Министерство образования и науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

С. Г. ПОПОВ

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

Учебно-методические указания к выполнению курсовой работы

Санкт-Петербург — 2015

УДК 004.047 (076.5)

ПОПОВ С.Г. Реляционные базы данных: учебно-методические указания. 2017. 50 с.

Учебно-методические рекомендации содержат справочные материалы, пояснения и примеры выполнения разделов курсовой работы дисциплин подготовки бакалавров “Реляционные базы данных”, “Базы данных”. Для каждой части курсовой работы описаны цели, сформулированы результат и требования к результату, предложена технология выполнения и приведены примеры реализации для СУБД MySQL и языков программирования C# и C++. Указания могут быть использованы студентами в качестве дополнительных материалов при самостоятельном выполнении разделов курсовой работы.

Рекомендации предназначены для студентов направлений подготовки бакалавров “Математика и компьютерные науки” и “Информатика и вычислительная техника”.

Иллюстраций **??**.

Содержание

1 Цель и задачи курсовой работы

Целью выполнения курсовой работы является овладение студентами практическими навыками описания процессов и данных предметной области, проектирования эффективной схемы реляционной базы данных и реализации типичных запросов к ней.

Достижение цели обеспечивается выполнением индивидуального задания для выполнения полного цикла управления данными, состоящего из связанных между собой этапов:

- исследования и формализации процессов и данных предметной области;
- проектирования базы данных
- программирования базы данных и формирования тестовых наборов данных;
- программирования запросов к базе данных.

На этапе исследования и формализации предметной области необходимо сформировать и зафиксировать общее понимание предметной области на естественном языке. Для этого требуется разработать текстовое описание выбранной предметной области фиксирующее значимые сущности, связи между ними и их атрибуты. Изобразить отношения между сущностями в форме иерархии объектов, графического представления основного процесса и ER–диаграмм.

Целью этапа является получение навыков описания предметной области в неформальном и формализованном видах. Умение выделять и фиксировать сущности и их атрибуты, определять типы и домены и ограничения. Выделять значимые отношения между сущностями.

Результатом выполнения первого этапа является текстовое описание предметной области и сущностей, графическое представление иерархии объектов и ER-диаграмма.

На этапе проектирования базы данных необходимо повысить уровень формализации представления сущностей и их атрибутов, превратив их в реляционные отношения, конкретизировать атрибуты, определить домены, ограничения доменов выбрать типы данных, спроектировать базу данных на основе принципов объектно-реляционной модели, для чего: выделить словари, определить соотношения “многие ко многим”, выделить промежуточные таблицы, и представить описание схемы в машинно-независимом графическом и текстовом виде.

Целью этапа является получение навыков формализации данных, и зависимостей между ними, понимание сущности метаданных, формирование понимания различий между доменом и типом, фиксация понятия словаря, устойчивое понимание сути связи “многие ко многим”. Результатом выполнения второго этапа является спроектированная схема базы данных в графической и текстовой формах, таблица с описанием отношений, атрибутов и реализованные зависимостей.

На этапе программирования базы данных и формализации тестовых наборов данных требуется реализовать спроектированную схему в конкретной реализации СУБД, определить содержимое словарей, выбрать объём и распределение данных по остальным отношениям, и заполнить базу данных тестовыми данными в согласованном объёме и с заданными распределениями вероятностей.

Целью этапа является приобретение навыков обращения с СУБД, освоения языка DDL, написания автоматизированных средств заполнения базы данных данными на языках программирования высокого уровня.

Результатом этапа является отлаженная и созданная в СУБД схема базы данных на языке DDL, включающая в себя описание внешних ключей и ограничений домена, набор данных словарей на языке DML, и программа на языке высокого уровня, заполняющая прочие таблицы данными, по ранее заданному распределению.

На этапе программирования запросов к базе данных требуется реализовать индивидуальные запросы на языке DML к схеме базы данных, объяснить технологию их выполнения, показать их эффективность и представить результаты в наглядном виде.

Целью этапа является получение навыков написания запросов, формирования понимания их выполнения в СУБД, и приобретение умения графического представления результатов.

Результатом выполнения этапа являются отчёт о выполнении 8-10 индивидуальных запросов.

По результатам работы оформляется отчёт о курсовой работе. Формат, способы отчётности и содержание разделов отчёта и презентации описаны в следующих разделах методических указаний. Составленный и защищённый отчёт может являться исходными данными для реализации курсовых работ по дисциплинам “Проектирование WEB–приложений” и “Программирование WEB–приложений” в следующих семестрах обучения.

2 Формы отчётности и технология защиты работы

Основным документом по реализации курсовой работы является отчёт. Структура и содержание отчёта изложено в разделе «Структура отчёта» на странице 9. Дополнительным отчётным документом является презентация. Презентация предназначена для демонстрации промежуточных результатов с целью их публичного обсуждения. Структура презентации изложена в разделе «Структура презентации» на странице 11. Публичное обсуждение проводится в форме краткого устного доклада и ответов на вопросы преподавателя и студентов. В ходе дискуссии формируется объективное восприятие предмета, формируются границы задачи, окончательно согласуется набор атрибутов и отношений, требуемый для решения задачи.

Защита работы осуществляется в четыре этапа:

Первый этап: обсуждение и согласование содержания предметной области, основного процесса, иерархии объектов, ER–диаграммы. Результат: согласованные описания, выполненная первая четверть отчёта. Форма отчётности: презентация, выступление, раздел отчёта. Срок: 4-5 занятие семестра.

Второй этап: обсуждение и согласование перечня атрибутов и отношений, схемы базы данных, доменов и типов. Результат: согласованная схема базы данных, отлаженная программа заполнения данными.Форма отчётности: презентация, выступление, раздел отчёта. Срок: 8-9 занятие семестра.

Третий этап: согласование объёмов заполнения базы данных, определение законов распределения данных между отношениями. Результат: определённые объёмы и распределения данных, согласованная технология заполнения базы данных. Форма отчётности: раздел отчёта. Срок: 11-12 занятие семестра.

Четвёртый этап:формулировка индивидуальных запросов к схеме базы данных. Результат: выполненные запросы. Форма отчётности: отчёт. Срок: зачётная неделя семестра.

Каждый этап сопровождается оформлением соответствующей части отчёта. Отчёт предоставляется в печатном или рукописном виде, на листах бумаги с титульным листом. Вопросы, возникающие в процессе работы, заносятся на оборот титульного листа. Каждое следующее обсужде-

ние начинается с просмотра списка оставшихся вопросов. Демонстрация результатов части второго, третьего и четвёртого этапов предполагает демонстрацию результатов на вычислительной технике.

3 Структура отчёта

Отчёт является основным отчётным документом о результатах выполнения курсовой работы, должен содержать описания всех этапов работ с фиксацией содержательного наполнения. Содержание отчёта должно обеспечить воспроизводимость любого этапа работ и быть понятным стороннему читателю.

Отчёт формируется по правилам, определённым ГОСТ 7.32-2001 «Отчёт о научно-исследовательской работе. Структура и правила оформления». Предпочтительный формат бумаги — А4, ER–диаграммы, схемы базы данных допускается выполнять на форматах А3. Предпочтительное размещение листа — вертикальное, для таблиц допускается горизонтальное. Все таблицы и рисунки должны быть подписаны и пронумерованы. Титульный лист считается, но не нумеруется.

Отчёт содержит:

1. Титульный лист.
2. Задание на курсовую работу.
3. Часть 1. Описание предметной области.
   - текстовое описание предметной области;
   - определение основных понятий;
   - описание ролей;
   - графическое представление основного процесса;
   - иерархия объектов предметной области;
   - ER–диаграмма. Описание ER–диаграммы.
4. Часть 2. Проектирование схемы базы данных.
   - описание атрибутов и отношений данных;
   - графическое представление базы данных на русском языке;
   - графическое представление базы данных на английском языке с указанием числа записей в таблицах;
   - текстовое описание схемы базы данных.
5. Часть 3. Программирование схемы, подготовка исходных данных.
   - исходный код программы на языке DDL для создания базы данных в СУБД MySQL. Текст приводится в приложении А;
   - исходный код программы на языке программирования высокого уровня для заполнения базы данных тестовыми данными. Текст приводится в приложении Б;

- исходный код программы на языке DML с командами заполнения словарей. Текст приводится в приложении В.

6. Часть 4. Программирование запросов к БД.
   - описание запроса;
   - анализ эффективности выполнения запроса;
   - анализ результатов запроса.
7. Выводы.

