы, которые охватываются разрабатываемым инструментом.

Глава 2. Разработка требований

2.1 Процесс разработки

В соответствии с классификацией требований, приведённой в [20], к инструменту QReal:Databases был составлен список требований из 7 групп.

В качестве функциональных требований выбраны возможности инструментов, рассмотренные в обзоре: создание логической и физической схем, генерация физической модели и SQL-кода, обратная генерация (см. гл. 1, раздел 2). Обучающиеся будут иметь возможность поработать с тем, чем им предстоит заниматься в будущем, если им нужно будет проектировать базы данных. В то же время более сложная функциональность (например, коллективная разработка), не являясь необходимой, скорее затруднит использование приложения.

В нефункциональных требованиях учтены замечания к продуктам-аналогам, а также добавлены пункты, необходимые для упрощения использования и поддержки приложения (например, простота добавления поддержки очередной СУБД).

Благодаря использованию платформы QReal несколько системных требований выполняется автоматически: кроссплатформенность, свободная лицензия, открытый исходный код проекта.

В системные требования включена сборка инсталлятора для Windows ввиду того, что на этой операционной системе планировалось проведение апробации. Сборка инсталлятора должна быть доступна для любой другой популярной платформы, но в нашем эксперименте на данный момент в этом нет необходимости.

2.2 Список требований

Ниже представлено 7 групп требований к проекту.

I. Бизнес-требования

Продукт должен позволять пользователям создавать и редактировать логические и физические схемы БД, а также осуществлять с ними следующие операции: преобразование физической схемы в логическую, генерация SQL-кода по физической схеме, обратное проектирование по файлу.

II. Требования пользователей

Пользователи могут с помощью системы:

- создать и отредактировать логическую схему БД;
- создать и отредактировать физическую схему БД;

- проверить корректность логической схемы БД;
- проверить корректность физической схемы БД;
- создать физическую схему из логической схемы для конкретной СУБД;
- сгенерировать по физической схеме SQL-код, создающий изображенные таблицы;
- сгенерировать по физической схеме SQL-код, добавляющий столбцы к существующим таблицам;
- спроектировать физическую схему по экземпляру БД.

III. Функциональные требования

Система должна позволять выполнять следующие операции.

- Создавать логические схемы БД на языке, основанном на модели «Сущность-связь» и состоящем из следующих элементов: сущность, атрибут, связь «один-к-одному», связь «один-ко-многим», связь «многие-ко-многим».
- Создавать физические схемы БД на языке, состоящем из следующих элементов: таблица, столбец, связь «один-ко-многим», связь «многие-ко-многим», индекс.
- Проверять корректность логической схемы: непустоту полей со свойствами элементов, корректность присоединения концов связей.
- Проверять корректность физической схемы БД: непустоту полей со свойствами элементов, корректность присоединения концов связей.
- Генерировать из логической схемы соответствующую физическую схему для СУБД Microsoft Access, MySQL 5, PostgreSQL, SQLite и Microsoft SQL Server 2008.
- Генерировать для конкретной СУБД SQL-код, выполняющий создание таблиц.
- Генерировать для конкретной СУБД SQL-код, выполняющий добавление столбцов к существующим таблицам.
- Проектировать физическую схему по экземпляру БД для SQLite и Microsoft Access.

IV. Системные требования

- Свободная лицензия.
- Кроссплатформенность.
- Открытый исходный код.
- Наличие инсталлятора для Windows.

V. Атрибуты качества

- Наличие структурированной пользовательской документации со снимками экранов приложения.

- Простота добавления поддержки новых СУБД (генерации физической модели и SQL-кода).

VI. Внешний интерфейс

- Легковесный пользовательский интерфейс, не предоставляющий доступ к функциональности, не относящейся к базам данных.
- Редактирование свойств элементов (кроме индексов) должно быть доступно из главного окна приложения.
- Редактирование свойств таблиц и столбцов должно быть доступно из отдельного меню.
- Наличие русскоязычного интерфейса.

VII. Ограничения

- Логическая схема должна быть основана на ER-модели.
- Отношение «один-к-одному» должно быть самостоятельной связью в графическом языке и иметь собственное изображение.
- Приложение должно предоставлять возможность добавления индексов к таблицам.

Учитывая функциональные требования к разрабатываемому инструменту, продукт можно будет использовать на занятиях по курсу CS430 «Database Systems» (см. гл. 1, раздел 5) в следующих темах (в скобках указаны только те темы, которые охватываются QReal:Databases):

1. проектирование баз данных — 6 часов (проектирование логической схемы реляционной БД, ER-модель, объекты в реляционной модели);
2. реляционные базы данных — 9 часов (проектирование, преобразование логической схемы в реляционную схему, функциональные зависимости, ключи, декомпозиция схемы и уточнение, нормальные формы);
3. языки запросов — 12 часов (SQL);
4. индексирование — 1.5 часов (базовая структура, индексы в SQL).

На основе представленных требований велась разработка инструмента QReal:Databases.

Глава 3. Реализация

Код проекта QReal:Databases доступен на сайте GitHub по ссылке <https://github.com/anastasia143/qreal/tree/qrealDatabases>.

3.1 Архитектура

3.1.1 Модули баз данных

QReal:Databases расширяет платформу QReal при помощи одного инструментального плагина и шести плагинов-редакторов (языки для задания физических схем для 5 диалектов SQL и язык для задания логической схемы). Архитектура инструментального модуля представлена на рис. 2.

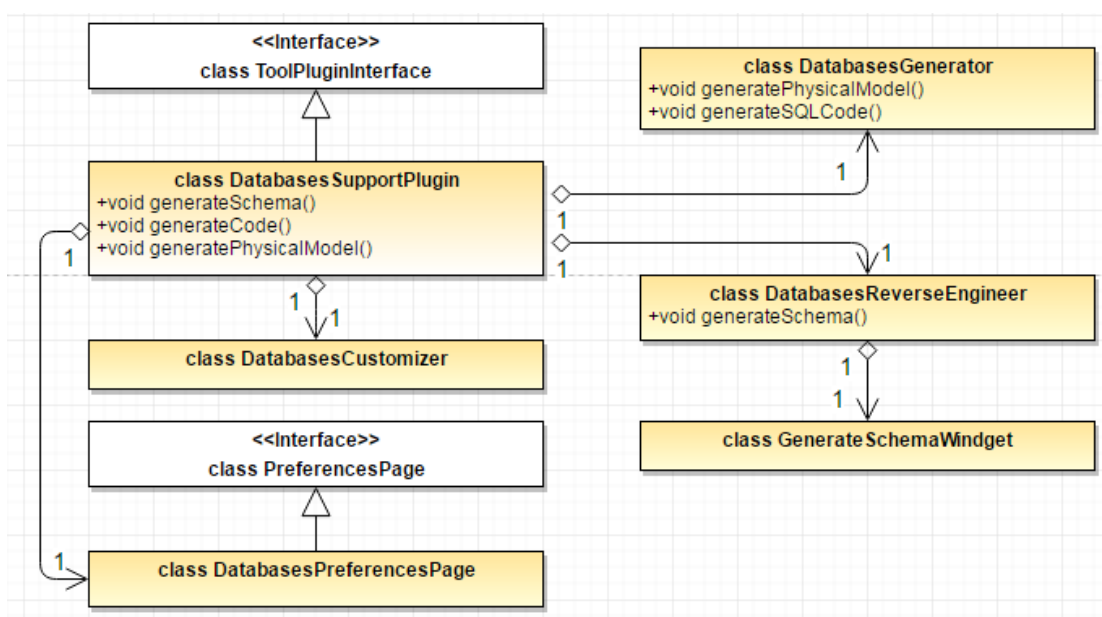


Рисунок 2. Архитектура реализованного модуля расширения

Класс `DatabasesSupportPlugin` реализует интерфейс `ToolPluginInterface`, предоставляемый платформой QReal, и является основным классом модуля. `DatabasesSupportPlugin` обеспечивает взаимодействие элементов плагина между собой и предоставляет доступ к логическому и графическому интерфейсу платформы, доступному внешним модулям. Класс `DatabasesGenerator` реализует генерацию физической модели из логической модели и генерацию SQL-кода по физической модели. Класс `DatabasesReverseEngineer` реализует функциональность обратного проектирования по файлу с СУБД. Класс `PreferencesPage` отвечает за вкладку настроек модуля баз данных в общем окне настроек QReal и позволяет взаимодействовать с ней. Класс `GenerateSchemaWidget` создает диалоговое окно для внесения пользователем информации, необходимой для работы алгоритмов обратного проектирования.

