Санкт-Петербургский государственный университет

Математико-механический факультет

Кафедра системного программирования

Создание системы проектирования БД на базе платформы QReal

Курсовая работа студентки 344 группы

Семеновой Анастасии Владимировны

Научный руководитель: ст. преп. Брыксин Т.А.

Санкт-Петербург

2015

Оглавление

1. Введение

1.1 Цели и задачи работы

Базы данных в настоящее время используются повсеместно, причем самые популярные СУБД - реляционные, а для работы с ними используются диалекты языка SQL. Поэтому достаточно важно делать процесс создания БД как можно более быстрым и удобным.

Удачным решением для упрощения проектирования БД является представление баз данных в виде наглядной графической ER-модели: пользователь оперирует различными сущностями со свойствами и может устанавливать связи между сущностями. А программа, в свою очередь, предлагает функции для работы с описанной графической моделью, основная из которых - генерация SQL-кода для различных систем управления базами данных. Системы управления, как правило, вносят свои изменения в язык, поэтому SQL специфичен для каждой СУБД. Также было бы удобно, если бы перед генерацией кода из логической схемы БД (сущности, атрибуты и связи) пользователь мог бы увидеть, какие таблицы будут созданы в БД физически. Полученная схема называется физической моделью базы данных.

На кафедре системного программирования СПбГУ существует проект QReal с открытым исходным кодом, который позволяет быстро создавать визуальные языки программирования и инструменты для них. На основе QReal уже создано большое количество визуальных языков. Существовали и наработки, связанные с базами данных. Так как моделирование баз данных на сегодняшний день - востребованная область, хотелось бы создать полноценный продукт для работы с базами данных в учебных целях, чтобы студенты могли пользоваться бесплатным инструментом, ориентированным на улучшение понимания процесса проектирования БД. Было решено создать проект "QReal:Databases", в котором был бы реализован необходимый функционал. Инструмент должен быть легковесным, но предоставляющим необходимые для обучения возможности.

Программных продуктов, предоставляющих возможность проектировать базы данных с помощью графической модели, достаточно много. Многие инструменты поддерживают большое количество возможностей и используются в коммерческой разработке баз данных. Обычно эти продукты платные и ориентированы на промышленность. Перед созданием собственного ПО необходимо сделать обзор на несколько инструментов и проанализировать результаты в рамках текущей задачи,.

Таким образом, появляется цель моей работы: проанализировать существующие решения; выявить их достоинства и недостатки; создать собственное ПО, предоставляющее возможности для работы с базами данных в учебных целях.

Работа над данным проектом рассчитана на 2 года. В первую очередь необходимо реализовать основной функционал программы, поэтому задачи первого года следующие:

- выявить и проанализировать достоинства и недостатки нескольких существующих систем для проектирования БД;
- создать графический язык на основе ER-модели для логической схемы БД;
- создать графический язык для физической схемы БД;
- реализовать генерацию физической модели БД из логической модели;
- реализовать генерацию SQL-кода из физической модели (создание таблиц);
- реализовать проверку диаграмм на корректность;
- обеспечить поддержку нескольких различных СУБД;
- произвести апробацию функциональности на нескольких примерах.

2. Обзор

2.1 Основные понятия и определения

I. Инструмент QReal

- Средство QReal - проект кафедры Системного Программирования СПбГУ с открытым исходным кодом, предназначенный для быстрого и удобного создания визуальных языков программирования[1].
- Плагин в QReal - подключаемый модуль, расширяющий функциональность платформы[2].
- Метамодель языка - модель, описывающая элементы языка предметной области[2].

II. Базы данных и системы управления базами данных

- База данных (БД) - совокупность специальным образом организованных данных, которые хранятся в памяти вычислительной системы и отображают состояние объектов и взаимосвязей между ними в выбранной предметной области [6].
- Система управления базами данных (СУБД) - комплекс языковых и программных средств, предназначенный для создания и использования БД многими пользователями [6].
- Модель "сущность-связь" (ER-модель данных) - модель данных, позволяющая описывать логические схемы выбранной предметной области. Была предложена в 1976 г. Питером Пин-Шэн Ченом [7].

  Элементы модели [2]:
  - сущность (entity) - предмет, который может быть идентифицирован по какому-либо признаку, отличающему его от других предметов;
  - связь (relationship)- это ассоциация между сущностями;
  - атрибут (attribute) - свойства, характеризующие сущность [2].
- Диаграмма сущность-связь - графическое представление модели сущность-связь. Существует множество различных нотаций [2].
- Концептуальная (логическая модель) базы данных - схема БД в виде ER-модели [5].

[1] Проект QReal, url: http://qreal.ru (дата обращения 17.05.2015)

[2] Документация проекта QReal, url: https://github.com/qreal/qreal/wiki/Плагины (дата обращения 17.05.2015)

- Физическая модель базы данных - описание структур БД, которые будут созданы в специальной среде разработки. Физическая модель строится на основе логической [6].
- SQL - непроцедурный язык программирования, используемый для работы с данными в реляционных БД [6] [9].

2.2 Обзор существующих решений

Для обзора в курсовой работе выбрана часть наиболее известных инструментов для проектирования БД. Для наглядности обзор представляет из себя перечисление достоинств и недостатков ПО, выявленных как в процессе анализа информации, предоставляемой компаниями о своем продукте, так и на основе личного опыта работы с приложениями.

1. **CA ERwin DataModelling** - среда моделирования компании Computer Associates с возможностью коллективной работы. Существует несколько версий программы, каждая из которых направлена на определенную специфику работы с ERwin[3].
Для анализа был выбран продукт CA ERwin Data Modeler Community Edition.

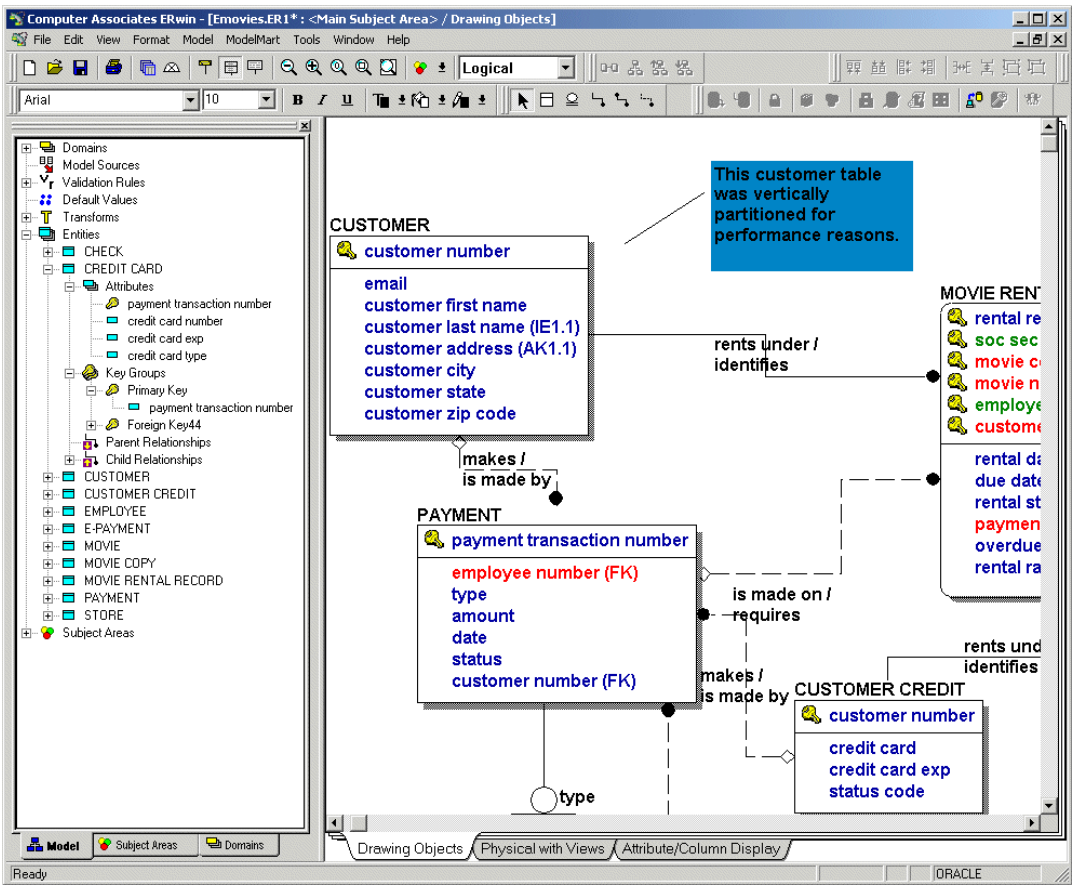


Рисунок 1: инструмент CA ERwin DataModelling

Достоинства:

- возможность создания физической, логической или логико-физической модели (можно выбрать удобное представление данных);

[3] Продукт Erwin, url: http://erwin.com/ (дата обращения 17.05.2015)

- возможность совместной работы на основе моделей и репозиториев, персонализация (как правило, большие базы данных разрабатываются коллективно, поэтому важно предоставлять удобный интерфейс для совместной работы);
- версия Community Edition является бесплатной (можно оценить базовый функционал);
- в начале работы при первом запуске предоставляется краткая справка о программе и обновлениях (пользователю подсказывают, что и как нужно делать);
- структура проекта отображается в отдельном окне (удобно оценивать диаграмму в целом);
- история действий отображается в отдельном окне (можно отслеживать действия);
- панель инструментов содержит всю часто используемую функциональность (не нужно открывать меню программы часто);
- справка по программе структурирована и сопровождена снимками экрана (пользователь даже без доступа в интернет может получить ответы на интересующие вопросы);
- возможность редактирования гарнитуры, размера и других параметров текста, используемого для отображения модели (внешний вид диаграммы можно настраивать в зависимости от целей пользователя);
- поддержка миникарты - схемы, которая изображает всю диаграмму в уменьшенном размере (удобно оценивать диаграмму в целом);
- множество настроек при генерации на основе шаблонов (позволяет пользователю выбрать нужные опции);
- поддержка большого количества СУБД (Access, IBM DB2, Informix, Ingres, MySQL, Oracle, Progress, MS SQL Server, Sybase, Teradata);
- автоматизированное создание структуры базы данных и обратное проектирование (после проектирования не нужно писать код, база данных создается из диаграммы; бывает необходимо получить наглядную диаграмму из существующей БД);
- поддержка документирования структур баз данных (при проектировании можно сразу документировать);
- перенос структур баз данных (но не самих данных) из одного типа СУБД в другой (нет необходимости рисовать диаграмму заново для другой СУБД);
- широкие возможности в процессе проектирования (индексы, триггеры, ограничения и пр.)