Каждый раздел отчёта необходимо начинать с начала новой страницы. При необходимости графические материалы размещать на отдельных "плавающих" страницах. Размер и внешний вид графических данных должен обеспечить их читаемость, для этого необходимо выбирать гаммы цветов и текстуры заполнения, позволяющие надёжно различить части изображения.

Выполнение отчёта возможно в технологии Microsoft Word for Windows или LaTeX.

Детальное описание содержания пунктов отчёта приведено в разделе «Содержание отчёта» на странице 12.

4 Структура презентации

Презентация является вспомогательным документом и служит для наглядной демонстрации текущих результатов с целью и обсуждения во время доклада.

Структура презентации соответствует первым двум частям отчёта и содержит графические материалы, обеспечивающие пояснения к выступлению автора.

Презентация содержит:

- титульный лист.
- определение основных понятий;
- описание ролей;
- графическое представление основного процесса;
- иерархия объектов предметной области;
- ER–диаграмму;
- описание атрибутов и отношений данных;
- графическое представление базы данных на русском языке;
- графическое представление базы данных на английском языке;

Рекомендуемое время доклада — 10 минут. Время дискуссии от 15 минут. Рекомендуется конспектировать результаты обсуждения. Периодически желательно фотографировать результаты изменений схем.

Презентация может быть выполнена в технологии Microsoft Word for Windows или при помощи пакета beamer LaTeX.

5 Содержание отчёта

Раздел содержит описание частей отчёта по курсовой работе. В каждом разделе приведены требования к оформлению материалов, технологические приёмы по реализации задачи и два примера результатов работы над разделом.

5.1 Текстовое описание предметной области

Описание предметной области содержит не формализованное описание содержания задачи в свободном изложении на русском или английском языках. Описание должно быть грамматически правильным, содержать описание основного процесса, ролей и связанных с ними данных. Описание должно быть на столько полным, чтобы обеспечивать построение и однозначное толкование иерархии классов, ER–диаграммы, определений основных понятий, перечня ролей и описания основного процесса. Объём описания — 2–2,5 листа.

Примеры описаний предметных областей.

Предметная область «Выставка собак» Лисенкова

Выставка собак — организованное мероприятие, направленное на сбор данных о представителях породистого собачьего племени для получения представления о существующем в данном месте и на данное время поголовье или его части, сравнение уровня разведения различных клубов, а также выявлении лучших представителей пород. Они подразделяются на всепородные, на которых проводится экспертиза собак всех пород, и монопородные — специализируются на конкретной породе. Любая выставка обязательно имеет ранг.

Всепородные выставки бывают:

– Интернациональные выставки ранга CACIB (Certificats d'Aptitude au Championnat International de Beaute — сертификат Кандидата в Интернациональные Чемпионы красоты).
– Национальные ранга CAC (Certificats d'Aptitude au Championnat National de Beaute — сертификат Кандидата в Национальные Чемпионы красоты). В России Национальные выставки бывают трёх типов – ранга Чемпион РКФ(ЧРФК), ранга Чемпион Федерации (ЧФ) и ранга Кандидат в Чемпионы Федерации(КЧФ).

Монопородные выставки в России бывают следующих рангов:

– Чемпион Национального Клуба Породы(ЧК).
– Победитель Клуба(ПК).
– Кандидат в Чемпионы Клуба(КЧК).

Необходимо знать, что участвовать в выставке может собака в любом возрасте. И, относительно своего возраста, она относится к определённому выставочному классу:

1. Baby – с 3 до 6 месяцев.
2. Puppy – с 6 до 9 месяцев.
3. Junior – с 9 до 18 месяцев.
4. Intermediate – с 15 до 24 месяцев.
5. Open – с 15 месяцев.
6. Winner – с 15 месяцев (класс присутствует только на монопородных выставках, для записи в этот класс необходимо наличие хотя бы одного титула CAC или КЧК).
7. Working – с 15 месяцев (для записи в этот класс нужно иметь диплом: российский для записи на выставки национальные и монопородные, международный – для записи на выставки интернациональные и любые другие за рубежом).
8. Champion – с 15 месяцев (для записи в класс необходимо иметь титул чемпиона страны FCI).
9. Veteran – с 8 лет.

На выставке происходит группировка пород по рингам, где по усмотрению судей присуждаются титулы и выдаются соответствующие сертификаты. Расписание (время конкретного ринга) обычно можно узнать у организаторов за два-три дня до начала мероприятия.

При записи собаки на выставку владелец должен предоставить:

– заполненный и подписанный собственноручно заявочный лист;
– копию родословной или копию щенячьей карты (только для класса Щенков и Юниоров до 15 месяцев);
– копию квитанции об оплате целевого взноса за участие в выставке;
– копии рабочих сертификатов и чемпионских сертификатов при записи в рабочий класс и класс чемпионов. Без вышеперечисленных сертификатов запись на выставку должна производиться только в открытый класс.

Хозяину, приехав на выставку, как минимум, за час до назначенного времени, надо успеть пройти ветконтроль (проверка ветпаспорта, справки). После осмотра выдаётся квитанция, которая необходима для полу-

чения номера участника. Далее, в назначенном ринге, судья производит осмотр каждой собаки, делает её описание и присуждает оценку. Требуется отметить, что движение собаки по рингу осуществляется рядом с хендлером – специально обученным человеком (чаще всего в его роли выступает сам хозяин). После прохождения всех этапов собаке присуждается титул и выдаётся соответствующий сертификат.

Предметная область «Запись на прием в больнице» Кацал

Больница является самостоятельным структурным подразделением, где проводятся обследования и осуществляется лечение пациентов, в ее структуру входят стационар, где круглосуточно находятся пациенты, и поликлиника. Поликлиника является частью больницы и необходима для приема, обследования и проведения лечения пациентов, которым не нужна госпитализация. В данной работе будут рассматриваться только те пациенты, которые обращаются в поликлинику.

Поликлиника представляет собой специализированное лечебно - профилактическое медицинское учреждение, где пациент может получить амбулаторную медицинскую помощь. Данное учреждение имеет в своем штате большое количество специалистов из самых различных областей медицины. При обращении в поликлинику, пациент получает лечение на территории учреждения или же лечится у себя дома. В поликлиниках производится профилактический осмотр, обследование, которое может проводить терапевт или же узкий специалист. Существует множество поликлиник (будем рассматривать поликлиники внутри одной страны). Каждая поликлиника характеризуется адресом и наименованием.

Каждая поликлиника логически разделена на части, называемые участками. К каждому участку приписано множество пациентов. Пациент характеризуется Ф.И.О., номером паспорта, номером полиса. Участок, к которому приписан пациент, определяется адресом, по которому он проживает. Пациент может лечиться в нескольких поликлиниках (т.к. в полисе пациента может указываться не одна поликлиника. Например, хирургическое отделение располагается в одной поликлинике, а офтальмологическое в другой).

В каждой поликлинике главврач или доверенное лицо составляет штатное расписание. Форма для штатного расписания может быть самостоятельно разработана организацией, исходя из своих потребностей, но все-таки в качестве бланка рекомендуется использовать унифициро-

ванную форму № Т-3. В штатном расписании указывается количество специалистов, их рабочий день и заработная плата.

Согласно штатному расписанию, поликлиника составляет расписание, в котором указывается Ф.И.О. врача, его специализация, кабинет, день недели и часы приема. Расписание может изменяться, если конкретный врач заболел или в ушел в отпуск. Врачи могут работать в нескольких поликлиниках и могут обладать несколькими специализациями. Врач работает согласно составленному больницей расписанию. При этом следуют помнить, что врач также может стать пациентом.

Для того, чтобы пациенту записаться на прием к врачу, ему необходимо получить талон, в котором указывается дата, время, Ф.И.О. врача, его специализация и кабинет. Каждый пациент может получить столько талонов, сколько ему необходимо. Рассмотрим полный цикл приема пациента врачом. Человек, который нуждается в медицинской помощи, оформляет талон. Сделать это можно 2 способами: через интернет и в регистратуре поликлиники. Поликлиника, в которую обращается пациент, определяется адресом его проживания (участком, к которому приписан пациент). В указанные в талоне дату и время пациент приходит в нужный кабинет к специалисту и получает медицинскую помощь. После приема врач забирает талон, тем самым подтверждается факт приема.