3.1.2 Класс меню таблиц

Для подключаемых модулей интерфейс QReal не предоставляет возможности создавать контекстное меню для элементов. Ввиду того, что данная функциональность на данный момент востребована только в QReal:Databases, было решено создавать меню таблиц на уровне платформы. Одним из значительных улучшений в будущем может стать реализация возможности создания подобных меню из плагинов проекта.

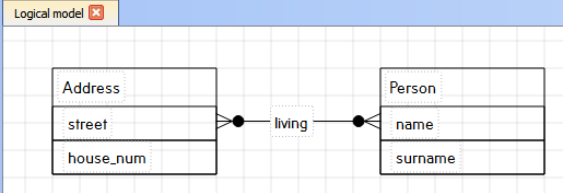
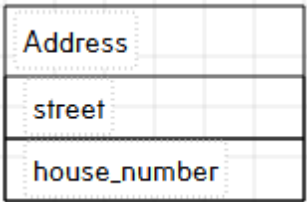
Экземпляры класса TableMenuWidget создаются в классе EditorViewScene, принадлежащем компоненте qgui и отвечающим за обработку пользовательского взаимодействия со сценой диаграммного редактора.

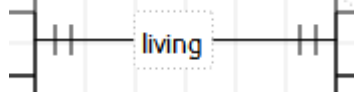
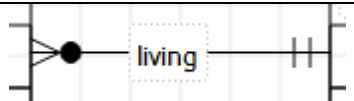
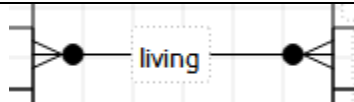
3.2 Создание графических языков

В QReal:Databases метамодель языков задаётся при помощи XML-формата.

3.2.1 Логическая схема БД

Язык логической схемы БД базируется на модели «Сущность-связь» и содержит следующие элементы.

Изображение	Описание (маркированным списком перечислены свойства элемента)
<i>Logical model (логическая модель)</i>	
 <p data-bbox="229 1361 628 1393">Фрагмент логической схемы.</p>	<p data-bbox="821 1160 1410 1236">Диаграмма логической модели БД. Содержит все остальные элементы языка.</p>
<i>Entity (сущность) и Attribute (атрибут)</i>	
 <p data-bbox="229 1720 804 1751">Сущность «Address» с двумя атрибутами.</p>	<p data-bbox="821 1451 1410 1572">Entity — абстрактный класс объектов, информацию о которых нужно хранить в БД. Содержит элементы Attribute.</p> <ul style="list-style-type: none"> <li data-bbox="874 1576 1075 1608">▪ name (имя) <p data-bbox="821 1617 1410 1738">Attribute — свойство сущности. Располагается внутри элемента, которому принадлежит.</p> <ul style="list-style-type: none"> <li data-bbox="874 1742 1075 1774">▪ name (имя) <li data-bbox="874 1783 1426 1859">▪ primary key (является ли первичным ключом) <li data-bbox="874 1868 1378 1899">▪ unique (является ли уникальным) <li data-bbox="874 1908 1378 1984">▪ not null (запрещены ли значения null) <li data-bbox="874 1993 1219 2024">▪ datatype (тип данных)

<i>One-to-one (связь «один-к-одному»)</i>	
	<p>Связь между двумя сущностями. Любому экземпляру первой сущности соответствует только один экземпляр второй и наоборот.</p> <ul style="list-style-type: none"> ▪ name (имя)
<i>One-to-many (связь «один-ко-многим»)</i>	
 <p>В левой части связи — конец «много», в правой — конец «один».</p>	<p>Связь между двумя сущностями. Экземпляру сущности с концом «много» соответствует только один экземпляр второй сущности. Экземпляру сущности с концом «один» может соответствовать несколько экземпляров другой сущности.</p> <ul style="list-style-type: none"> ▪ name (имя) ▪ column name (имя столбца, который будет сгенерирован)
<i>Many-to-many (связь «многие-ко-многим»)</i>	
	<p>Связь между двумя сущностями. Любому экземпляру первой сущности может соответствовать несколько экземпляров второй и наоборот.</p> <ul style="list-style-type: none"> ▪ name (имя) ▪ table name (имя таблицы, которая будет сгенерирована)

Тип данных атрибутов является редактируемым полем с со списком, содержащим наиболее популярные типы данных.

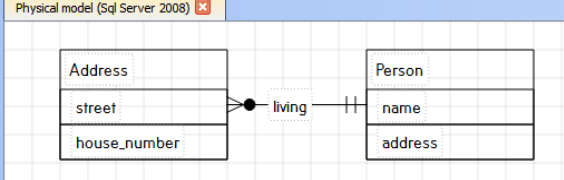
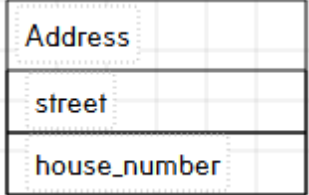
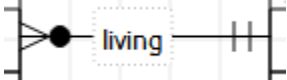
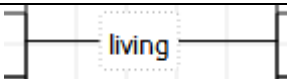
3.2.2 Физическая схема БД

Изначально при обсуждении физической модели БД обсуждался вариант создания единого языка физической схемы, который бы менялся в зависимости от СУБД. Данный путь имеет несколько недостатков. Во-первых, из-за больших различий между конкретными СУБД описание языка получилось бы громоздким и ненаглядным. Во-вторых, подобная функциональность в QReal еще не поддерживалась и потребовалось бы время на поддержку «изменяющихся» языков.

Был выбран более простой в реализации и удобный для пользователя способ. Для каждой СУБД создается отдельный язык физической схемы, который содержит элементы с именами: «Table», «Column», «One-to-many», «Many-to-many», «Index». Эти абстракции действительно присутствуют в каждом диалекте (кроме связей, т.к. они нужны для наглядности схемы), а вот свойства элементов оказываются специфичны для каждой СУБД. Таким образом, с одной стороны, язык физической схемы прост в создании и наглядно демонстрирует все особенности диалекта, но, с другой стороны, достигается

необходимая гибкость решения. Например, в ряде случаев (при генерации физической модели из логической, при обратной генерации) используются только имена элементов (таблица, столбец и пр.). Благодаря тому, что каждый язык физической модели содержит элементы с одинаковыми именами, достаточно менять имя редактора, и в результате по имени будет использован элемент с нужными свойствами диалекта.

Язык физической модели содержит следующие элементы.