Недостатки:

- полная версия программы платная;
- поддерживаются ОС только семейства Windows (нет кроссплатформенности);
- интерфейс программы неудобен, а именно: добавление элементов на диаграмму невозможно непосредственно из окна логической модели; для добавления атрибутов нужно открывать отдельное окно, которое перекрывает часть диаграммы; никакие другие настройки схемы также нельзя выполнить в главном окне (это усложняет взаимодействие с пользователем);
- после генерации кода пользователь не оповещен об успешности/об ошибках генерации, сгенерированный файл не открывается.

2. **Dezign for Databases** - программный продукт моделирования баз данных, разработанный Heraut Solutions в 1998. Программа основана на ER-модели и поддерживает нотацию Crow`s Foot[4].

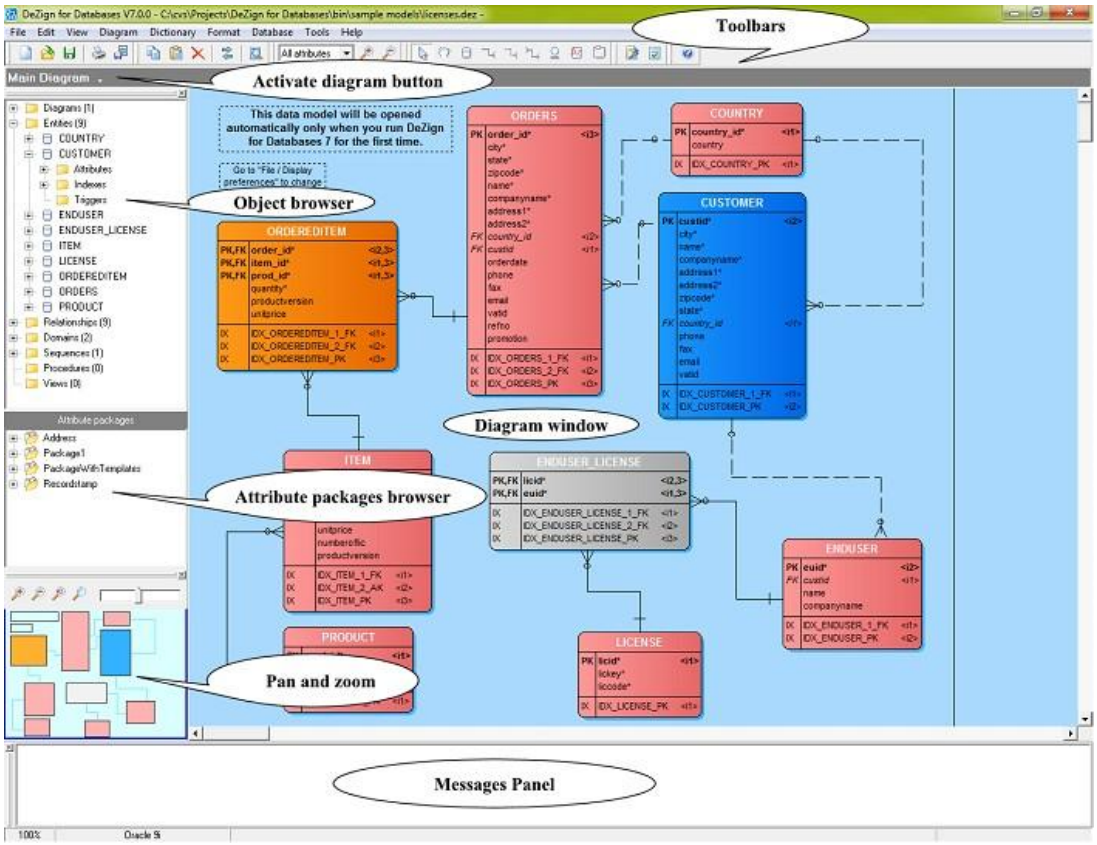


Рисунок 2: инструмент Dezign for Databases

[4] Продукт Dezign for Databases, url: http://www.datanamic.com/company/dezign-for-databases-v72-released.html (дата обращения 17.05.2015)

Достоинства:

- поддержка многих СУБД (MS SQL Server, MySQL, Oracle, IBM DB2, Firebird, InterBase, Informix, MS Access, SQLite, PostgreSQL, Sybase, Paradox);
- архитектура базы данных отображается в отдельном окне;
- присутствует миникарта;
- широкие возможности в процессе проектирования: поддержка ограничений, присутствие всех типов данных, можно создавать триггеры и индексы;
- поддержка физической и логической моделей;
- удобное графическое представление модели: атрибуты модели размещаются в прямоугольнике сущности, можно менять размер объектов, есть возможность настраивать толщину и цвет связей, имена сущностей и первичные ключи выделены;
- возможность обратного проектирования;
- при генерации скриптов в код добавляются комментарии (при работе со скриптами можно увидеть описание созданного кода);
- поддержка репозиториев.

Недостатки:

- при сохранении файл со сгенерированным кодом нигде не открывается (необходимо самостоятельно искать файл на диске);
- справка к приложению не содержит большого количества снимков экрана (пользователю сложнее разобраться с продуктом);
- типы связей один-к-одному и один-ко-многим неразличимы на диаграмме;
- присутствует только платная версия, бесплатная действует 30 дней;
- функция "сгенерировать код" труднонаходима;
- поддерживаются ОС только семейства Windows.

3. **PowerDesigner** - программа для коллективного моделирования, созданная компанией Sybase. PowerDesigner поддерживает модельно-ориентированную архитектуру (MDA). Программа может быть запущена как обособленное приложение и как плагин Eclipse[5].

[5] Продукт PowerDesigner, url: http://www.sybase.ru/products/powerdesigner (дата обращения 17.05.2015)

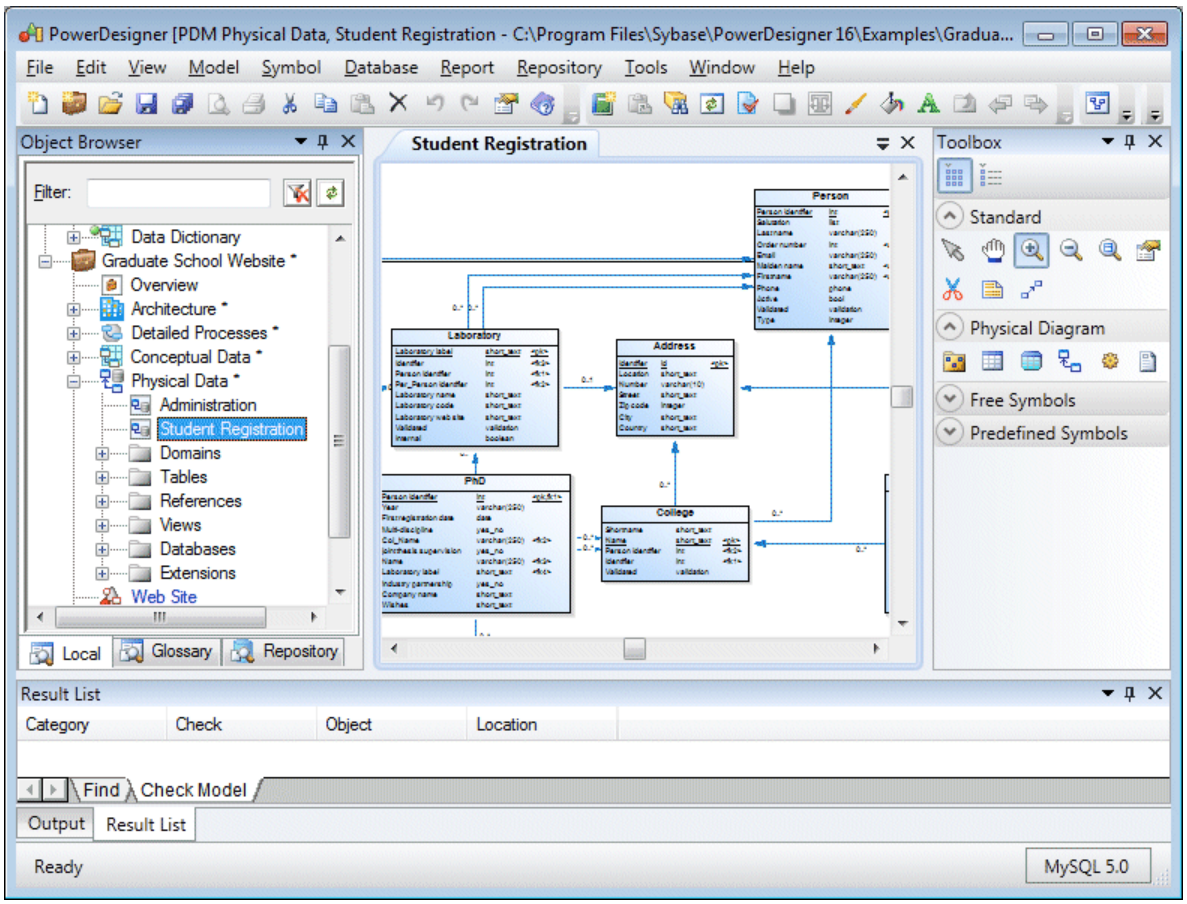


Рисунок 3: инструмент PowerDesigner

Достоинства:

- большое количество поддерживаемых СУБД (MS SQL Server, Oracle, PostgreSQL, MySQL, IBM DB2, Informix);
- возможность моделирования логической и физической модели данных;
- поддержка коллективной работы;
- настраиваемость внешнего вида;
- возможность создания новых и внесения изменений в имеющиеся шаблон для генерации кода (существует) пользовательский интерфейс;
- архитектура проекта отображается в отдельном окне;
- поддержка возможности обратного проектирования;
- сгенерированный файл с кодом открывается после процесса генерации.

Недостатки:

- процесс создания диаграмм неудобен: редактирование возможно лишь в отдельном окне;

- существует только платная версия;
- поддерживаются ОС только семейства Windows.

4. **Software Ideas Modeler** - продукт компании Dusan Rodina, позволящий моделировать UML-диаграммы[6].

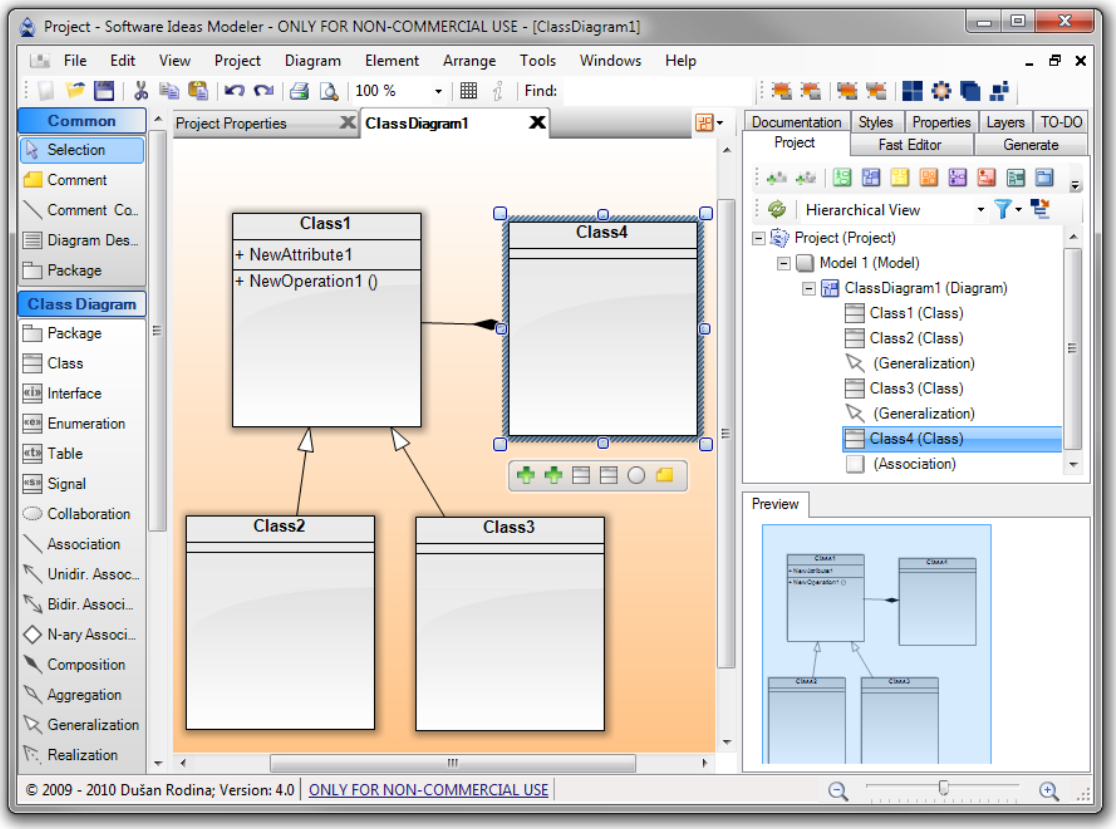


Рисунок 4: инструмент Software Ideas Modeler

Достоинства:

- возможность настраивать внешний вид диаграмм (не менее 30 опций);
- добавление свойства и комментариев возможно из главного окна с помощью контекстного меню;
- поддержка слоев (возможность менять наложение элементов друг на друга);
- поддерживаются операционные системы Windows, Linux;
- программа доступна бесплатно для некоммерческого использования.

[6] Продукт Software Ideas Modeler, url: https://www.softwareideas.net/ (дата обращения 17.05.2015)

Недостатки:

- поддержка множества различных диаграмм, не связанных с базами данных, затрудняет работу;
- небольшое количество поддерживаемых СУБД (MS SQL Server, MySQL);
- кнопка для запуска генерации кода труднонаходима;
- справка доступна только по интернету.

5. **Toad Data Modeler** - инструмент для создания и сопровождения баз данных компании Quest Software[7].

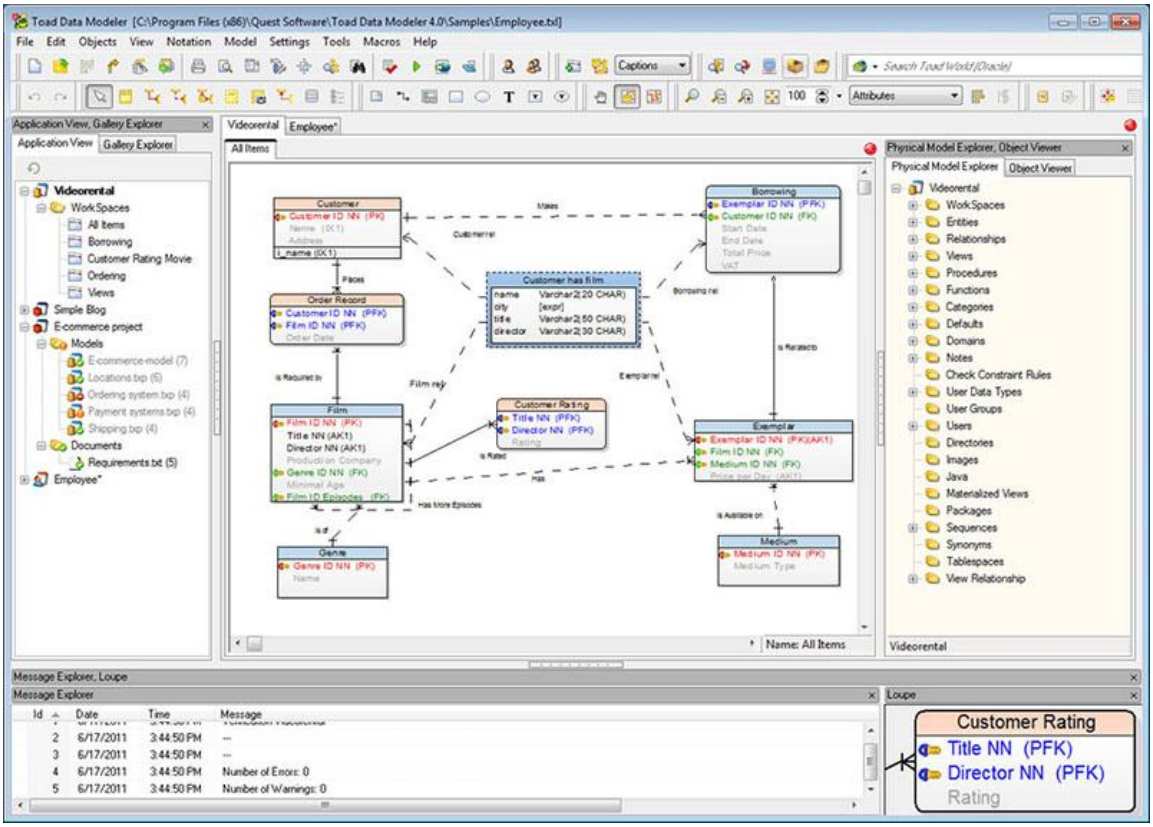


Рисунок 5: инструмент Toad Data Modeler

Достоинства:

- поддержка большого количество СУБД (Access, IBM DB2, Informix, MySQL, Oracle, PostgreSQL, MS SQL Server, SQLite);
- поддержка синхронизации моделей с физически существуемой базой данных;
- присутствует система контроля версий;

[7] Продукт Toad Data Modeler, url: http://software.dell.com/products/toad-data-modeler/ (дата обращения 17.05.2015)

- создание и поддержка задач с помощью "списка задач" (помогает пользователю структурировать план разработки);
- есть возможность настройки внешнего вида схемы;
- можно создавать как логические, так и физические схемы баз данных;
- возможность создания подробной HTML и RTF документации (документацию не нужно создавать отдельно);
- поддерживается галерея объектов, которую можно использовать совместно;
- возможность реверс-инжиниринга баз данных;
- возможность проверки моделей на ошибки (непустота имен, уникальность имен, допустимая длина имен, уникальность индексов, непустота объектов и пр.);
- физическая модель отображается в отдельном окне;
- предпросмотр SQL-кода (то, что будет сгенерировано, видно сразу);
- поддержка ограничений, триггеров,индексов, функций, процедур;
- присутствует карта с увеличением (возможность рассмотреть детали диаграммы);
- подробная справка со снимками экрана.