5.2 Определение основных понятий

В разделе приводятся определения основных понятий предметной области. Каждое определение должно ссылаться на источник данных. Все специальные термины раздела текстового описания должны быть определены в этом разделе. Определения используются для пояснения основных понятий предметной области и формирования иерархии классов. Типичное число определений 12–15. Типичный объём 3–4 страницы.

Примеры определений для предметных областей:

Предметная область “Расписание экзаменов” Дементьев

Экзамен — итоговая форма оценки знаний определенного предмета;

Дисциплина — определенный круг базовых знаний определенной науки, преподаваемый студентам;

Преподаватель — лицо, обучающее студентов и принимающее экзамен;

Группа — совокупность студентов, учащихся в высшем учебном заведении, сгруппированных по одному, общему для всех, направлению обучения;

Поток — совокупность групп, объединенных для сдачи экзамена;

Аудитория — помещение, предназначенное для проведения лекции, семинаров, экзаменов и т.д.;

Время экзамена — совокупность возможных временных промежутков для проведения экзамена;

Комиссия — совокупность преподавателей, принимающих экзамен по дисциплине;

Расписание — вид календаря с указанием информации о предстоящих экзаменах.

Предметная область "Администрирование зоопарками"

Зоологический парк (зоопарк) — учреждение для содержания животных с целью их демонстрации, сохранения, воспроизводства и изучения, в том числе и научного.

Отделы зоопарка — единицы управления (администрирования) зоопарком.

Вид животного — таксон, основная структурная единица биологической систематики животных. Любое животное имеет вид.

Город — крупный населённый пункт, жители которого заняты, как правило, не сельским хозяйством.

Страна — территория, на которой располагаются города и другие населенные пункты, имеющая определённые политические, климатические, культурные или исторические границы.

Класс — один из основных рангов иерархической классификации в биологической систематике.

Хранитель — человек, имеющий необходимое образование, занимающийся содержанием и уходом за определенным животным или животными.

5.3 Описание ролей

В разделе формируется определение ролей участников основного процесса. Каждая роль характеризуется устойчивым набором функций, периодически выполняемой в автоматизируемом процессе. Перечень ролей не обязательно должен совпадать с перечнем пользователей информаци-

онной системы. В учебной задаче может быть выделено, в среднем 3–5 ролей. Ниже приведён пример ролей для двух предметных областей.

Роли участников задачи "Расписание экзаменов":

Заведующий кафедрой — составитель учебного плана. При наличии учебного плана происходит составление расписания экзаменов.

Преподаватель — человек, который может принимать экзамен в единственном лице или находиться в составе экзаменационной комиссии. Также деканатом учитываются пожелания преподавателя по поводу порядка проведения экзаменов.

Староста — студент, отвечающий за группу. При составлении расписания экзаменационной сессии, в его обязанности входит: опросить свою группу на предмет желаемого расписания экзаменов и донести эту информацию до деканата, в письменной или устной форме.

Студент — учащийся университета, имеющий право делиться своим мнением со старостой по поводу порядка проведения экзаменов.

Составитель расписания — человек, собирающий информацию об экзаменационной сессии и структурирующий расписание экзаменов на основе собранной информации.

Оператор — человек, занимающийся вводом готового расписания экзаменов в информационную систему.

Роли участников задачи "Администрирование зоопарком"

Посетитель — человек, собирающийся посетить зоопарк или обладать необходимой информацией для посещения зоопарка. Посетителю необходимо обладать информацией о том какие животные содержатся в зоопарках и какие из них экспонируются.

Сотрудник — человек, обладающий специальными навыками и соответствующим образованием, работающий в зоопарке.

Хранитель — сотрудник зоопарка, который содержит животных. Сотруднику необходим доступ к большему числу данных о животных, которых он содержит.

Начальник отдела — человек, который руководит отделом. Начальник отдела располагает большим числом данных о сотрудниках своего отдела.

Начальник зоопарка — человек, который руководит зоопарком.

5.4 Описание основного автоматизируемого процесса

Раздел содержит текстовое описание и графическое представление автоматизируемого процесса. Схема приводится в виде “как должно быть”. Текстовое описание должно быть разделено на шаги. Каждый шаг должен содержать перечисление причастных ролей и кратное описание действий над данными.

5.5 Иерархия объектов предметной области

Кацал Медведев

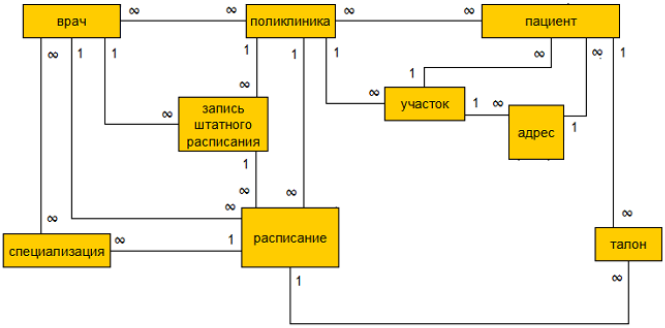

Рис. 1. Иерархия объектов предметной области “Запись на прием в больнице”

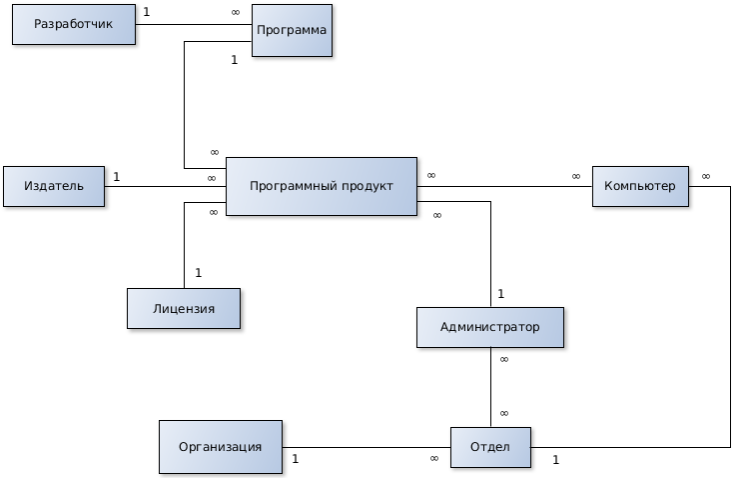

Рис. 2. Иерархия объектов предметной области “Система учета установленного ПО в организациях”

5.6 ER-диаграмма

Раздел содержит примеры описания ER–диаграмм. ER–диаграммы описывают отношения сущностей предметной области и частично формализуют описание до графических схем. Диаграммы могут отображать отношения сущностей в процессе реализации функций ролей предметной области, а могут отображать отношения объектов информационной системы. Отображение ER-модель приводится в нотации Питера Чена.

В дальнейшем, на основе диаграммы принимается решение о построении схемы базы данных.

Примеры ER–диаграмм приведены на рисунках. Кацал Медведев

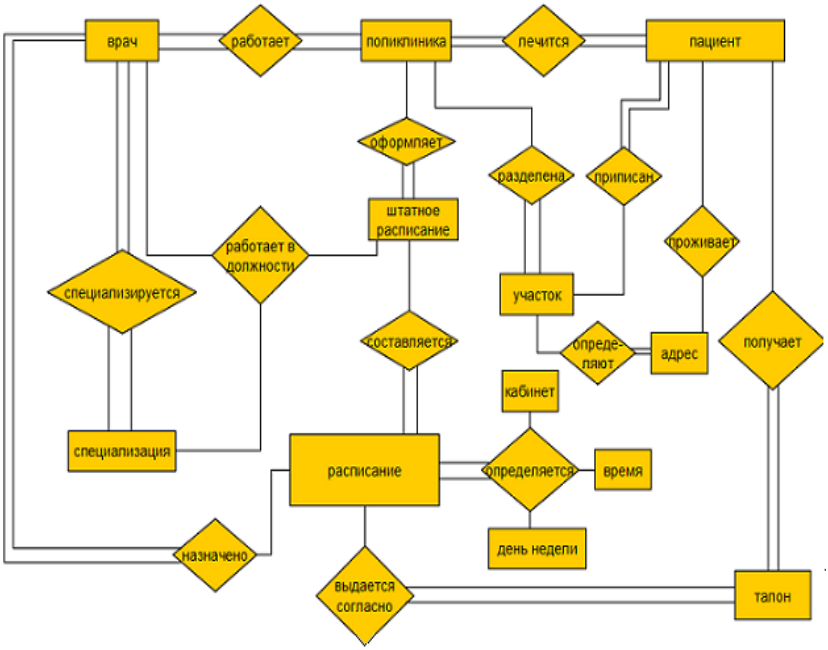

Рис. 3. ER–диаграмма предметной области “Запись на прием в больнице”