Изображение	Описание (маркированным списком перечислены свойства элемента, которые присутствуют во всех диалектах)
<i>Physical model (физическая модель)</i>	
 <p>Фрагмент физической схемы.</p>	<p>Диаграмма физической модели БД. Содержит все остальные элементы языка.</p>
<i>Table (Таблица) и Column (Столбец)</i>	
 <p>Таблица «Address» с двумя столбцами.</p>	<p>Table — таблица базы данных. Содержит элементы Column</p> <ul style="list-style-type: none"> ▪ name (имя) ▪ comment (комментарий к таблице) ▪ sql-code (SQL-код)¹ <p>Column — столбец таблицы. Располагается внутри элемента, которому принадлежит.</p> <ul style="list-style-type: none"> ▪ name (имя) ▪ datatype (тип данных)
<i>One-to-many (связь «один-ко-многим»)</i>	
 <p>Связь присутствует в графической схеме для наглядности.</p>	<p>Связь между двумя таблицами. Экземпляру таблицы с концом «много» соответствует только один экземпляр второй таблицы; экземпляру таблицы с концом «один» может соответствовать несколько экземпляров другой таблицы.</p> <ul style="list-style-type: none"> ▪ name (имя)
<i>Many-to-many (связь «многие-ко-многим»)</i>	
 <p>Связь присутствует в графической схеме для наглядности.</p>	<p>Связь между двумя таблицами. Соединяет таблицу с таблицей, сгенерированной для отношения многие-ко-многим.</p> <ul style="list-style-type: none"> ▪ name (имя)

¹ Добавление из приложения SQL-кода не только для непосредственного создания таблицы является распространенной функциональностью инструментов проектирования БД.

<i>Index (Индекс)</i>	
Индексы не отображаются в графической схеме. Они создаются и редактируются через меню таблиц.	<ul style="list-style-type: none"> ▪ name (имя) ▪ table (идентификатор таблицы, которой принадлежит индекс) ▪ column names (имена столбцов, на которые распространяется индекс)

Языки физической модели созданы для следующих СУБД: Microsoft SQL Server 2008, Microsoft Access, MySQL 5, SQLite, PostgreSQL.

3.3 Генерация физической схемы из логической схемы

Алгоритм генерация физической модели из логической в QReal:Databases использует известные правила отображения ER-модели в реляционную [21]. Строго говоря, в реляционной модели отсутствует понятие «таблица», фундаментальным является понятие «отношение». Однако ввиду того, что для визуальной интерпретации отношений часто используют таблицы, в научном сообществе допускается использование данного термина, если речь идет об отношениях [21]. В контексте данной работы используется понятие «таблица», так как подразумевается не только интерпретация отношения, но и имя элемента физической схемы, который будет создан для данного отношения.

Правила, используемые алгоритмом, можно сформулировать следующим образом. Сущности становятся таблицами, атрибуты сущностей — столбцами таблиц. Для связи «многие-ко-многим» создается отдельная таблица, содержащая ключи связываемых сущностей. Для связи «один-ко-многим» к сущности с концом «много» добавляется столбец, содержащий ключ второй сущности. Связь «один-к-одному» преобразует две сущности в одну таблицу. Отметим, что для связи «один-к-одному» существуют различные варианты отображения. Был выбран один из способов, который проходят на занятиях в университете.

Для обработки связи «один-к-одному» логическая схема рассматривается как граф, узлы которого — сущности, а ребра — связи «один-к-одному». В графе поиском в глубину выделяются компоненты связности, для которых будет создана общая таблица.

В качестве примера рассмотрим диаграмму на рис. 3. Данной логической схеме соответствует следующий набор компонент: ABD, CF, E.

В процессе обработки может получиться так, что две компоненты будут связаны друг с другом более чем одной связью (рис. 4). Данная ситуация может быть корректной с точки зрения предметной области, а может быть следствием ошибки проектирования.

Информацию о случаях, когда компоненты связаны несколькими различными связями, было решено выводить пользователю.

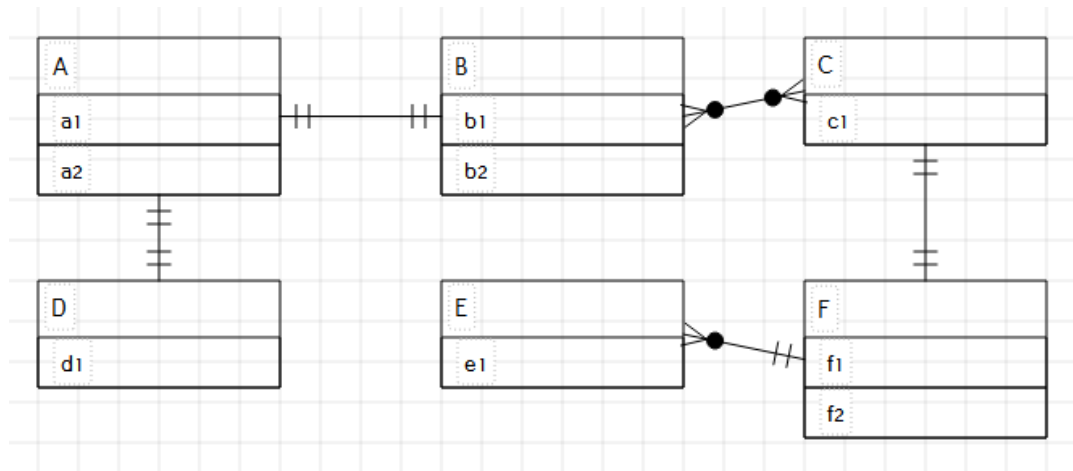


Рисунок 3. Пример логической схемы БД, созданной с помощью QReal:Databases

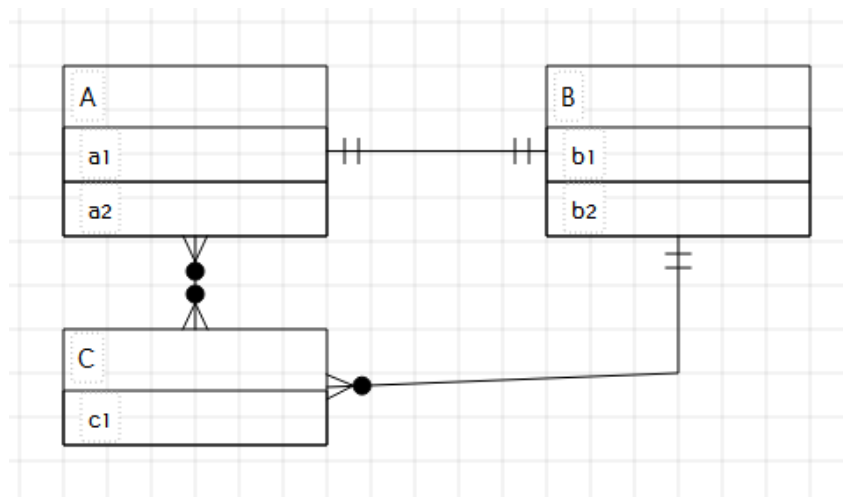


Рисунок 4. Пример логической схемы БД, созданной с помощью QReal:Databases, в которой компоненты АВ и С связаны двумя разными связями

Для обработки связей «один-ко-многим» и «многие-ко-многим» логическая схема рассматривается как граф, где узлы — компоненты связности, а ребра — отношения, отличные от «один-к-одному». Для графа создается изначально пустая матрица смежности, которая заполняется в процессе просмотра связей схемы. Если элемент компоненты i связан с элементом компоненты j связью «один-ко-многим», это значит, что вся компонента i связана с компонентой j этой связью. В таком случае, если ячейка пуста, при обработке данной связи в матрице в столбце i и строке j будет поставлен идентификатор. Если ячейка не пуста, это означает, что между компонентами уже есть связь, поэтому пользователю будет выведено соответствующее отношение.