Недостатки:

- элементы схемы нельзя редактировать из главного окна, нужно открывать отдельное окно;
- поддерживаются ОС только семейства Windows;
- существует только платная версия.

2.3 Анализ в рамках поставленной цели

В процессе анализа существующих решений были составлены требования к проекту.

Функциональные требования:

- поддержка создания физической и логической модели БД;
- генерация SQL-кода для множества СУБД;
- проверка диаграмм на корректность.

Нефункциональные требования:

- бесплатность;
- кроссплатформенность;
- неперегруженный интерфейс;
- структурированная пользовательская документация со снимками работы приложения (для упрощения работы с инструментом).

Важно отметить, что работа с инструментом с открытым исходным кодом QReal, созданным с использованием Qt[8], делает проект бесплатным и кроссплатформенным.

[8] Инструмент Qt, url: http://www.qt.io/developers/ (дата обращения 17.05.2015)

2.4 Научные работы на кафедре Системного Программирования СПбГУ

На кафедре системного программирования математико-механического факультета СПбГУ существует довольно много работ на тему моделирования БД. Программные инструменты, которые разрабатывались на кафедре, использовали в качестве ядра CASE-пакет REAL. Для обозначения совокупности методик, позволяющих работать с пакетом, был выбран термин "Технология REAL-IT".

Самые близкие по тематике к данному проекту разработки были сделаны Ивановым А.Н. [3] и Нестеровым Н.С [5].

В процессе работы над диссертацией "Автоматизированная генерация информационных систем, ориентированных на данные" Иванов А.Н. достиг следующих результатов: создана модель пользовательского интерфейса ИС, ориентированного на работу с данными, методика ее построения и алгоритм генерации кода по модели; в REAL-IT расширена модель классов UML для проектирования БД; создан диаграммный язык для описания ограничений на модель классов; представленные решения были протестированы в ряде промышленных проектов.

Нестеров А.С. в дипломной работе "Перенос технологии REAL-IT на платформу Microsoft .Net" указывает следующие итоги: созданы генераторы графического пользовательского интерфейса для технологии REAL-IT/.Net; разработан механизм внесения изменений в автоматически созданный генераторами код форм; создано демонстрационное приложение.

Поддержка REAL-IT была прекращена, так как было накоплено много ошибок и технологии, используемые для разработки ядра системы, устарели. Изначально планировалось переписать части REAL на Qt, но разработчики пришли к выводу, что создание новой технологии будет более простым решением. Так появился проект QReal, который стал не просто CASE-средой, как это было с REAL, а платформой для быстрой разработки других языков.

Целью проекта "QReal:Databases" является не повторить существующие решения, которые были реализованы с помощью технологии REAL-IT, а поддержать возможность моделирования баз данных и на новой платформе, но на данный момент в учебных целях.

На момент создания проекта в QReal уже существовали наработки, связанные с БД. Студентами была поддержана возможность генерировать SQL-код из физической схемы БД и создавать диаграмму из SQL-кода. В отличие от реализованной функциональности, в QReal:Databases в конечном итоге планируется:

- предоставить пользователям возможность полностью проследить процесс проектирования: создание логической модели; создание физической модели; генерация SQL-кода;
- реализовать генерацию SQL-кода для множества СУБД, сделать процесс добавления новых СУБД простым;
- реализовать возможность генерировать код для редактирования таблиц базы данных;
- создать пользовательскую документацию.

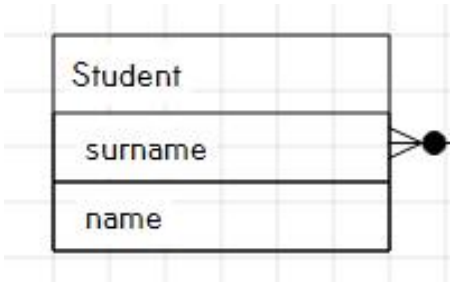
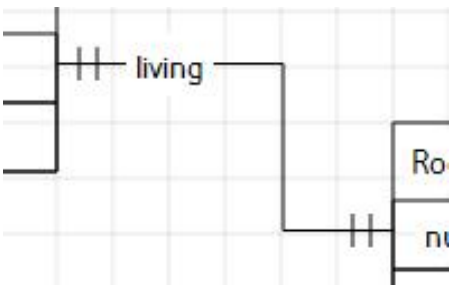
3. Описание выбранного подхода

3.1 Создание графических языков

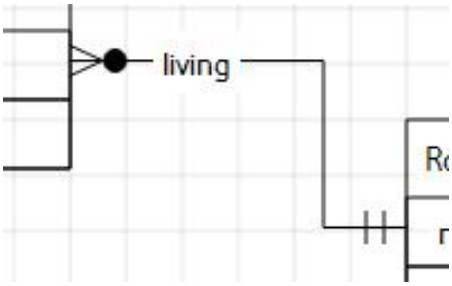
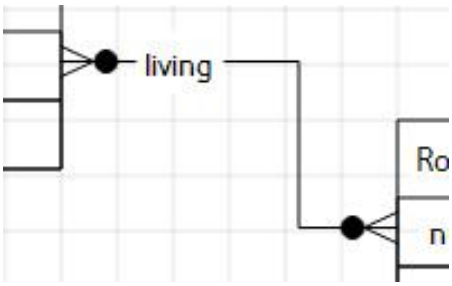
Первый этап проекта - это создание визуальных языков для логической и физической моделей базы данных. Логическая модель всегда одна, а физические специфичны для каждой конкретной СУБД. Изначально обсуждался вариант создания одного языка для логической модели и другого для физической, который бы изменялся в зависимости от выбранного диалекта. Но выбрано было более удобное решение, которое проще как в реализации, так и для пользователя: создан один язык для логической модели БД и по одному языку физической модели для каждой поддерживаемой СУБД.

В QReal метамодель может описываться текстовым файлом или графически. В данном проекте все метамодели заданы текстовым файлом. На данный момент создан один редактор для логической модели и редакторы для СУБД: Sql Server 2008[9] [1], My SQL 5[10] [4], SQLite[11] [8].

Язык для логической модели основан на ER-модели и содержит следующие элементы:

Изображение	Имя и свойства	Описание
 Сущность "Student" с атрибутами "surname" и "name"	Entity (сущность) • name	Класс объектов, информацию о которых необходимо хранить. Свойство "name" должно быть непустым.
	Attribute (атрибут сущности) • name	Свойство сущности, которое хотим учитывать при построении модели. Атрибут располагается внутри элемента "сущность", которому принадлежит. Свойство "name" должно быть непустым.
	One-to-one relationship (связь один-к-одному) • name	Связь между двумя сущностями, которая означает, что одному объекту первой сущности соответствует один объект второй сущности.

[9] Инструмент Sql Server 2008, url: https://www.microsoft.com/ru-ru/download/details.aspx?id=1695
[10] Инструмент MySQL, url: https://www.mysql.com/
[11] Инструмент SQLite, url: https://www.microsoft.com/ru-ru/download/details.aspx?id=169

	One-to-many relationship (связь один-ко-многим) • name • columnName	Связь между двумя сущностями, которая означает, что одному объекту первой сущности может соответствовать несколько объектов второй сущности. Связь изображается в виде стрелки с двумя разными концами: два штриха - для первой сущности, разветвление - для второй сущности. Свойство "columnName" используется при генерации физической модели БД (подробнее в разделе III).
	Many-to-many relationship (связь многие-ко-многим) • name • tableName	Связь между двумя сущностями, которая означает, что одному объекту первой сущности может соответствовать несколько объектов второй сущности и наоборот. Свойство "tableName" используется при генерации физической модели БД.

Используя полученную модель, можно создавать логические схемы различной степени сложности.