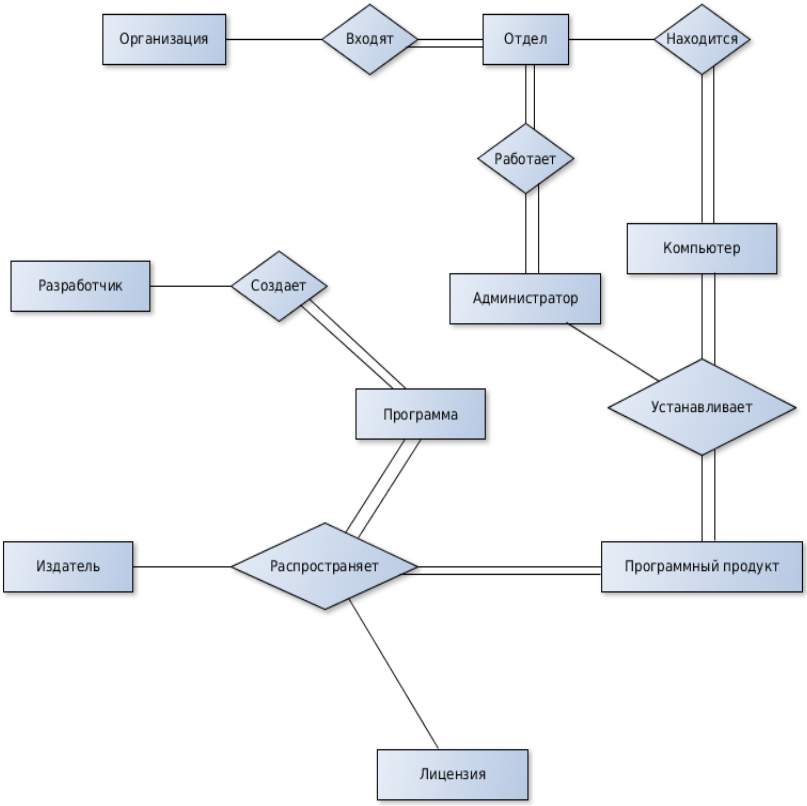

Рис. 4. ER–диаграмма предметной области “Систему учета установленного ПО в организациях”

5.7 Схемы базы данных

Лисенкова Касимов

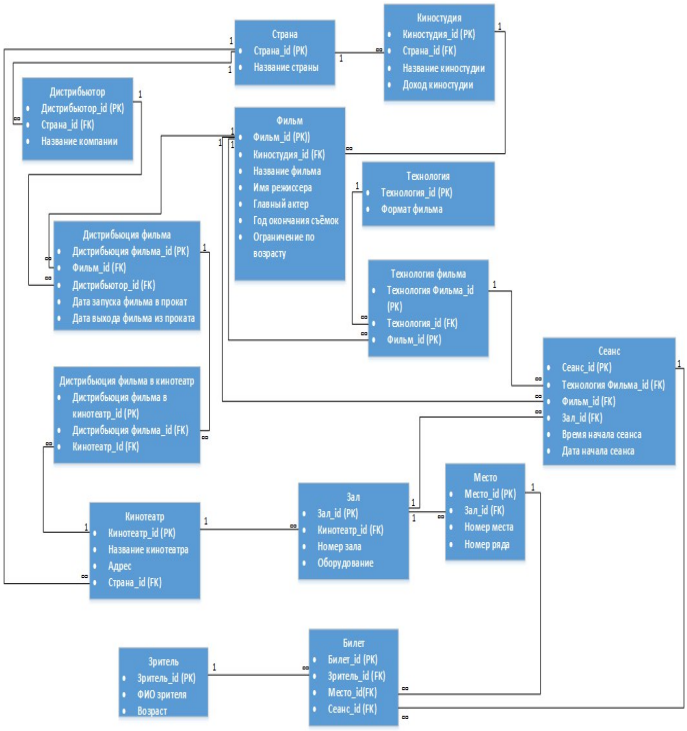

Рис. 5. Схема база данных “Выставка собак”

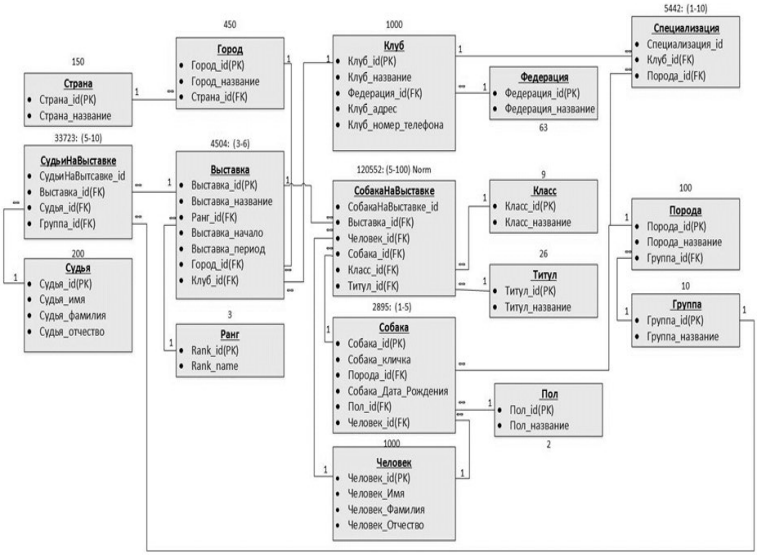

Рис. 6. Схема база данных “Кинодистрибьютор”

5.8 Описание распределения данных

Предметная область “Расписание экзаменов” Дементьев
Каждая комиссия состоит из одного-трёх (1-3) преподавателей.
Каждый преподаватель читает от трёх до пяти (3-5) дисциплин.
Каждая группа сдает от трёх до пяти (3-5) экзаменов.
Каждая комиссия принимают каждую дисциплину.

Таблица	Количество записей
Факультет	59
Кафедра	59
Специальность	59
Комиссия	59
Дисциплина	59
Аудитория	59
Время экзамена	9
Преподаватель	59
Специализация	238
Состав комиссии	121
Состав дисциплин комиссии	3481
Расписание сессии	100000
Поток	400424
Группа	100000

5.9 Генерация исходных данных

5.10 Реализация и описание результатов запросов

5.10.1 Описание запроса на естественном языке

Запрос №1. Предметная область "Кинодистрибьютор" Касимов

Вывести название кинотеатров, в которых демонстрируется фильм «A» в технологии «B» в зале «C» (A,B,C выбирались на собственное усмотрение).

Запрос №2. Предметная область "Кинодистрибьютор" Касимов

Вывести название кинотеатров, в которых демонстрируется фильм «A» в технологии «B» в зале «C», у которых продано «n» билетов. (A,B,C выбирались на собственное усмотрение).

Запрос №3. Предметная область "Выставка собак" Лисенкова

Запрос №4. Предметная область "Выставка собак" Лисенкова

5.10.2 Текст запроса на языке SQL

Запрос №1. Предметная область "Кинодистрибьютор" Касимов

```
SELECT Cinema.Cinema_name
```

```
  FROM Cinema
  JOIN Hall ON Hall.Cinema_id = Cinema.Cinema_id
  JOIN SessionDB ON SessionDB.Hall_id = Hall.Hall_id
  JOIN FilmTechnology ON
       FilmTechnology.FilmTechnology_id =
       SessionDB.FilmTechnology_id
 WHERE (FilmTechnology.Technology_id = 9 AND
        FilmTechnology.Film_id = 486 AND
        Hall.Number_Hall = 1);
```

Запрос №2. Предметная область "Кинодистрибьютор" Касимов

```
SELECT Cinema.Cinema_name
  FROM (
SELECT Hall.Cinema_id as numberCinema
  FROM (
SELECT tableHallAndTicket.numberHall,
       tableHallAndTicket.numberTickets
  FROM (
SELECT checkHallAndSession.numberHall,
       checkHallAndSession.numberSession,
 COUNT (Ticket.Ticket_id) as numberTickets
  FROM (
SELECT SessionDB.Hall_id as numberHall,
       SessionDB.Session_id as numberSession
  FROM (
SELECT FilmTechnology.FilmTechnology_id as numberFilm
  FROM FilmTechnology
 WHERE (Technology_id = 9 AND Film_id = 486))\
        as checkFilmTechnology
  JOIN SessionDB ON checkFilmTechnology.numberFilm
       = SessionDB.FilmTechnology_id)
        as checkHallAndSession
  JOIN Ticket  ON Ticket.Session_id =
       checkHallAndSession.numberSession
GROUP BY checkHallAndSession.numberSession)
       as tableHallAndTicket
 WHERE tableHallAndTicket.numberTickets = 15)
```

```
      as tableForHall
  JOIN Hall ON tableForHall.numberHall = Hall.Hall_id
 WHERE Hall.Number_Hall = 4) as checkCinema
  JOIN Cinema ON checkCinema.numberCinema =
      Cinema.Cinema_id;
```