Алгоритм

Алгоритм генерации физической схемы выглядит следующим образом.

1. Происходит проверка элементарных правил корректности (присоединены ли к элементам с двух концов; указаны ли имена элементов).
2. Отдельные сущности (без связей) преобразуются в таблицу, атрибуты сущности преобразуются в столбцы новой таблицы.
3. Для связанных сущностей определяется компонента связности.
4. Каждая компонента связности преобразуется в таблицу, атрибуты всех сущностей компоненты становятся столбцами новой таблицы.
5. Происходит обработка связей «один-ко-многим» и «многие-ко-многим»: для один-ко-многим к таблице компоненты с концом «много» добавляется столбец с первичным ключом второй компоненты; для «многие-ко-многим» создается таблица со столбцами, содержащими ключи связываемых компонент.

Взаимодействие с пользователем

Для запуска генерации физической модели в настройках приложения пользователю необходимо указать имя СУБД. Окно настроек реализовано в классе DatabasesPreferencesPage.

3.4 Генерация SQL-кода

Генерация SQL-кода доступна в двух режимах: создание таблиц и редактирование таблиц. Для каждого SQL-диалекта реализуется по две функции, соответствующие этим режимам. Генерация доступна для следующих СУБД: Microsoft SQL Server 2008, Microsoft Access, MySQL5, SQLite, PostgreSQL.

Алгоритм

Для каждой таблицы создается соответствующий скрипт вида «CREATE TABLE имя_таблицы ()» или «ALTER TABLE имя_таблицы», учитывающий свойства таблиц конкретного диалекта. Для каждого столбца создается скрипт вида «имя_столбца тип_столбца» или «ADD COLUMN имя_столбца ()», учитывающий свойства столбцов диалекта.

Взаимодействие с пользователем

Для запуска функциональности в настройках приложения пользователю необходимо указать имя СУБД для генерации SQL-кода и файл, в который код будет сохранен.

3.5 Обратная генерация

Класс `DatabasesReverseEngineer` инкапсулирует функциональность, относящуюся к генерации физической модели по экземпляру БД. Обратная генерация доступна для СУБД SQLite и Microsoft Access.

Алгоритм

Для реализации используются механизмы Qt — работа с классами `QSqlDatabase`, `QSqlRecord`, `QSqlField`. Алгоритм выглядит следующим образом.

1. В экземпляр класса `QSqlDatabase` передается тип СУБД и путь к файлу, создается соответствующая физическая схема.
2. У экземпляра запрашиваются имена всех содержащихся таблиц.
3. Для каждой таблицы:
 - создается элемент «Table» физической схемы с соответствующим именем;
 - сохраняется список столбцов таблицы в экземпляр класса `QSqlRecord`;
 - при помощи `QSqlField` осуществляется доступ к отдельному столбцу и создается элемент «Column» физической модели с соответствующими свойствами.

Взаимодействие с пользователем

Для взаимодействия с пользователем используется экземпляр класса `GenerateSchemaWidget`. Форма класса позволяет указывать тип СУБД и путь к файлу, по которому будет осуществляться обратное проектирование. Пользователь вызывает функциональность, заполняет форму, после этого запускается генерация.

3.6 Создание меню для работы с элементом «Таблица»

Изначально подразумевалось, что редактирование свойств элементов языков будет осуществляться через стандартный редактор `QReal`. Данный способ удобен тем, окно редактирования располагается в стороне от диаграммы, не перекрывая ее. Таким образом, переключение между объектами происходит достаточно просто. К тому же элементы визуально остаются частью диаграммы, поэтому проще оценивать таблицу как часть базы данных, а не отдельно.

Однако в процессе работы над проектом стало понятно, что некоторые свойства элемента «Таблица» (индексы таблицы, SQL-запросы к таблице) было бы удобнее обрабатывать с помощью отдельного меню. В связи с этим было решено предоставить

пользователю возможность работы с таблицей и ее столбцами не только через редактор свойств, но и через меню таблицы.

Меню таблицы должно быть похоже для всех СУБД, но в то же время позволять редактировать специфичные свойства диалекта. Для обеспечения гибкости для меню был реализован класс, содержащий форму со вкладками, часть из которых меняет вид содержимого в зависимости от СУБД. Общий стиль внешнего вида неизменчив.

Далее представлено описание вкладок меню.

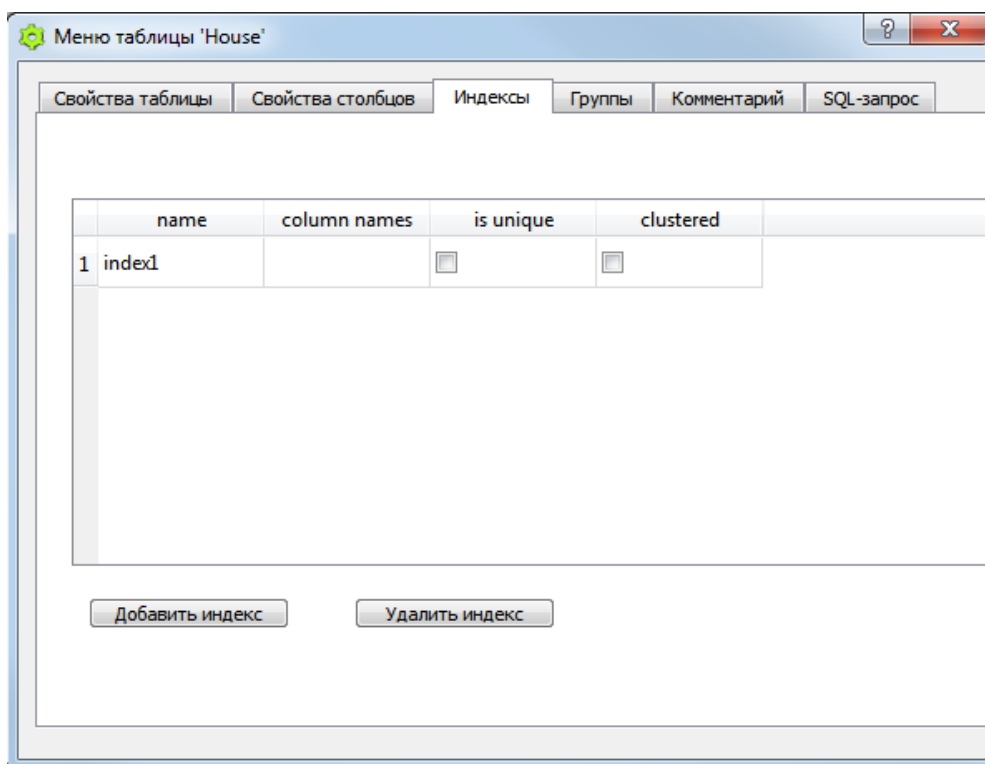


Рисунок 5. Фрагмент меню для элемента «Таблица»

3.6.1 Свойства таблицы

Данная вкладка позволяет редактировать свойства элемента «Таблица» с помощью экземпляра QTableWidgetItem (стандартные таблицы форм в Qt). Строки таблицы — свойства элемента «Таблица» для конкретного диалекта, единственный столбец — значение свойства. Каждое свойство строго связано с номером строки, поэтому при редактировании пользователем ячейки таблицы свойств просто определить свойство, которое подлежит изменению. Соответствующие обновления свойств элемента «Таблица» производятся в репозитории.