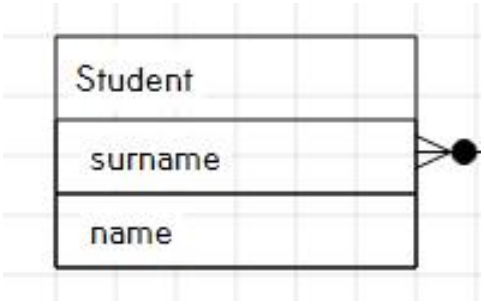
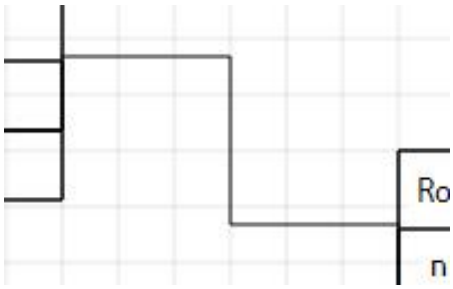
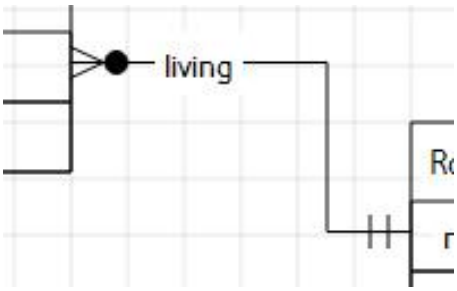
Стоит отметить, что в продуктах для коммерческой разработки БД связь "один-к-одному" не поддерживается, это значительно упрощает процесс создания физической модели и кода. Но данный проект - учебное приложение, и хотелось бы показать, что такая связь существует, а также продемонстрировать ее физическую реализацию.

Визуальные языки для отдельных СУБД были созданы не сразу, а после того, как была осознана их необходимость - для наглядного отображения физической модели проекта в QReal удобно создавать редакторы, которые специфичны для разных систем управления: тогда каждый редактор может создавать свою диаграмму и отображать на ней полученную модель.

На данный момент языки физических моделей состоят из одинаковых элементов, но свойства каждого элемента для различных СУБД отличаются. Это обосновано тем, что каждая СУБД оперирует одними объектами (таблица, строка), а различия, возникающие у каждого диалекта SQL, можно указать в свойствах этих объектов. Между тем, составив

языки из объектов с одинаковыми названиями, можно упростить процесс генерации кода: для того, чтобы сгенерировать код, не нужно значить ничего, кроме имени редактора.

Элементы языков для физической модели:

Изображение	Имя и обязательные свойства	Описание
 Таблица "Student" со столбцами "surname" и "name"	Table (таблица)	Элемент, который будет сгенерирован как таблица базы данных
	Column (столбец) • datatype	Атрибут таблицы, который будет сгенерирован как столбец таблицы БД, которой принадлежит. Располагается внутри элемента "Таблица".
	Many-to-many relationship (связь многие-ко-многим)	Связь, предназначенная для соединения участников отношения многие-ко-многим со сгенерированной для отношения таблицей. Присутствует в схеме для наглядности.
	One-to-many relationship (связь один-ко-многим)	Связь, предназначенная для соединения участников отношения один-ко-многим. Присутствует в схеме для наглядности.

Физическая модель базы данных генерируется из логической (подробнее в разделе III) в зависимости от выбранной СУБД.

3.2 Архитектура проекта

Для гибкости разработки и масштабируемости в QReal введено понятие плагина - это отдельный модуль, который можно добавить в проект, не изменяя код основной части. При запуске QReal инициализирует плагины, предоставляя им объекты для взаимодействия с репозиторием и основным окном программы. Для данного проекта создан плагин DatabasesSupportPlugin, который отвечает за связь между главным окном приложения и другими частями проекта. Класс содержит ссылки на экземпляры классов окна настроек и генератора и вызывает у экземпляров функции, требуемые пользователю.

Класс DatabasesPreferencesPage предназначен для работы с окном настроек плагина баз данных. Класс оповещает плагин об изменениях, внесенных пользователем в настройки.

Класс DatabasesGenerator инкапсулирует всю функциональность, связанную с генерацией: позволяет генерировать физическую модель БД из логической и SQL-код для различных СУБД.

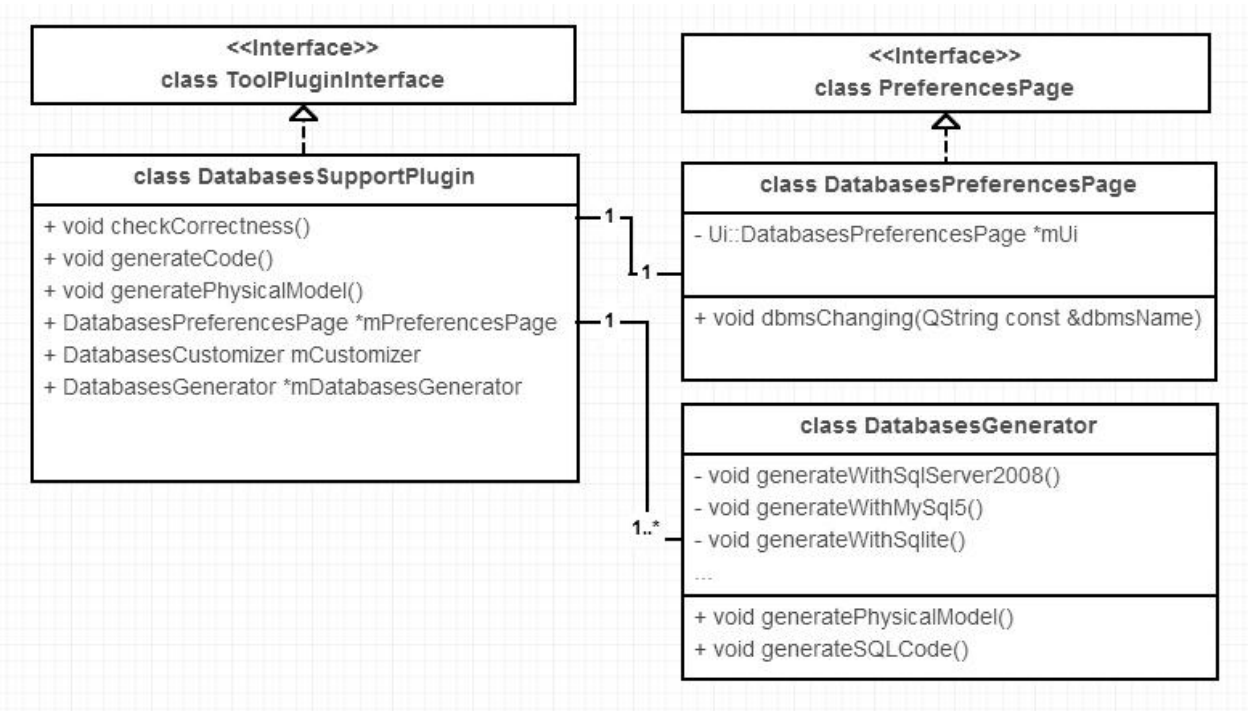


Рисунок 6: Иерархия классов созданного решения.

3.3 Реализация функциональности

Основная функциональность проекта - это генерация физической модели БД из логической модели и генерация SQL-кода. Она реализована в классе DatabasesGenerator.

Проверка диаграммы на корректность

Ошибки в построении связей выявляются только в процессе генерации физической модели, однако некоторые другие ошибки можно обнаружить и раньше. По запросу пользователя в диаграмме проверяются: для сущностей - непустота поля "name", для атрибутов - непустота поля "name", для связей - соединение с сущностями с двух сторон.

Такую проверку можно вызвать из меню приложения, автоматически она вызывается при генерации.

Генерация физической модели БД из логической модели

Каждый язык для физической модели состоит только из элементов с именами: "Table", "Column", "PhysicalManyToManyRelationship", "PhysicalOneToManyRelationship". А в процессе генерации используются только имена создаваемых элементов и имя редактора. В результате для того, чтобы генерировать физические модели для разных СУБД достаточно менять имя редактора (рис. 7) - элемент будет взят из указанного редактора и будет обладать специфичными для него свойствами.

```
al::Id DatabasesGenerator::createElementFromString(QString const &elemNa
  Id id = Id::loadFromString(QString("qrm:/" + mDbms
      + "/DatabasesPhysicalModelMetamodel/" + elemName));
  Id logicalId = mLogicalModelApi.createElement(Id::rootId(), id);
  Id graphicalParentId = Id::rootId();
```

Рисунок 7: Фрагмент кода из функции класса DatabasesGenerator: для создания элементов физической модели для различных СУБД требуется изменять только название генератора, которое хранится в переменной mDbms.

В процессе написания алгоритма генерации физической модели возникла такая проблема: два объекта, связанные отношением один-к-одному, по сути, являются одной таблицей [8]. И если создать, например, логическую схему с рис. 8, она окажется некорректной и сгенерировать код по ней окажется невозможным. (Существуют и другие случаи, когда связи указаны некорректно - рис. 9).

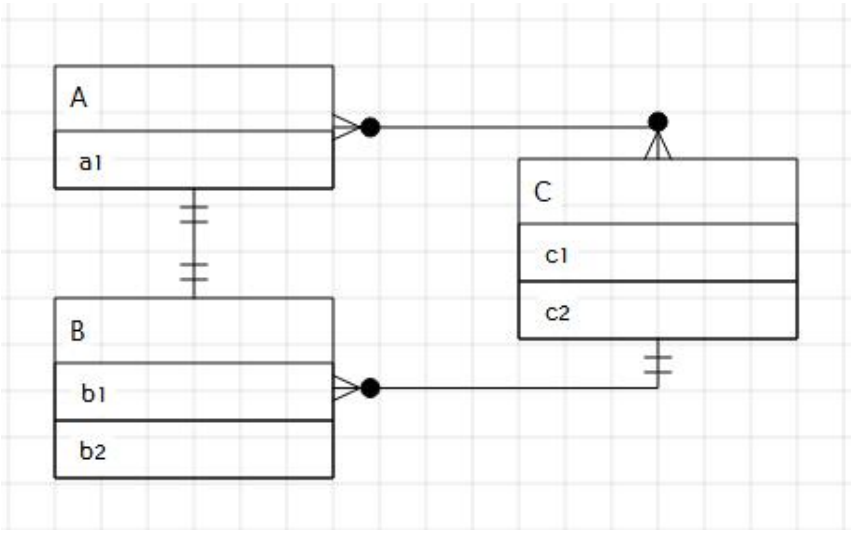


Рисунок 8: Пример ошибки проектирования, о которой нужно оповещать пользователя. Сущности A и B, связаны отношением один-к-одному и будут соединены в одну таблицу. Но сущность C связана с A и B разными отношениями - это невозможно.