5.10.3 Реляционная формула запроса

5.10.4 Результаты выполнения запроса

Запрос №1. Предметная область "Кинодистрибьютор" Касимов

```
+-------------+
| Cinema_name |
+-------------+
| field       |
| rice        |
| fish        |
| promote     |
| spirit      |
+-------------+
5 rows in set (0,00 sec)
```

Запрос №2. Предметная область "Кинодистрибьютор" Касимов

```
+-------------+
| Cinema_name |
+-------------+
| available   |
+-------------+
1 row in set (0,02 sec)
```

5.10.5 Вычисление характеристик распределения данных

5.10.6 Построение графиков и диаграмм

5.11 Выводы

Выводы

В ходе выполнения работы был проведён анализ предметной области, в результате которого была составлена ER-диаграмма, а также схема объектов, которая содержит 3 связи "многие ко многим"и 5 связей "один ко многим". После этого была спроектирована схема базы данных, которая отражает связи между таблицами, используя foreign и primary keys, содержит 53 атрибута, 17 таблиц (7 из которых являются словарями), промежуточные таблицы для организации связей "многие ко многим а также распределения значений атрибутов. Также, были написаны программы на языке Java для заполнения данных в таблице. Для того чтобы проверить корректность созданной базы данных, было реализовано 7 запросов. Анализируя explain этих запросов можно сделать следующие выводы:

1. Использованные типы запросов: SIMPLE (простой без подзапросов), PRIMARY (самый внешний запрос в join), DERIVED (часть подзапроса внутри from).
2. Типы связей таблиц: system (таблица имеет только одну строку), eq_ref(все части индекса используются для связывания), ref (все соответствующие строки индексного столбца считываются для каждой комбинации строк из предыдущей таблицы), index (сканируется всё дерево индексов для нахождения соответствующих строк), all (для нахождения соответствующих строк используется сканирование всей таблицы).

Время выполнения запросов зависит от количества подзапросов, а также от типа связей таблиц. Наихудшим типом связи является ALL, однако иногда без его использования не обойтись. Кроме этого по результатам запросов было построено два графика (2-х мерный и 3-х мерный). Следует отметить, что трёхмерный график представляет собой не содержит острых пиков.

Вывод (Клюшкин)

В качестве задания мне было предложена разработка билинговой системы сотового оператора. Для этого была изучена предметная область данной темы. Выявленные главные составляющие этой системы: 8 сущностей, таких как абонент, договор, лицевой счет, пополнение, тарифный план, операция, расход, тарифный план по договору. Были проанализированы и установлены типы данных, используемые в моей задаче. Это int,char,float,data и тд. Построена схема иерархий, позволяющая проналюдать процесс взаимодействий сущностей в пределах моей темы. Она включает в себя 2 связи многие ко многим и 2 тернарные связи. Далее

была разработана ЕК-диаграмма более наглядно показывающая процесс работы билинговой системы, так как на схеме указаны все атрибуты для каждой сущности. Основываясь на полученной информации, была спроектирована схема базы данных на русском и английском языках. Схема содержит 10 таблиц 2 из которых для раскрытия связи многие ко многим, а так же указаны способы соединения их. Были проанализированы и установлены типы данных, используемые в моей задаче. По разработанной схеме был написан 50.1–код создающий базу данных билинговой системы длиной 120 строк. Для отладки и проверки был написан генератор тестовых данных. Размер генератора — 370 строк. Генератор создает случайные значения для всех таблиц БД. И на заключительном этапе были написаны 9 запросов к базе данных, позволяющих получать некую полезную и правильную информацию за удобное для пользователя время. Запросы были разные по сложности и на разные темы: поиск минимальных и максимальных значений,самопересечения, построение зависимостей одного атрибута от другого, декартовы произведения. По результатам 3–х запросов были построены графики. По каждому запросу был рассмотрен EXPLAIN, что позволило точнее оценить эффективность. Так же каждый запрос был описан с точки зрения реляционной алгебры Для выполнения этих заданий были использованы:

1. MicrosoftVisualStudio 2010 На языке C++ был написан генератор данных для БД.
2. Mysql 5.5.28-win32 35 На основе данного дистрибутива была написана БД и выполнены запросы.
3. Notepad++ — Текстовый редактор с подсветкой синтаксиса и умением работы с кодировкой был использован для более удобного программирования на SQL.

На выполнение всей работы был потрачен целый семестр. Выполнить поставленную задачу за более короткий срок не представляется возможным, так как анализ, проектирование и написание кода — весьма трудоёмкие задачи, требующие усердия и внимательности. В процессе работы возникали трудности рабочего плана, такие как ошибки кодировки, проектирования и прочие. Проблемы решались довольно быстро благодаря большому количеству полезной информации, полученной на лекциях, а так же в документации на официальном сайте разработчика MySql. К примеру, вопрос с кодировкой решился, как только было обращено внимание на следующие факты: БД должна создаваться в нужной кодиров-

ке(на Windows это cp1251), файл, из которого происходит запись данных тоже должен быть в такой же кодировке, как и БД, а так же должен иметь разрешение sql. Проделав всю данную работу, я научился анализировать предметную область, для которой нужно создать БД. Научился разрабатывать схемы БД и писать по ним код создания БД. Так же я изучил основные типы запросов и синтаксис языка SQL. Выводы В ходе выполнения работы был проведён анализ предметной области, в результате которого была составлена ER–диаграмма, а также схема объектов.

Была спроектирована схема базы данных, которая отражает связи между таблицами, используя первичные и внешние ключи. Спроектированная схема содержит:

– 41 атрибут
– 18 таблиц, среди которых 10 словарей
– Промежуточная таблица для организации связи "многие ко многим"

Была написана программа на языке Python для генерации sql–файла, содержащего запросы для заполнения таблиц базы данных. Кроме того, было реализовано 6 запросов. Анализируя explain этих запросов можно сделать следующие выводы:

1. Использованные типы запросов:
   – PRIMARY — внешний запрос
   – DERIVED — подзапросв "FROM"
   – SUBQUERY — первый SELECT в подзапросе
   – DEPENDENT SUBQUERY - первый SELECT в подзапросе, зависящий от внешнего запроса
2. Использованные типы связей таблиц
   – ALL — полный проход по всем записям таблицы
   – eq_ref — все части индекса используются для связывания
   – ref — все соответствующие строки индексного столбца считываются для каждой комбинации строк из предыдущей таблицы
   – range — используются строки таблицы из заданного диапазона
   – index — сканируется всё дерево индексов для нахождения соответствующих строк

Время выполнения запросов зависит от количества подзапросов, а также от типа связей таблиц. Наихудшим типом связи является ALL.

Также по результатам запросов было построено два графика — 2-х и 2–х мерный.