3.6.2 Свойства столбцов

Данная вкладка позволяет редактировать свойства столбцов, принадлежащих таблице, с помощью экземпляра QTableWidgetItem. Строки таблицы — столбцы элемента «Таблица». Столбцы — свойства элемента «Столбец», набор которых формируется в

зависимости от диалекта. Для идентификации элементов «Столбец» с реальными объектами диаграммы в таблице хранится невидимый пользователю столбец с индексом элемента в репозитории.

Каждый столбец связан с определенным свойством, строка — с определенным элементом «Столбец», поэтому по номеру строки и столбца ячейки можно определить элемент и свойство для изменения. Соответствующие обновления свойств элементов производятся в репозитории.

3.6.3 Создание и редактирование индексов

Таблица работы с индексами реализована аналогично таблице для столбцов: элемент `QTableWidget`, строки — индексы, столбцы — свойства, специфичные для диалекта. Для индексов добавлена возможность добавления и удаления прямо из меню. При нажатии кнопки «добавить индекс» к таблице `QTableWidget` добавляется новая строка, а в репозитории создается соответствующий элемент. При нажатии «удалить индекс» с выделенной строкой удаляется соответствующий индекс из таблицы и из репозитория.

3.6.4 Комментарии к таблице

Содержит текстовое поле для добавления комментариев к скрипту, соответствующему элементу «Таблица».

3.6.5 SQL-запросы к таблице

Содержит текстовое поле для добавления SQL-запросов к элементу «Таблица».

Взаимодействие с пользователем

Для работы с меню пользователю необходимо кликнуть правой кнопкой мыши на элемент «Таблица» на диаграмме и выбрать пункт «Меню таблицы». После этого появится модальное окно, предоставляющее вышеописанную функциональность.

После реализации функциональности приложения осуществлялось приближение полученного прототипа к программному продукту.

Глава 4. Подготовка продукта

4.1 Пользовательская документация

Для приложения была создана подробная пользовательская документация с большим количеством снимков экрана, охватывающая всю функциональность приложения. Документация доступна в виде связанных html-страниц по вызову пункта меню «Помощь» из главного окна приложения.

Структура документации содержит разделы.

1. Пользовательский интерфейс: описывает интерфейс приложения в целом. Содержит подразделы: главное меню, диалог настроек.
2. Моделирование: дает пользователю представление о создании и редактировании диаграмм. Содержит подразделы: создание диаграмм, создание элементов и связей, редактирование свойств элементов.
3. Работа с базами данных: описывает способы работы с диаграммами баз данных через приложение. Содержит подразделы: проверка диаграмм на корректность, генерация физической схемы БД из логической, генерация SQL-кода из физической схемы, обратное проектирование.
4. Описание языков: содержит описание элементов языков, с помощью которых можно создавать диаграммы. Содержит подразделы: язык логической схемы БД, язык физической схемы БД.

Документация проекта доступна на GitHub по ссылке

<https://github.com/anastasia143/qreal/tree/qrealDatabases/plugins/databases/doc/help>.

4.2 Локализация системы

По умолчанию интерфейс в QReal создается на английском языке, с помощью инструмента Qt Linguist была осуществлена также поддержка русского языка.

4.3 Сборка инсталлятора

Для создания инсталляторов в QReal используется Qt Installer Framework. Перед сборкой инсталлятора система должна быть организована в иерархичную структуру специального вида, принятую в QReal. Структура состоит из компонент, общих для любого проекта, и из компонент, специфичных для конкретного плагина [13].

Для QReal:Databases были написаны скрипты, организовывающие правильную иерархическую структуру проекта и запускающие процесс инсталляции. Для проведения апробации собран инсталлятор для Windows.

Глава 5. Апробация

5.1 Цели апробации и выбор целевой группы

Главная цель апробации QReal:Databases — выявление недостатков приложения для дальнейшего исправления. В первую очередь интересно было посмотреть на замечания по удобству использования, так как комфорт работы с приложением сложно оценивать без пользователей; а также найти ошибки, возникшие в ходе реализации решения. Другими целями исследования стали: оценка интереса пользователя в функциональности, которая не будет охвачена апробацией; поиск направлений для дальнейшего развития; выявление положительных качеств инструмента; оценка простоты поиска ошибок в работах учеников преподавателем.

Для участия в апробации выбраны 10 человек в возрасте 18-21, на момент апробации являющимися начинающими в проектировании баз данных. Из 10 человек 9 еженедельно посещали вводный курс «Базы данных и СУБД».

5.2 Проведение апробации

Перед апробацией была проведена презентация о возможностях приложения. По результатам опроса инструкции были понятны всем участникам. Далее было выполнено задание, в котором нужно было создать логическую схему БД, сгенерировать по ней физическую схему для Microsoft SQL Server 2008, внести изменения в физическую модель и сгенерировать SQL-код. Предлагаемую базу данных можно найти в приложении. Среднее время работы над заданием составило 44 минуты. Минимальное время — 20 минут, максимальное — 70 минут.

После апробации участникам было предложено заполнить анкету.

5.3 Результаты апробации

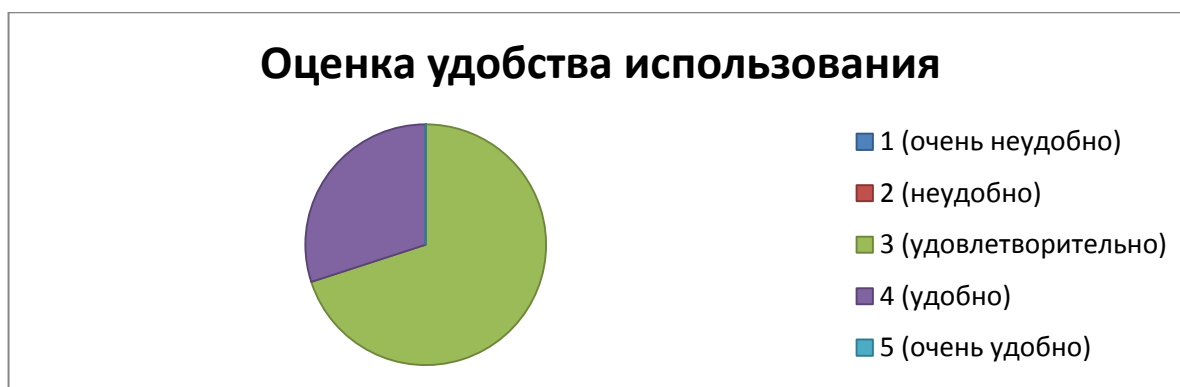


Рисунок 6. Оценка удобства использования приложения

Оценка удобства использования в среднем составила 3.3 балла по шкале от 1 до 5, где 1 — очень неудобно, 5 — очень удобно (рис. 6).

Главной причиной неудобства работы с приложением можно назвать большое количество замечаний к функциональности и взаимодействию с пользователем (рис. 7).