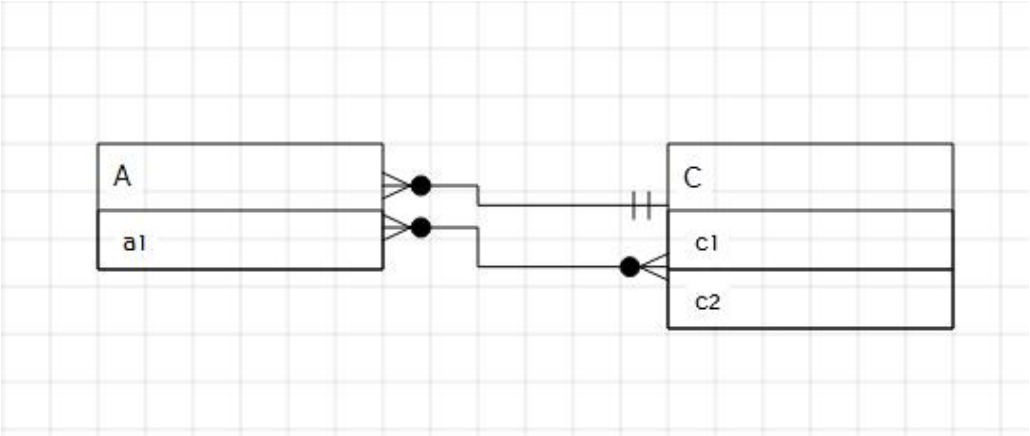


Рисунок 9: Пример ошибки проектирования. A и C не могут быть связаны двумя различными отношениями. Попытка сгенерировать физическую модель вызовет ошибку.

На небольших по размеру диаграммах некорректность не слишком сложно заметить, однако на большой схеме это может оказаться затруднительно. Таким образом, перед процессом генерации необходимо выяснить, какие сущности связаны друг с другом отношением один-к-одному. Это отношение транзитивно и симметрично, поэтому можно создавать множества, которые все связаны друг с другом этим отношением. Для краткости множество сущностей, связанных отношением один-к-одному, будем называть компонентой связности. Было принято решение находить компоненты связности и, учитывая информацию о компонентах, выводить пользователю информацию об ошибках.

Алгоритм нахождения компонент связности прост: сущность добавляется в компоненту; выясняется, есть ли среди отношений, в которых задействована сущность, связь один-к-одному; если нет - заканчиваем добавление в компоненту, если есть - повторяем алгоритм для второго участника отношения. В процессе алгоритма помечаем пройденные сущности, чтобы не уйти в бесконечный цикл.

В качестве примера рассмотрим диаграмму на рис.10. Сущность A связана только B, B связана только с C. Можем объединить их в первое множество ABC. D и E связаны друг с другом - это второе множество DE. F отношением один-к-одному не связано ни с кем - это третье множество, состоящее из одного элемента.

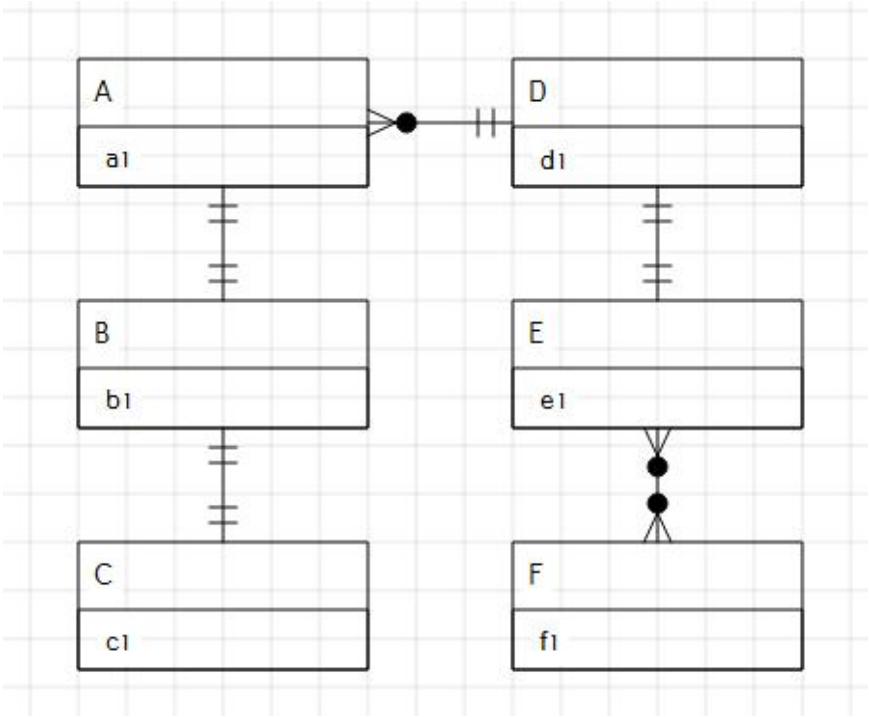


Рисунок 10: Пример диаграммы, нарисованной в QReal:Databases.

Можно реализовывать отношение один-к-одному и не слиянием двух сущностей в одну таблицу, но данным способ в проекте еще не создан.

Для исправления ошибок создается двумерная "матрица связей", которая, по сути, является матрицей смежности для графа, узлы которого - индексы компонент связности, а ребра - все связи, кроме "один-к-одному". Если в процессе генерации выяснится, что один из элементов компоненты i связан с элементом из компоненты j отношением один-ко-многим - значит, эти две компоненты связаны отношением один-ко-многим (то же самое со связью многие-ко-многим). Если элемент массива с индексами i,j еще пуст - значит,

ошибки нет, можно поставить вместо этого элемента идентификатор найденного отношения. Но если не пуст - значит, до этого встретилась другая связь, диаграмма некорректна и нужно завершить генерацию, выдав пользователю информацию ошибке.

Посмотрим на матрицу связей, которая получится в примере из рис. 10.

Между сущностями A и D есть связь один-ко-многим. A принадлежит множеству ABC, D принадлежит множеству DE. Значит, нужно поставить знак связи между множествами ABC и DE. То же самое со связью многие-ко-многим между E и F: необходимо поставить знак связи между их множествами DE и F соответственно.

Получится такая матрица (первые строчка и столбец - индексы компонент связности):

	ABC	DE	F
ABC	0	1	0
DE	-1	0	2
F	0	2	0

Обозначения:

0 - нет связей

1 - связь "один-ко-многим" (строка - конец "много", столбец - конец "один")

-1 - связь "один-ко-многим" (строка - конец "один", столбец - конец "много")

2 - связь "многие-ко-многим"

Далее следует алгоритм генерации, который можно разделить на этапы:

1. Первичная проверка корректности диаграммы (непустота некоторых свойств, наличие валидных концов у отношений).
2. Для сущностей, не связанных отношениями с другими сущностями, создается элемент "Table". Атрибуты, находящиеся внутри сущности, становятся атрибутами таблицы (элементами "Column").
   Для сущностей, связанных какими-либо отношениями, составляется компонента связности и добавляется в общий список компонент. Обработанные сущности помечаются.
3. Обрабатываются связи один-ко-многим. Каждая компонента связности становится таблицей (атрибуты всех сущностей просто объединяются в созданной таблице). К компоненте, соединенной с отношением концом "много", добавляется столбец - ключ второй компоненты [7]. Если ключ не указан в свойстве отношения, он будет сгенерирован автоматически.

4. Обрабатываются связи многие-ко-многим. Каждая компонента связности становится таблицей (атрибуты всех сущностей просто объединяются в созданной таблице) [8]. Создается отдельная таблица для отношения многие-ко-многим, содержащая столбцы для ключей связанных сущностей. Если имя таблицы не указано в свойстве отношения, оно будет сгенерирован автоматически.
5. Выводится информации об успешной генерации физической модели.

Генерация SQL-кода для различных СУБД

Для каждой СУБД создана функция, реализующая алгоритм генерации SQL-кода из физической модели. При нажатии пункта меню "сгенерировать код" выбирается функция, соответствующая СУБД, выбранной в окне настроек. Код генерируется в файл, имя которого указывается программно при создании объекта класса DatabasesGenerator.

Алгоритм генерации кода:

1. Находятся все объекты физической модели типа "Table"
2. Проверяется содержание свойств типа "Table" для текущего экземпляра, указанных в метамодели языка. Свойство описывается в файле с помощью синтаксиса текущего диалекта SQL.
3. Для экземпляра находятся атрибуты, которые вместе со свойствами атрибутов записываются в файл с помощью синтаксиса выбранного диалекта SQL.
4. Выводится информации об успешной генерации кода.