5.12 Приложения

5.12.1 Приложение А. Программа создания схемы базы данных

Код создания базы данных. Предметная область "Расписание экзаменов" Дементьев

```
drop database Schedule;create database Schedule;
use Schedule;

CREATE TABLE discipline (
‘id_discipline‘ INT(3),
‘name_discipline‘ VARCHAR(25) NOT null,
‘code_discipline‘ INT(3) NOT null,
‘type_disciplin‘ VARCHAR(25) NOT null,
PRIMARY KEY(id_discipline));

CREATE TABLE educator
(‘id_educator‘ INT(3),
‘surname‘ VARCHAR(25) NOT NULL,
‘name‘ VARCHAR(25) NOT NULL,
‘patronymic‘ VARCHAR(25) NOT NULL,
‘faculty‘ VARCHAR(25),
‘science_degree‘ VARCHAR(25),
‘position‘ VARCHAR(25),
‘appointment‘ VARCHAR(25),
PRIMARY KEY(id_educator));

CREATE TABLE specialization
(‘id_discipline‘ INT(3),
‘id_educator‘ INT(3),
PRIMARY KEY(id_discipline,id_educator))
CREATE TABLE commission_structure
(‘id_commission‘ INT(6),
‘id_educator‘ INT(3) ,
‘status‘ VARCHAR(25) NOT NULL,
PRIMARY KEY(id_commission,id_educator));

CREATE TABLE commission
```

```
(‘id_commission‘ INT(6),
PRIMARY KEY(id_commission));

CREATE TABLE ‘group‘
(‘id_group‘ INT(6),
‘amount_people‘ INT(3) NOT NULL,
‘year_set‘ INT(4) NOT NULL,
‘form_of_education‘ VARCHAR(25) NOT NULL,
‘contact_person‘ VARCHAR(25) NOT NULL,
‘year_of_issue‘ INT(4),
‘id_specialty‘ INT(3) NOT NULL,
‘id_chair‘ INT(3) NOT NULL,
‘id_department‘ INT(3) NOT NULL,
PRIMARY KEY (id_group));

CREATE TABLE flow
(‘id_group‘ INT(6),
‘id_exam‘ INT(6),
‘number_of_groups‘ INT(3),
 PRIMARY KEY(id_group,id_exam));

CREATE TABLE exam_time
(‘id_period‘ INT(3),
‘exam_start‘ VARCHAR(25) NOT NULL,
‘exam_end‘ VARCHAR(25),
 PRIMARY KEY(id_period));

CREATE TABLE schedule_session
(‘id_exam‘ INT(6),
‘id_commission‘ INT(6) NOT NULL,
‘id_auditorium‘ INT(3),
‘id_period‘ INT(3) NOT NULL,
‘date_exam‘ DATE,
 PRIMARY KEY(id_exam));

CREATE TABLE auditorium
(‘id_auditorium‘ INT(3),
```

```
‘type_auditorium‘ VARCHAR(25) NOT NULL,
‘housing‘ VARCHAR(25) NOT NULL,
‘floor‘ INT(3) NOT NULL,
‘auditorium‘ INT(3) NOT NULL,
‘number_of_seats‘ INT(3),
 PRIMARY KEY(id_auditorium));

CREATE TABLE specialty
(‘id_specialty‘ INT(3),
 ‘name_specialty‘ VARCHAR(25) NOT NULL,
PRIMARY KEY(id_specialty));

CREATE TABLE chair
(‘id_chair‘ INT(3),
‘name_chair‘ VARCHAR(25) NOT NULL,
‘director‘ VARCHAR(25) NOT NULL,
 ‘deans_office‘ VARCHAR(25),
 ‘contact_number‘ INT(15),
 PRIMARY KEY(id_chair));

CREATE TABLE department
(‘id_department‘ INT(3),
 ‘name_department‘ VARCHAR(25) NOT NULL,
‘contact_number‘ INT(15),
‘location‘ VARCHAR(25),
‘head_of_department‘ VARCHAR(25),
PRIMARY KEY(id_department));

CREATE TABLE commission_disciplines
 (‘id_discipline‘ INT(3),
‘id_commission‘ INT(6),
 PRIMARY KEY(id_discipline,id_commission));


ALTER TABLE ‘group‘ ADD FOREIGN KEY(id_specialty)
 REFERENCES specialty (id_specialty);
ALTER TABLE ‘group‘ ADD FOREIGN KEY(id_chair)
```

```
 REFERENCES chair (id_chair);
ALTER TABLE `group` ADD FOREIGN KEY (id_department)
 REFERENCES department (id_department);
ALTER TABLE flow ADD FOREIGN KEY(id_group)
 REFERENCES `group` (id_group);
ALTER TABLE flow ADD FOREIGN KEY(id_exam)
 REFERENCES schedule_session (id_exam);
ALTER TABLE commission_structure ADD FOREIGN KEY
(id_educator)
 REFERENCES educator (id_educator);
ALTER TABLE commission_structure ADD FOREIGN KEY
(id_commission)
 REFERENCES commission (id_commission);
ALTER TABLE commission_disciplines ADD FOREIGN KEY
(id_commission)
 REFERENCES commission (id_commission);
ALTER TABLE commission_disciplines ADD FOREIGN KEY
(id_discipline)
 REFERENCES discipline (id_discipline);
ALTER TABLE schedule_session ADD FOREIGN KEY
(id_commission)
 REFERENCES commission (id_commission);
ALTER TABLE schedule_session ADD FOREIGN KEY
(id_period)
 REFERENCES exam_time (id_period);
ALTER TABLE schedule_session ADD FOREIGN KEY
(id_auditorium)
 REFERENCES auditorium (id_auditorium);
ALTER TABLE specialization ADD FOREIGN KEY
(id_educator)
 REFERENCES educator (id_educator);
ALTER TABLE specialization ADD FOREIGN KEY
(id_discipline)
 REFERENCES discipline (id_discipline);
```

Код создания базы данных. Предметная область “Администрирование зоопарками” Микулик

```
DROP DATABASE IF EXISTS Zoo;
CREATE DATABASE Zoo DEFAULT CHARSET='cp1251';
USE Zoo;

CREATE TABLE Country(
country_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
name char(50) NOT NULL,
capital char(50) NOT NULL,
population int NOT NULL,
official_language char (50) NOT NULL,
currency char(50) NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE City(
city_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
name char(50) NOT NULL,
time_zone char(10) NOT NULL,
mayor char (100) NOT NULL,
country_id int NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Zoo(
zoo_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
name char(50) NOT NULL,
location char(100) NOT NULL,
hours char(100) NOT NULL,
website char(100) NOT NULL,
phone char(20) NOT NULL,
city_id int NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Place_of_detention(
pd_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
number int NOT NULL,
```

```
name char(60) NOT NULL,
area int NOt NULL,
department_id int NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Department(
department_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
name char(50) NOT NULL,
zoo_id int NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Species(
species_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
name char (50) NOT NULL,
scientific_name char (50) NOT NULL,
way_of_eating_id int NOT NULL,
conservation_status_id int NOT NULL,
activity_time char (20) NOT NULL,
habitat char (100) NOT NULL,
temperature_min int NOT NULL,
temperature_max int NOT NULL,
class_id int NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Class(
class_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
name char (20) NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Animal(
animal_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
name char (30) NOT NULL,
weight int NOT NULL,
```

```
growth int NOT NULL,
date_of_birth date NOT NULL,
sex char (10) NOT NULL,
feeding_time char (40) NOT NULL,
exposition boolean NOT NULL,
temporary boolean NOT NULL,
species_id int NOT NULL,
PD_id int NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Custodian(
custodian_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
name char (50) NOT NULL,
surname char (50) NOT NULL,
education_id int NOT NULL,
specialty_id int NOT NULL,
date_of_birth date NOT NULL,
sex char(10) NOT NULL,
passport_id int NOT NULL,
phone int NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Availability(
availability_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
department_id int NOT NULL,
species_id int NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Welfare(
welfare_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
species_id int NOT NULL,
custodian_id int NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';
```

```
CREATE TABLE Care(
care_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
animal_id int NOT NULL,
custodian_id int NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Education(
education_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
name char (40) NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Specialty(
specialty_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
name char (50) NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Way_of_eating(
way_of_eating_id int PRIMARY KEY AUTO_INCREMENT NOT NULL,
name char (50) NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

CREATE TABLE Conservation_status(
conservation_status_id int PRIMARY KEY AUTO_INCREMENT
NOT NULL,
name char (4) NOT NULL
)
ENGINE=InnoDB DEFAULT CHARSET='cp1251';

ALTER TABLE City ADD FOREIGN KEY (country_id) REFERENCES
 Country (country_id);
ALTER TABLE Zoo ADD FOREIGN KEY (city_id) REFERENCES
 City (city_id);
```

```
ALTER TABLE Place_of_detention ADD FOREIGN KEY
 (department_id) REFERENCES Department (department_id);
ALTER TABLE Department ADD FOREIGN KEY (zoo_id)
 REFERENCES Zoo(zoo_id);
ALTER TABLE Availability ADD FOREIGN KEY (department_id)
 REFERENCES Department(department_id);
ALTER TABLE Availability ADD FOREIGN KEY (species_id)
 REFERENCES Species(species_id);
ALTER TABLE Species ADD FOREIGN KEY (class_id)
 REFERENCES Class(class_id);
ALTER TABLE Species ADD FOREIGN KEY (way_of_eating_id)
 REFERENCES Way_of_eating (way_of_eating_id);
ALTER TABLE Species ADD FOREIGN KEY
 (conservation_status_id) REFERENCES
 Conservation_status (conservation_status_id);
ALTER TABLE Animal ADD FOREIGN KEY (species_id)
 REFERENCES Species (species_id);
ALTER TABLE Animal ADD FOREIGN KEY (pd_id)
 REFERENCES Place_of_detention (pd_id);
ALTER TABLE Welfare ADD FOREIGN KEY (species_id)
 REFERENCES Species (species_id);
ALTER TABLE Welfare ADD FOREIGN KEY (custodian_id)
 REFERENCES Custodian (custodian_id);
ALTER TABLE Care ADD FOREIGN KEY (custodian_id)
 REFERENCES Custodian (custodian_id);
ALTER TABLE Care ADD FOREIGN KEY (animal_id)
 REFERENCES Animal (animal_id);
ALTER TABLE Custodian ADD FOREIGN KEY (education_id)
 REFERENCES Education(education_id);
ALTER TABLE Custodian ADD FOREIGN KEY (specialty_id)
 REFERENCES Specialty(specialty_id);
```