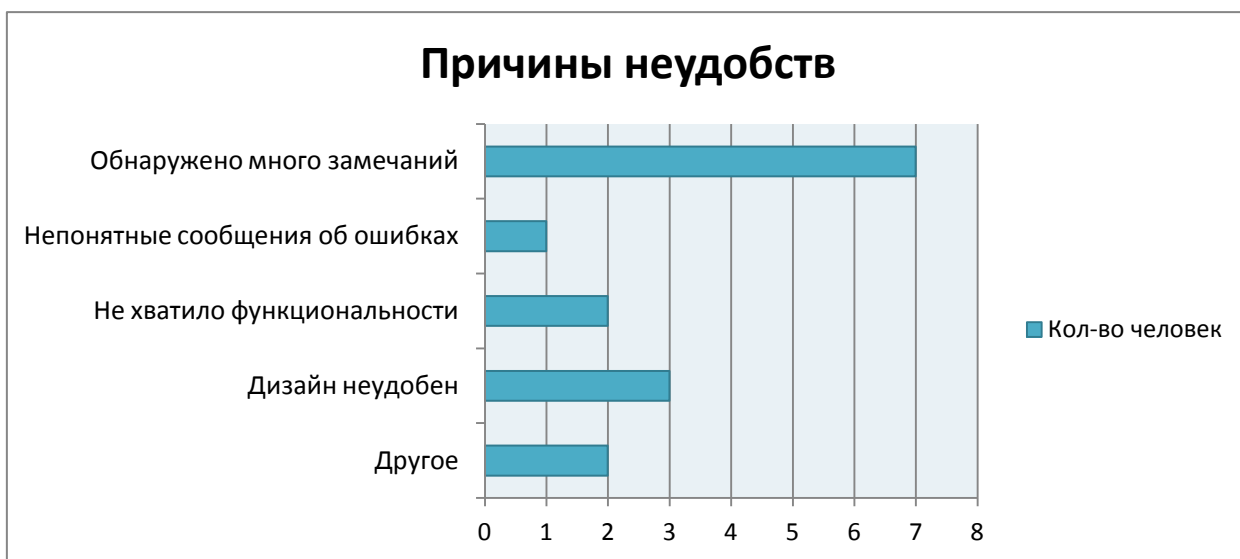


Рисунок 7. Неудобства, отмеченные пользователями

Замечания пользователей были классифицированы по области работы с приложением и по принадлежности к модулям QReal:Databases или к базовой платформе QReal. Замечания также разделены на замечания, не являющиеся ошибками, и ошибки. Среди ошибок были выделены критические.

Больше всего замечаний относилось к графическому интерфейсу (рис. 8).

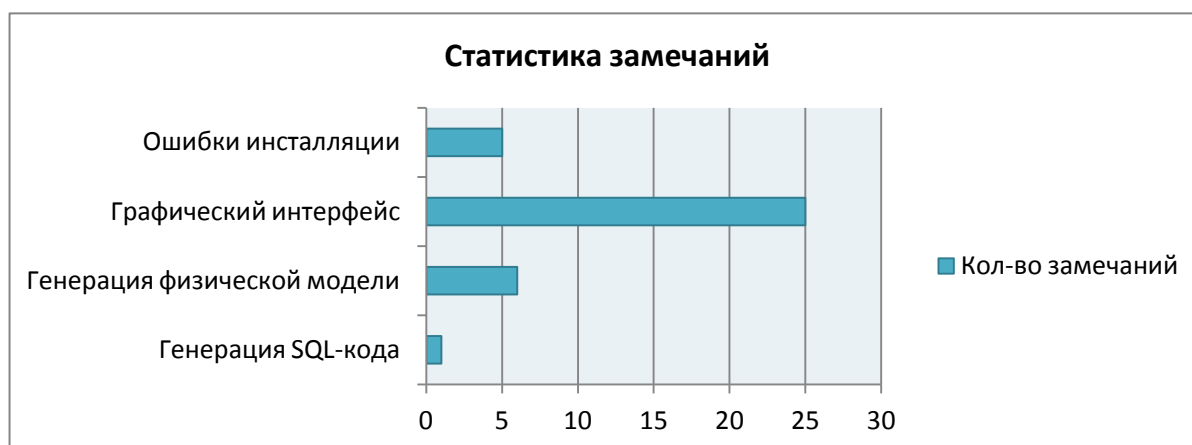


Рисунок 8. Классификация замечаний по точке взаимодействия с приложением

Более подробная статистика выглядит следующим образом:

1. Инсталляция:

- модули QReal:Databases: 4 ошибки (их них одна критическая) и одно замечание.

2. Графический интерфейс:
 - модули QReal:Databases: одна ошибка и 11 замечаний;
 - платформа: 11 ошибок (из них две критических) и 5 замечаний.
3. Генерация физической модели:
 - модули QReal:Databases: три ошибки и три замечания.
4. Генерация SQL-кода:
 - модули QReal:Databases: два замечания.

Пользователи в процессе апробации взаимодействовали с инструментами создания логической и физической схем БД, генерации физической модели и генерации SQL-кода. По результатам опроса пользователи выразили желание поработать и с другой функциональностью системы (рис. 9).