3.4 Апробация

Пример 1: небольшая база данных для подробной демонстрации работы приложения. База данных предназначена для хранения информации о детских мероприятиях. Каждый ребенок посещает мероприятие ровно с одним сопровождающим. Для посещения мероприятия нужен билет, который может быть общим для ребенка и его класса в школе. Каждый билет прикреплен к одному мероприятию.

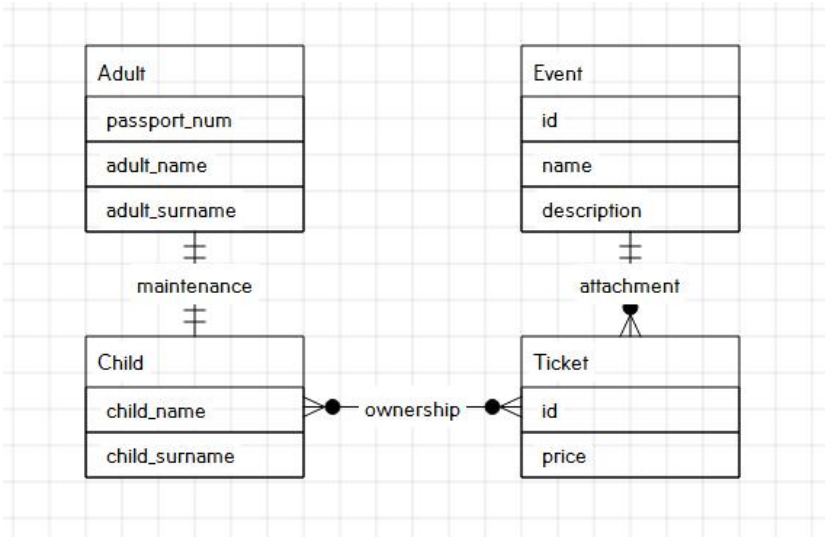


Рисунок 11: Логическая схема БД, составленная по описанию задачи.

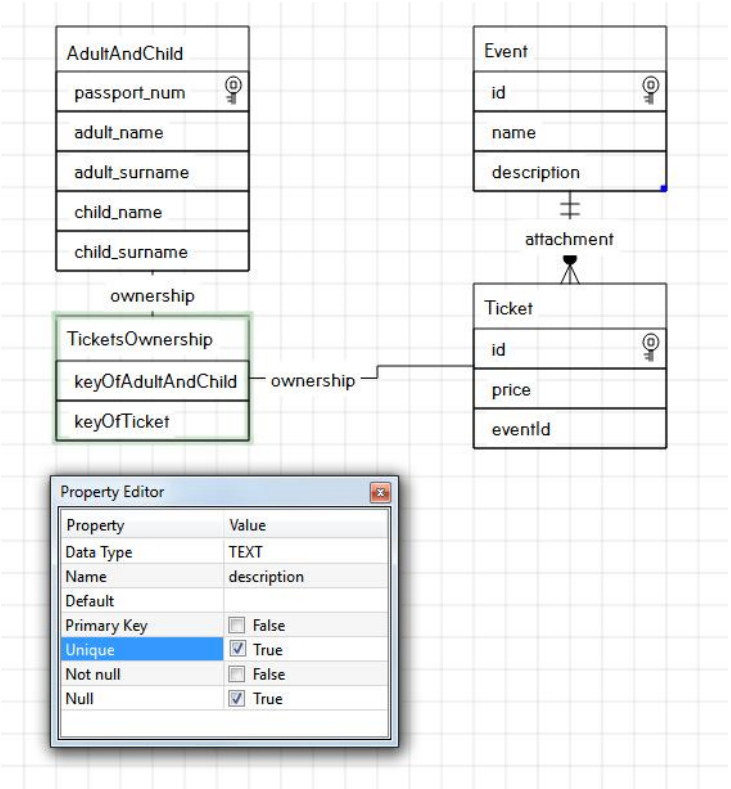


Рисунок 12: Редактирование физической схемы БД для Sql Server 2008, сгенерированной по логической модели.

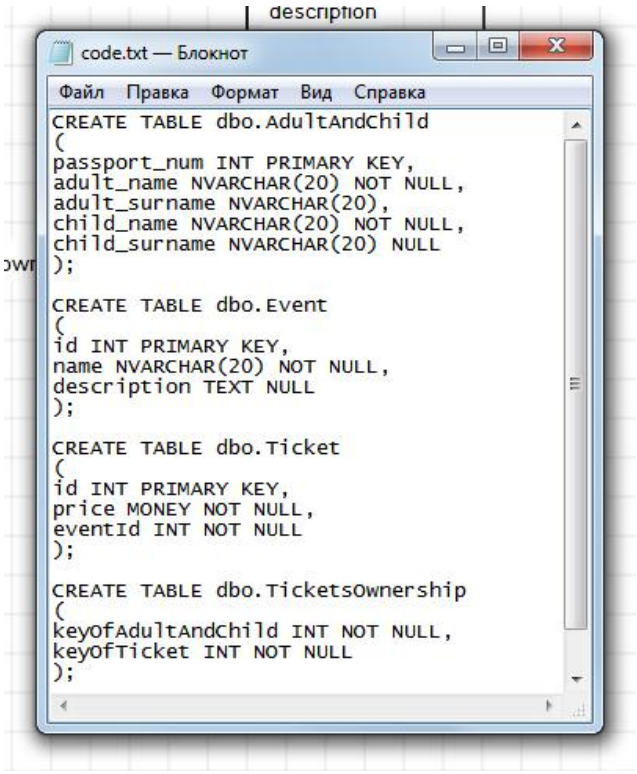


Рисунок 13: Текстовый файл со сгенерированным кодом для Sql Server 2008.

Пример 2: большая база данных. Пример предназначен для демонстрации работы в целом, позволяет сгенерировать базу данных для университета. Для простоты создания тип данных у всех столбцов указан по умолчанию.

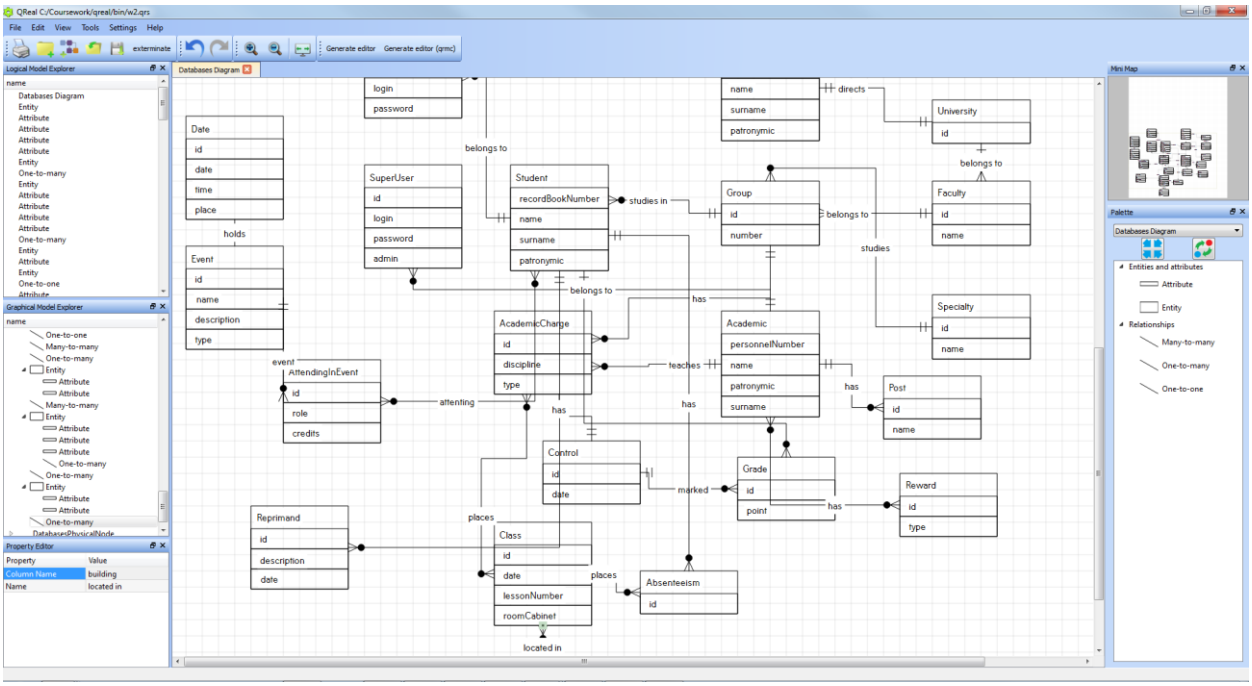


Рисунок 14: Нарисованная пользователем логическая схема БД.