5.12.2 Приложение Б. Программа генерации данных

Код программы заполнения базы данных. Предметная область “Расписание экзаменов” Дементьев

```
1. #include <QCoreApplication>
```

```
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <string.h>
5. #include <iostream>
6. #include <vector>
7. #include <map>
8. #include <list>
9. #include <fstream>
10. #include <conio.h>
11. #include <cmath>
12. #include <ctime>
13. #include <QString>
14. #include <QVector>
15. #include <QFile>
16. #include <QTextStream>
17. #include <random>
18. using namespace std;
19. int main(int argc, char *argv[])
20. {
21.     QCoreApplication a(argc, argv);
22.     QVector <QString> direktor;
23.     direktor.push_back("Алмазова");
24.     direktor.push_back("Ватин");
25.     direktor.push_back("Макаров");
26.     direktor.push_back("Сильников");
27.     direktor.push_back("Колесников");
28.     direktor.push_back("Заборовский");
29.     direktor.push_back("Боровков");
30.     direktor.push_back("Фролов");
31.     direktor.push_back("Забелин");
32.     direktor.push_back("Щепинин");
33.     QVector <QString> student;
34.     student.push_back("Дементьев");
35.     student.push_back("Сидоров");
36.     student.push_back("Востров");
37.     student.push_back("Кацал");
38.     student.push_back("Медведев");
```

```
39.     student.push_back("Богославец");
40.     student.push_back("Кладиков");
41.     student.push_back("Малышев");
42.     student.push_back("Микулик");
43.     student.push_back("Лисенкова");
44.     QVector<QString> adress;
45.     adress.push_back("Пушкина 12");
46.     adress.push_back("Гагарина 21");
47.     adress.push_back("Сталина 51");
48.     adress.push_back("Бутлерова 9");
49.     adress.push_back("Науки 22");
50.     adress.push_back("Ленина 55");
51.     adress.push_back("Горького 342");
52.     adress.push_back("Губкина 22");
53.     adress.push_back("Муллагалиева 1");
54.     QVector<QString> instityt;
55.     instityt.push_back("ИКНТ");
56.     instityt.push_back("ИПММ");
57.     instityt.push_back("ИИУС");
58.     instityt.push_back("ИМОП");
59.     instityt.push_back("ИСИ");
60.     instityt.push_back("ИПМЭИТ");
61.     QVector<QString> kafedra;
62.     kafedra.push_back("Телематика");
63.     kafedra.push_back("СЛАУ");
64.     kafedra.push_back("КИТ");
65.     kafedra.push_back("Реклама");
66.     kafedra.push_back("Филология");
67.     kafedra.push_back("Гидравлика");
68.     QVector<QString> typeaud;
69.     typeaud.push_back("Компьютерная");
70.     typeaud.push_back("Лекционная");
71.     typeaud.push_back("Для Лаб");
72.     typeaud.push_back("Лолкек");
73.     QVector<QString> housing;
74.     housing.push_back("Главное здание");
75.     housing.push_back("1 корпус");
```

```
76.     housing.push_back("2 корпус");
77.     housing.push_back("11 корпус");
78.     housing.push_back("9 корпус");
79.     housing.push_back("Хим корпус");
80.     housing.push_back("3 корпус");
81.     housing.push_back("4 корпус");
82.     housing.push_back("5 корпус");
83.     housing.push_back("Гидрокорпус");
84.     housing.push_back("6 корпус");
85.     housing.push_back("7 корпус");
86.     QVector<QString> stepen;
87.     stepen.push_back("Доцент");
88.     stepen.push_back("Профессор");
89.     stepen.push_back("Кандидат наук");
90.     stepen.push_back("Доктор");
91.     QVector<QString> doljnost;
92.     doljnost.push_back("Зам кафедры");
93.     doljnost.push_back("Никто");
94.     doljnost.push_back("Зам зав кафедры");
95.     doljnost.push_back("Некто");
96.     QVector<QString> zvanie;
97.     zvanie.push_back("Не понял");
98.     zvanie.push_back("Лаборант");
99.     zvanie.push_back("Как так");
100.     zvanie.push_back("Логика");
101.     QVector<QString> PrepFam;
102.     PrepFam.push_back("Курочкин");
103.     PrepFam.push_back("Новицкий");
104.     PrepFam.push_back("Кекович");
105.     PrepFam.push_back("Иванов");
106.     PrepFam.push_back("Петров");
107.     PrepFam.push_back("Сидоров");
108.     PrepFam.push_back("Касимов");
109.     QVector<QString> Name;
110.     Name.push_back("Василий");
111.     Name.push_back("Миша");
112.     Name.push_back("Денис");
```

```
113.     Name.push_back("Сергей");
114.     Name.push_back("Семен");
115.     Name.push_back("Максим");
116.     Name.push_back("Герман");
117.     Name.push_back("Степан");
118.     Name.push_back("Илья");
119.     Name.push_back("Александр");
120.     QVector<QString> Otchestvo;
121.     Otchestvo.push_back("Михайлович");
122.     Otchestvo.push_back("Семеныч");
123.     Otchestvo.push_back("Валерьевич");
124.     Otchestvo.push_back("Иванович");
125.     Otchestvo.push_back("Петрович");
126.     Otchestvo.push_back("Александрович");
127.     Otchestvo.push_back("Сергеевич");
128.     QVector<QString> predmet;
129.     predmet.push_back("Матан");
130.     predmet.push_back("Физика");
131.     predmet.push_back("История");
132.     predmet.push_back("Базы данных");
133.     predmet.push_back("Логика");
134.     predmet.push_back("Физкультура");
135.     predmet.push_back("Геометрия");
136.     predmet.push_back("Микроконтроллеры");
137.     predmet.push_back("Графика");
138.     predmet.push_back("Химия");
139.     predmet.push_back("Статистика");
140.     QVector<QString> typedisc;
141.     typedisc.push_back("Общая");
142.     typedisc.push_back("Специализированная");
143.     QVector<QString> special;
144.     special.push_back("Математика и комп науки");
145.     special.push_back("Мат обеспечение");
146.     special.push_back("Реклама и тд");
147.     special.push_back("Мехатроника");
148.     special.push_back("Роботехника");
149.     QVector<QString> vremya;
```

```
150.     vremya.push_back("10:00',’12:00");
151.     vremya.push_back("11:00',’13:00");
152.     vremya.push_back("12:00',’15:00");
153.     vremya.push_back("13:00',’15:00");
154.     vremya.push_back("14:00',’16:00");
155.     vremya.push_back("10:30',’12:00");
156.     vremya.push_back("11:30',’13:30");
157.     vremya.push_back("12:30',’14:30");
158.     vremya.push_back("14:30',’16:00");
159.     QVector<QString> formaOby;
160.     formaOby.push_back("Очная");
161.     formaOby.push_back("Заочная");
162.     formaOby.push_back("Очная вечерняя");
163.     QVector<QString> status;
164.     status.push_back("Лаборант");
165.     status.push_back("Лектор");
166.           QFile file("CREATE_AND_INSERT.txt");
167.     if (!file.open(QIODevice::WriteOnly
 | QIODevice::Text))
168.         return 0;
169.     QTextStream out(&file);
170.     out.setCodec("UTF-8");
171.     int fakyltet=59;
172.     out<<"INSERT INTO chair
(id_chair,name_chair  ,director , deans_office,
 contact_number ) VALUES";
173.
174.     //Факультет
175.        for(int i=0;i<fakyltet;i++){
176.     out <<"('"<<i<<"',’"<<instityt[qrand()%