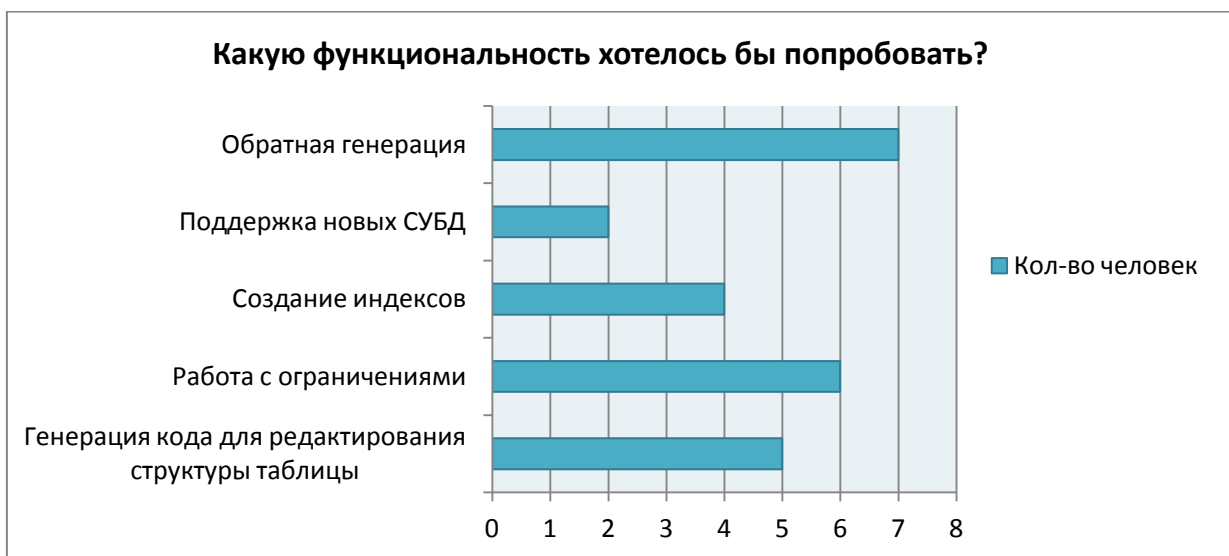


Рисунок 9. Отмеченная пользователями желаемая функциональность

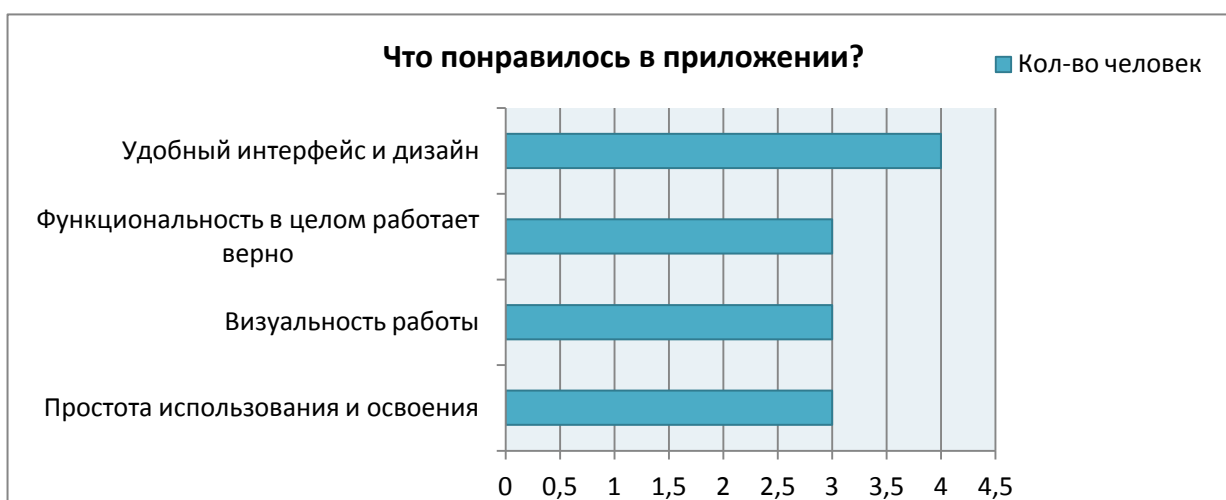


Рисунок 10. Отмеченные пользователями положительные качества приложения

Несмотря на большое количество замечаний, участники апробации отметили удобство интерфейса и простоту освоения приложения (рис. 10).

Также хотелось бы отметить, что благодаря визуальной работе со схемами проверяющему было просто анализировать работы обучающихся. У студентов были замечены ошибки проектирования (лишние зависимости, логически неправильные решения), выявлено непонимание некоторых моментов из теории баз данных (например, непонимание влияния типа связи на физическую модель).

5.4 Доработка по результатам апробации

В процессе доработки по результатам апробации обработаны следующие замечания и ошибки:

1. Инсталляция
 - модули QReal:Databases: исправлены три ошибки
2. Графический интерфейс
 - модули QReal:Databases: исправлены одна ошибка и три замечания
 - платформа: 6 ошибок внесены в отчет об ошибках платформы QReal
3. Генерация физической модели
 - модули QReal:Databases: исправлены три ошибки и два замечания
4. Генерация SQL-кода
 - модули QReal:Databases: исправлены два замечания

Заключение

В результате работы над данным проектом решены следующие задачи:

- разработаны требования к системе графического проектирования БД для обучающихся;
- реализована среда моделирования QReal:Databases, позволяющая обучающимся работать с логическими схемами БД и физическими схемами для разных СУБД: создавать и редактировать схемы, осуществлять генерацию физической схемы по логической, генерацию SQL-кода для создания и редактирования таблиц, обратную генерацию по экземпляру БД;
- создана локализация системы для русского и английского языков, создана пользовательская документация, собран инсталлятор для Windows;
- произведена апробация приложения на целевой группе, по результатам апробации в систему внесён ряд изменений.

Направления дальнейших исследований:

- проверка правильности построения диаграмм по некоторым критериям, указанным преподавателем;
- проведение более глубокого исследования удобства использования приложения, сравнение с инструментами, представленными в обзоре;
- расширение возможностей работы с редактированием структуры таблиц;
- поддержка возможности создания меню элементов на уровне ядра системы;
- поддержка миграции БД из одной СУБД в другую;
- реализация удобного способа расположения графических элементов в процессе генерации физической схемы;
- добавление горячих клавиш для создания элементов диаграммы;
- поддержка нескольких способов обработки отношения «один-к-одному».

Список литературы

1. Computer Science Curricula 2013. Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. / The Joint Task Force on Computing Curricula. Association for Computing Machinery (ACM), IEEE Computer Society — URL: <https://www.acm.org/education/CS2013-final-report.pdf> (дата обращения: 15.05.2016).
2. Database Workbench. Официальный сайт Upscene [Электронный ресурс] — URL: http://www.upscene.com/database_workbench/ (дата обращения: 01.10.2015).
3. DB-Main. Официальный сайт продукта [Электронный ресурс] — URL: <http://www.db-main.eu/> (дата обращения: 01.04.2016).
4. Dezign for Databases, официальный сайт Datanamic Solutions BV [Электронный ресурс] — URL: <http://www.datanamic.com/company/dezign-for-databases-v72-released.html> (дата обращения: 01.04.2016).
5. Enterprise Architect 12, официальный сайт Sparx Systems [Электронный ресурс] — URL: <http://www.sparxsystems.com.au/products/ea/12/index.html> (дата обращения: 01.10.2015).
6. Erwin, официальный сайт продукта [Электронный ресурс] — URL: <http://erwin.com/> (дата обращения: 01.04.2016).
7. Information technology. Database languages. SQL: ISO/IEC 9075:2011 / ANSI, ISO, IEC — 2011. — 1434 с.
8. MySQL Workbench, официальный сайт продукта [Электронный ресурс] — URL: <https://www.mysql.com/products/workbench/> (дата обращения: 01.10.2015).
9. Open ModelSphere, официальный сайт продукта [Электронный ресурс] — URL: <http://www.modelsphere.com/org/> (дата обращения: 01.10.2015).
10. Oracle SQL Developer Data Modeler, официальный сайт Oracle [Электронный ресурс] — URL: <http://www.oracle.com/technetwork/developer-tools/datamodeler/overview/index.html> (дата обращения: 01.04.2016).
11. Peter Pin-Shan Chen. The Entity-Relationship Model — Toward a Unified View of Data // ACM Transactions on Database Systems (TODS) : Сб. — Нью-Йорк: ACM, 1976. — Т. 1. — 36 с.
12. Power Designer, официальный сайт Sybase [Электронный ресурс] — URL: <http://www.sybase.ru/products/powerdesigner> (дата обращения: 01.04.2016).
13. QReal, документация проекта на GitHub [Электронный ресурс] — URL: <https://github.com/qreal/qreal/wiki/> (дата обращения: 15.05.2016).

14. QReal, официальный сайт проекта [Электронный ресурс] – URL: <http://qreal.ru> (дата обращения: 15.05.2016).
15. Ramakrishnan, R., Gehrke, J. Database management systems. — Сингапур: McGraw-Hill, 2003. — 1098 с.
16. Software Ideas Modeler, официальный сайт продукта [Электронный ресурс] — URL: <https://www.softwareideas.net/> (дата обращения: 01.04.2016).
17. SQL Power Architect, официальный сайт продукта [Электронный ресурс] — URL: <http://www.sqlpower.ca/page/architect> (дата обращения: 01.04.2016).
18. Toad Data Modeler, официальный сайт Dell Inc. [Электронный ресурс] — URL: <http://software.dell.com/products/toad-data-modeler/> (дата обращения: 01.04.2016).
19. Бен-Ган И. Microsoft SQL Server 2008. Основы T-SQL: Пер. с англ. — СПб: БХВ-Петербург, 2009. — 432 с.
20. Вигерс К., Битти Д. Разработка требований к программному: Пер. с англ. — СПб: БХВ-Петербург, 2004. — 576 с.
21. Гарсиа-Молина Г., Ульман Д.Д., Уидом Д.. Системы баз данных. Полный курс. : Пер. с англ. — Москва: Вильямс, 2003. — 1088 с.
22. Диго С.М. Базы данных. Проектирование и создание : Учебно-методический комплекс. – Москва: Изд. центр ЕАОИ. 2008. – 171 с.
23. Димов Э.М., Диязитдинова А.Р., Качков Д.А. Проектирование информационных систем: Учебное пособие — Самара: ПГАТИ, 2003 — 78 с.
24. Иванов А.Н. Автоматизированная генерация информационных систем, ориентированных на данные. — Санкт-Петербург, 2005. — 130 с.
25. Кузнецов М., Симдянов И. MySQL 5. — СПб: БХВ-Петербург, 2010. — 1024 с.
26. Нестеров А.В. Перенос технологии REAL-IT на платформу Microsoft .Net. — Санкт-Петербург, 2007. — 21 с.
27. Пушников А.Ю. Введение в системы управления базами данных. Часть 1. Реляционная модель данных: Учебное пособие / Изд-е Башкирского ун-та. - Уфа, 1999. - 108 с.
28. Семенова А.В. Создание системы проектирования БД на базе платформы QReal. – Курсовая работа 3 курса, Санкт-Петербургский государственный университет. — 2015. – URL: <http://se.math.spbu.ru/SE/YearlyProjects/2015/YearlyProjects/2015/344/344-Semenova-report.pdf> (дата обращения: 03.12.2015).
29. Терехов А.Н., Брыксин Т.А., Литвинов Ю.В. QReal: платформа визуального предметно-ориентированного моделирования // Программная инженерия, №6. — 2013. — с. 11-19

30. Хоменко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник для высших учебных заведений. — 6-е изд., доп. — СПб: КОРОНА-Век, 2009. — 726 с.