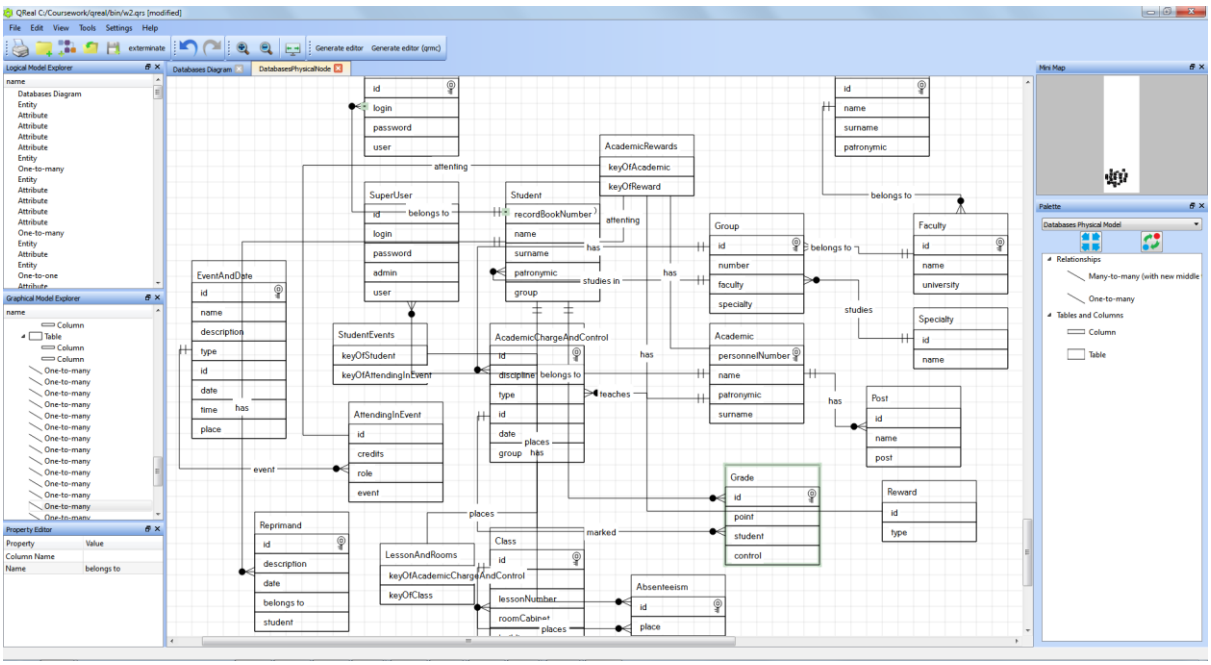
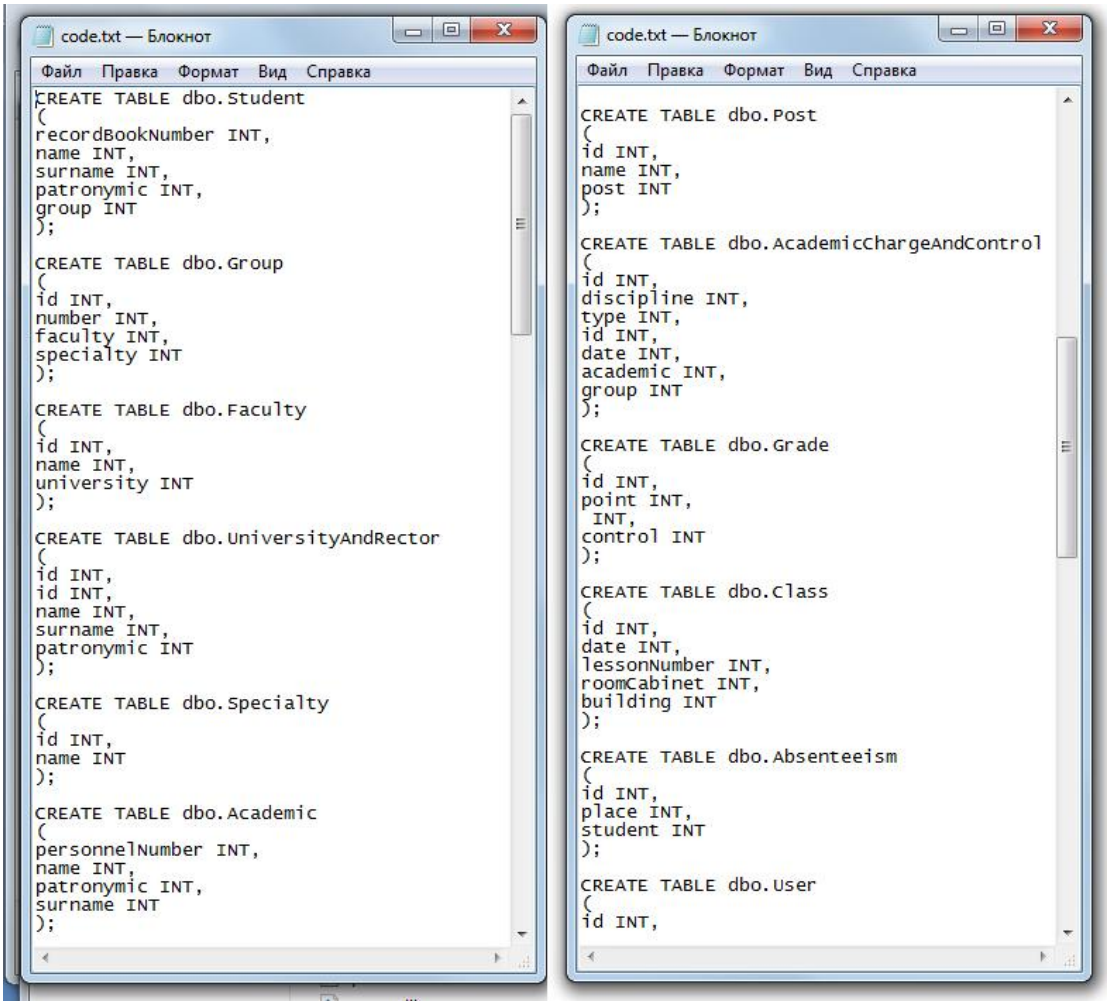


Рисунок 15: Сгенерированная физическая схема БД после ручной корректировки расположения элементов на диаграмме (для СУБД Sql Server 2008).



```
CREATE TABLE dbo.Student
(
recordBookNumber INT,
name INT,
surname INT,
patronymic INT,
group INT
);

CREATE TABLE dbo.Group
(
id INT,
number INT,
faculty INT,
specialty INT
);

CREATE TABLE dbo.Faculty
(
id INT,
name INT,
university INT
);

CREATE TABLE dbo.UniversityAndRector
(
id INT,
id INT,
name INT,
surname INT,
patronymic INT
);

CREATE TABLE dbo.Specialty
(
id INT,
name INT
);

CREATE TABLE dbo.Academic
(
personnelNumber INT,
name INT,
patronymic INT,
surname INT
);
```

```
CREATE TABLE dbo.Post
(
id INT,
name INT,
post INT
);

CREATE TABLE dbo.AcademicChargeAndControl
(
id INT,
discipline INT,
type INT,
id INT,
date INT,
academic INT,
group INT
);

CREATE TABLE dbo.Grade
(
id INT,
point INT,
 INT,
control INT
);

CREATE TABLE dbo.Class
(
id INT,
date INT,
lessonNumber INT,
roomCabinet INT,
building INT
);

CREATE TABLE dbo.Absenteeism
(
id INT,
place INT,
student INT
);

CREATE TABLE dbo.User
(
id INT,
```

Рисунок 16: Фрагменты файла с кодом для создания БД с помощью СУБД Sql Server 2008.

4. Заключение

4.1 Результат

В рамках курсовой работы были получены следующие результаты:

- проанализированы несколько систем для проектирования БД, выявлены их достоинства и недостатки, результаты анализа были рассмотрены в рамках текущей задачи;
- создан графический язык на основе ER-модели для логической схемы БД;
- созданы языки физических моделей БД для СУБД: Sql Server 2005, My SQL 5, SQLite;
- реализована генерация физической модели БД из логической модели;
- реализована генерация SQL-кода из физической модели для Sql Server 2005, My SQL 5, SQLite (создание таблиц);
- реализована проверка некоторых правил корректности диаграмм;
- произведена апробация функциональности на нескольких примерах.

4.2 Направления дальнейшей работы

Точные задачи следующего года работы над проектом находятся в процессе составления, на данный момент планируется:

- проанализировать еще некоторые системы для проектирования БД;
- реализовать возможность генерировать код для редактирования таблиц базы данных;
- обеспечить поддержку большего количества СУБД;
- продумать и реализовать поддержку таких возможностей, как: добавление индексов к таблицам, добавление кластеров, функций и процедур и пр.;
- создать пользовательскую документацию.

5. Список литературы

1. Бен-Ган И. Microsoft SQL Server 2008. Основы T-SQL: Пер. с англ. — СПб:БХВ-Петербург, 2009. — 432 с.
2. Димов Э.М., Диязитдинова А.Р., Качков Д.А. Проектирование информационных систем: Учебное пособие — Самара: ПГАТИ, 2003 — 78 с.
3. Иванов А.Н. Автоматизированная генерация информационных систем, ориентированных на данные. — Санкт-Петербург, 2005. — 96 с.
4. Кузнецов М., Симдянов И. MySQL 5. — СПб:БХВ-Петербург, 2010. — 1024 с.
5. Нестеров А.В. Перенос технологии REAL-IT на платформу Microsoft .Net. — Санкт-Петербург, 2007. — 21 с.
6. 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.
7. 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.
8. Rick F. van der Lans. The SQL Guide to SQLite. — Lulu, 2009. — 524 с.
9. Руководство по проектированию реляционных баз данных (часть I), url: http://habrahabr.ru/post/193136/, дата обращения: 07.05.2015
10. Руководство по проектированию реляционных баз данных (часть II), url: http://habrahabr.ru/post/193284/, дата обращения: 07.05.2015
11. Руководство по проектированию реляционных баз данных (часть III), url:: http://habrahabr.ru/post/193380/, дата обращения: 07.05.2015