instityt.size()]<<"',’"<<direktor[qrand()%direktor.size()]
<<"',’"<<qrand()%400+100<<"k"<<"',’"
<<(qrand()%1000000+2000000)<<"')";
177.     if (i<fakyltet-1)
178.     out<<",";
179.     }
180.     out<<";\n";
```

```
181.
182.      //Кафедра
183.      int kaf=59;
184.     out<<"INSERT INTO department
(id_department,name_department,
 contact_number,location, head_of_department) VALUES";
185.     for(int i=0;i<kaf;i++){
186.
187.         out <<"('"<<i<<"','"<<kafedra[qrand()%kafedra.si
<<"','"<<qrand()%1000000+3000000<<"','"<<adress[qrand()
%adress.size()]<<"','"<<direktor[qrand()%direktor.size()]<<"'
188.         if (i<kaf-1)
189.             out<<",";
190.     }
191.     out<<";\n";
192.
193.     //Аудитория
194.   int audit=59;
195.
196.   out<<"INSERT INTO auditorium
(id_auditorium , type_auditorium ,housing,
 floor, auditorium , number_of_seats) VALUES ";
197.     for(int i=0;i<audit;i++){
198.         out <<"('"<<i<<"','"<<typeaud[(qrand()%typeaud.s
<<"','"<<housing[(qrand()%housing.size())]
<<"','"<<qrand()%4+1<<"','"<<qrand()%500+100
<<"','"<<qrand()%100+1<<"')";
199.         if (i<audit-1)
200.             out<<",";
201.     }
202.     out<<";\n";
203.     //Преподаватель
204.     out<<"INSERT INTO educator
 ( id_educator , surname , name,
 patronymic , faculty , science_degree , position,
 appointment) VALUES";
205.     int prepod=59;
```

```
206.     for(int i=0;i<prepod;i++){
207.         out<<"('"<<i<<"','"<<PrepFam[(qrand()%PrepFam.si
<<"','"<<Name[qrand()%Name.size()]<<"','"<<Otchestvo[qrand()
%Otchestvo.size()]<<"','"<<kafedra[qrand()%kafedra.size()]
<<"','"<<stepen[qrand()%stepen.size()]<<"','"<<zvanie[qrand()
%zvanie.size()]<<"','"<<doljnost[qrand()%doljnost.size()]<<"'
208.         if (i<prepod-1)
209.             out<<",";
210.     }
211.     out<<";\n";
212.
213.     //Дисциплина
214.     out<<"INSERT INTO discipline
(id_discipline , name_discipline , code_discipline ,
 type_disciplin) VALUES ";
215.     int disc=59;
216.     for(int i=0;i<disc;i++){
217.
218.
219.         out <<"('"<<i<<"','"<<predmet[(qrand()%predmet.s
<<"','"<<(qrand()%9000+10000)<<"','"
<<typedisc[qrand()%typedisc.size()]<<"')";
220.         if (i<disc-1)
221.             out<<",";
222.     }
223.     out<<";\n";
224.
225.     //Специальность
226.     int spec=59;
227.     out<<"INSERT INTO specialty (id_specialty ,
  name_specialty) VALUES ";
228.     for(int i=0;i<spec;i++){
229.
230.
231.
232.         out<<"('" <<i<<"','"<<special[(qrand()
%special.size())]<<"')";
```

```
233.         if (i<spec-1)
234.             out<<",";
235.     }
236.     out<<";\n";
237.
238.     //Время экз
239.     out<<"INSERT INTO exam_time
( id_period , exam_start , exam_end) VALUES ";
240.     for(int i=0;i<vremya.size();i++){
241.
242.
243.         out<<"('" <<i<<"','"<<vremya[i]<<"')";
244.         if (i<vremya.size()-1)
245.             out<<",";
246. }
247.     out<<";\n";
248.
249.     //Комиссия
250.     out<<"INSERT INTO  commission (id_commission) VALUES
251.     int kommis=59;
252.     for(int i=0;i<kommis;i++){
253.         out<<"('" <<i<<"')";
254.         if (i<kommis-1)
255.             out<<",";
256.     }
257.     out<<";\n";
258.
259.     //Группа
260.
261.     int grypa=100000;
262.     out<<"INSERT INTO `group`
(id_group ,amount_people ,
 year_set , form_of_education ,
 contact_person , year_of_issue ,id_specialty ,
 id_chair , id_department) VALUES ";
263.     for(int i=0;i<grypa;i++){
264.         out<<"('" <<i<<"','"<<(qrand()%20+10)<<"','"<<qr
```

```
<<formaOby[qrand()%formaOby.size()]<<"','"<<student[qrand()
%student.size()]<<"',"<<qrand()%8+2016<<"',"<<qrand()
%spec<<"',"<<qrand()%fakyltet<<"',"<<qrand()%kaf<<"')";
265.          if (i<grypa-1)
266.              out<<",";
267.      }
268.      out<<";\n";
269.
270.         //Специализация
271.
272.      out<<"INSERT INTO specialization
(id_discipline , id_educator) VALUES ";
273.      for(int i=0;i<59;i++){
274.
275.      int max1=qrand()%3+3;
276.      int ed[max1];
277.      for(int j=0;j<max1;j++){
278.
279.          ed[j]=qrand()%59;
280.          for(int k=0;k<max1;k++){
281.              if (j!=k){
282.                  if (ed[j] == ed[k])
283.                      ed[j] = qrand()%59, k = 0;
284.              }
285.          }
286.          out<<"('" <<i<<"','"<<ed[j]<<"')";
287.          if (j<max1-1 | i<59-1)
288.              out<<",";
289.      }
290. }
291. //
292. out<<";\n";
293.
294.      //Состав комиссии
295.      out<<"INSERT INTO  commission_structure
 ( id_commission , id_educator , status) VALUES ";
296.      for(int i=0;i<59;i++){
```

```
297.
298.     int max2=qrand()%3+1;
299.     int prep[max2];
300.     for(int j=0;j<max2;j++){
301.
302.         prep[j]=qrand()%59;
303.         for(int k=0;k<max2;k++){
304.             if (j!=k){
305.                 if (prep[j] == prep[k])
306.
307.                     prep[j] = qrand()%59, k = 0;
308.             }
309.         }
310.         out <<"('"<<i<<"','"<<prep[j]<<"','"<<status[qra
%status.size()]<<"')";
311.         if (i<59-1 | j<max2-1)
312.             out<<",";
313.     }
314. }
315.      out<<";\n";
316.
317.     //Состав дисциплин комиссии
318.      out<<"INSERT INTO commission_disciplines
 ( id_discipline ,id_commission) VALUES ";
319.     for(int i=0;i<59;i++){
320.         for(int j=0;j<59;j++){
321.             out<<"('" <<i<<"','"<<j<<"')";
322.             if (j<59-1 | i<59-1)
323.                 out<<",";
324.         }}
325.      out<<";\n";
326.
327.     //Расписание сессии
328.       int exam=100000;
329.     out<<"INSERT INTO schedule_session
 (id_exam , id_commission , id_auditorium ,
 id_period , date_exam) VALUES ";
```

```
330.     for(int i=0;i<exam;i++){
331.
332.         //int dekanat=qrand()%300+100;
333.         out<<"('" <<i<<"',"<<(qrand()%kommis)<<"',"<<q
%audit<<"',"<<qrand()%vremya.size()<<"',"<<"2016-"<<qrand()
%12+1<<"-"<<qrand()%29+1<<"')";
334.         if (i<exam-1)
335.             out<<",";
336.     }
337.     out<<";\n";
338.       //Поток
339.     out<<"INSERT INTO flow
(id_group , id_exam , number_of_groups) VALUES ";
340.     for(int i=0;i<100000;i++){
341.
342.         int max=qrand()%3+3;
343.          int ekzam[max];
344.         for(int j=0;j<max;j++){
345.
346.             ekzam[j]=qrand()%100000;
347.             for(int k=0;k<max;k++){
348.                 if (j!=k){
349.                     if (ekzam[j] == ekzam[k])
350.
351.                         ekzam[j] = qrand()%100000, k = 0
352.                 }
353.             }
354.             out<<"('" <<i<<"',"<<ekzam[j]<<"',"<<max<<
355.             if (j<max-1 | i<100000-1)
356.                 out<<",";
357.         }
358.     }
359.     out<<";";
360.     file.close();
361.     return a.exec();
}
```