Приложение

Задания апробации

1. Создайте логическую схему, содержащую перечисленные сущности и атрибуты. Соедините сущности корректными связями.
2. Сгенерируйте по логической схеме физическую схему. Внесите необходимые изменения (свойства и названия таблиц и столбцов).
3. Сгенерируйте по физической схеме SQL-код и запустите его в СУБД. Исправьте скрипт при наличии ошибок.

Список сущностей, атрибуты находятся в скобках:

Ректор (имя, фамилия, отчество)

Университет (cid, название)

Кафедра (cid, название, номер_кабинета)

Преподаватель (cid, имя, фамилия, отчество)

Студент (номер_зачетки, имя, фамилия)

Группа (номер_группы, курс)

Предмет (cid, название)

На рисунках 1 и 2 представлены варианты выполнения первых двух заданий.

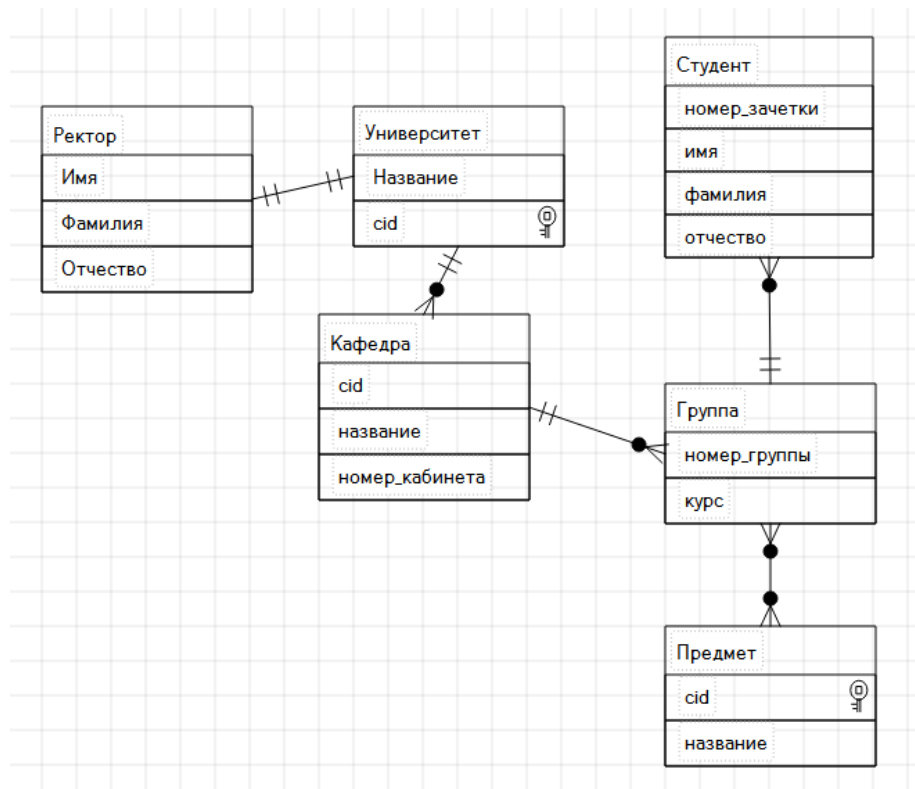


Рисунок 1. Логическая схема БД, созданная в QReal:Databases

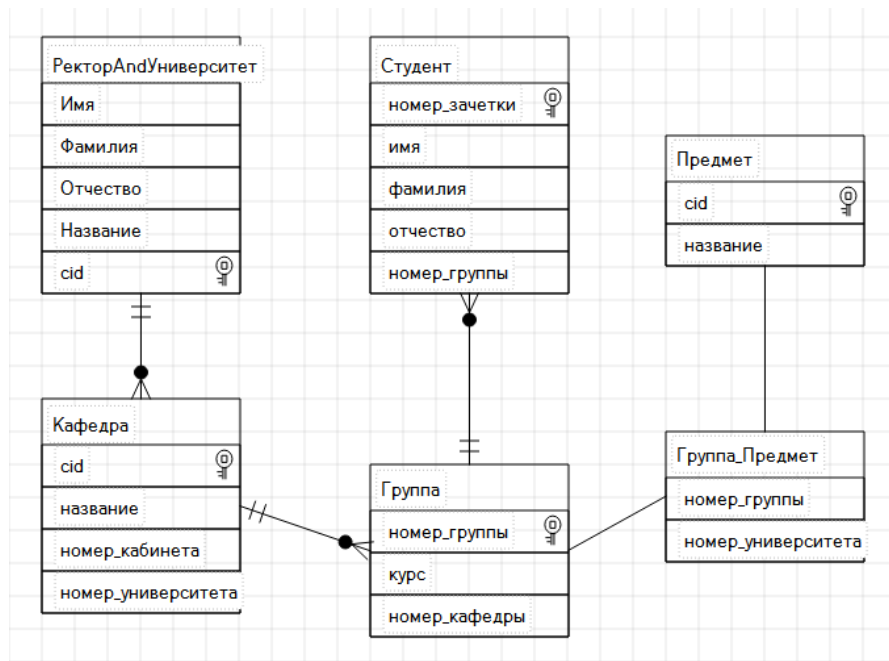


Рисунок 2. Физическая схема БД, сгенерированная по логической модели на рис. 1

По физической схеме на рис.2 сгенерирован следующий SQL-код:

```
CREATE TABLE РекторAndУниверситет
(
Имя NVARCHAR(30),
Фамилия NVARCHAR(30) NOT NULL,
Отчество NVARCHAR(30) NULL,
Название NVARCHAR(30) NOT NULL,
cid INT PRIMARY KEY
);
CREATE TABLE Кафедра
(
cid INT PRIMARY KEY,
название INT,
номер_кабинета INT,
номер_университета INT NOT NULL
);
```

```
CREATE TABLE Студент
(
номер_зачетки INT PRIMARY KEY,
имя NVARCHAR(30) NOT NULL,
фамилия INT NOT NULL,
отчество NVARCHAR(30) NULL,
номер_группы INT NOT NULL
);

CREATE TABLE Группа
(
номер_группы INT PRIMARY KEY,
курс INT NOT NULL,
номер_кафедры INT NULL
);

CREATE TABLE Предмет
(
cid INT PRIMARY KEY,
название NVARCHAR(30) NOT NULL
);

CREATE TABLE Группа_Предмет
(
номер_группы INT NOT NULL,
номер_университета INT NOT NULL
);
```