Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ТЕХНОЛОГИИ БАЗ ДАННЫХ

Методические указания
к выполнению лабораторных работ
для студентов, обучающихся по направлению 09.03.02
«Информационные системы и технологии»
по учебному плану подготовки бакалавров
дневной и заочной форм обучения

Севастополь
СевГУ
2018

УДК 004.65 (075.8)

Р е ц е н з е н т:
Е.Н. Мащенко - канд. техн. наук, доцент кафедры ИТиКС

Составители: Ю.В. Доронина, И.В. Дымченко, А.В. Волкова

Технологии баз данных: метод. указания к выполнению лабораторных работ / Сост. Ю.В. Доронина, И.В. Дымченко, А.В. Волкова. – Севастополь: СевГУ, 2018. – 60 с.

Методические указания предназначены для проведения лабораторных работ по дисциплине «Технологии баз данных». Целью методических указаний является помощь студентам в изучении процесса построения логической и физической моделей баз данных, возможностей и условий применения СУБД, реализующих поддержку пространственных данных на примере PostgreSQL, в разработке интерфейса баз данных с использованием различного программного обеспечения и в получении навыков работы с инструментом MySQL Workbench, а также по настройке удаленного подключения к серверу баз данных. Излагаются теоретические и практические сведения необходимые для выполнения лабораторных работ, требования к содержанию отчета.

УДК 004.65 (075.8)

Методические указания рассмотрены и утверждены на заседании кафедры информационных систем и методическом семинаре института информационных технологий и управления в технических системах, протокол № 10 от 18 мая 2018 г.

СОДЕРЖАНИЕ

Лабораторная работа № 1. Анализ и исследование предметной области, подлежащей информатизации 4
Лабораторная работа № 2. Организации архитектуры «клиент-сервер» в системах баз данных. Построение полной атрибутивной модель базы данных в нотации IDEF1X 25
Лабораторная работа № 3. Исследование процессов построения информационных моделей в системе управления базами данных PostgreSQL43 43
Лабораторная работа № 4. Создание программного приложения для работы с базой данных. Тестирование базы данных 54
Список использованных источников 60

Лабораторная работа № 1

АНАЛИЗ И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ, ПОДЛЕЖАЩЕЙ ИНФОРМАТИЗАЦИИ

1.1 Цель работы

Углубление теоретических знаний и осуществление исследования и анализа предметной области, построение диаграммы «сущность-связь», модели данных, основанной на ключах и получение практических навыков работы с инструментом MySQL Workbench.

1.2 Общие сведения

MySQL – это бесплатная система управления базами данных (СУБД) с открытым кодом, разработанная в начале 90-х годов компанией MySQL AB исключительно для своих нужд. В последствие этот программный продукт вышел за рамки этой компании и приобрел огромную популярность за счет своей простоты и компактности, т.к. первые дистрибутивы этого сервера были примерно 4 мегабайта [9].

MySQL можно запустить на разных платформах, например: Windows, Linux, Mac OS X, FreeBSD, HP-UX, Solarisи других.

Широкую популярность MySQL приобрела в Интернете, в качестве сервера баз данных. Несмотря на то, что по сравнению с другими, платными СУБД, такими как Oracle или MS SQL Server MySQL немного проигрывает, в большинстве (а для web-мастеров практически во всех) случаях она полностью удовлетворяет все потребности. MySQL сегодня принадлежит Oracle и постоянно обновляется.

MySQL поддерживают практически все популярные языки программирования, например: Delphi, C, C++, Java, Perl, PHP, Python, Ruby и другие.

Необходимо понимать, что MySQL это не чисто база данных и что в ней хранится только данные и все, MySQL обладает практически всеми возможностями, которыми должна обладать настоящая СУБД это: возможность писать собственные процедуры, функции и многое другое

СУБД MySQL можно скачать на официальном сайте MySQL https://dev.mysql.com/downloads/installer. У MySQL есть собственный интерфейс для организации взаимодействия с клиентами, с помощью которого можно перемещать данные и изменять параметры баз данных. Чтобы иметь возможность работать с базой данных, необходимы учетная запись и пароль. Каждый сервер MySQL может содержать несколько баз данных, где группируются таблицы. Если MySQL установлен на локальном компьютере, то по умолчанию именем пользователя является root.

Первое место среди продуктов для разработки и администрирования баз данных MySQL принадлежит инструменту Workbench (разработка компании Sun Systems/Oracle), который может работать на платформах Microsoft Windows, Mac OS X и Linux. Workbench объединяет в себе разработку и администрирование баз данных [10].

MySQL Workbench распространяется под свободной лицензией – Community Edition и с ежегодной оплачиваемой подпиской – Standard Edition. Последняя включает в себя дополнительные возможности, которые способны существенно улучшить производительность, как разработчиков, так и администраторов баз данных. В данной лабораторной работе используется MySQL Workbench Community Edition.

В данной лабораторной работе будет подробно рассмотрен процесс установки СУБД MySQL версии 5.7.21 на операционную систему Windows 7, помимо этого будет установлено средство разработки и администрирования MySQL Workbench 6.3.6.

1.3 Методические указания по установке MySQL для Windows

1.3.1 Загрузка последней версии MySQL для Windows

Лучше всего скачивать MySQL Installer (установщик, дистрибутив) который будет включать не только сервер MySQL, но и многое другие сервисы. Для того чтобы скачать установщик, необходимо перейти на официальную страницу загрузки http://dev.MySQL.com/downloads/windows/installer/.

После перехода по представленной выше ссылке в нижней части страницы, появившейся в браузере, появится блок «MySQL Installer» (на момент написания методических указаний актуальной была версия 5.7.21). Для начала загрузки установочного файла необходимо активировать опцию «Download».

Затем появится предложение авторизоваться. Можно зарегистрироваться или воспользоваться существующим аккаунтом от Oracle или выбрать опцию «No thanks, just start my download» и активировать процесс загрузки дистрибутива.

В итоге загрузится пакет установщика Windows файл **MySQL-installer-community-5.7.21.0.msi** размером 381.4 мегабайт.

1.3.2 Установка MySQL 5.7.21 на Windows

Для активации установки MySQL необходимо запустить загруженный с официального сайта установочный файл. В процессе установки установщик будет проверять систему на наличие необходимых внешних компонентов, которые нужны для работы каждого из выбранных продуктов MySQL, и выдаст их список, и в случае необходимости можете их быстро установить.

Язык программы установки английский также, как и интерфейс программы MySQL Workbench.

Последовательность установки MySQL 5.7.21 на Windows представлена ниже.

1. Активировав опции «I accept the license terms» и «Next» необходимо согласиться с условиями лицензионного соглашения.

2. Затем необходимо выбрать компоненты, которые требуется установить, а поскольку необходимо установить все то, что нужно начинающему разработчику, то достаточно выбрать тип установки по умолчанию, т.е. «Developer Default» и активировать опцию «Next».

3. Далее программа установки снова будет проверять систему на наличие необходимых компонентов, недостающие компоненты необходимо установить (т.е. выделить их и активировать опцию «Execute») или активировать опцию «Next», но в этом случае соответствующие компоненты не будут работать. Но, например, если эти компоненты не нужны, допустим, нет необходимости в использовании Visual Studio, то и устанавливать компонент не нужно. А поскольку MySQL Workbench 6.3.6 нужен, поэтому его надо тоже установить (см. рис. 1.1).

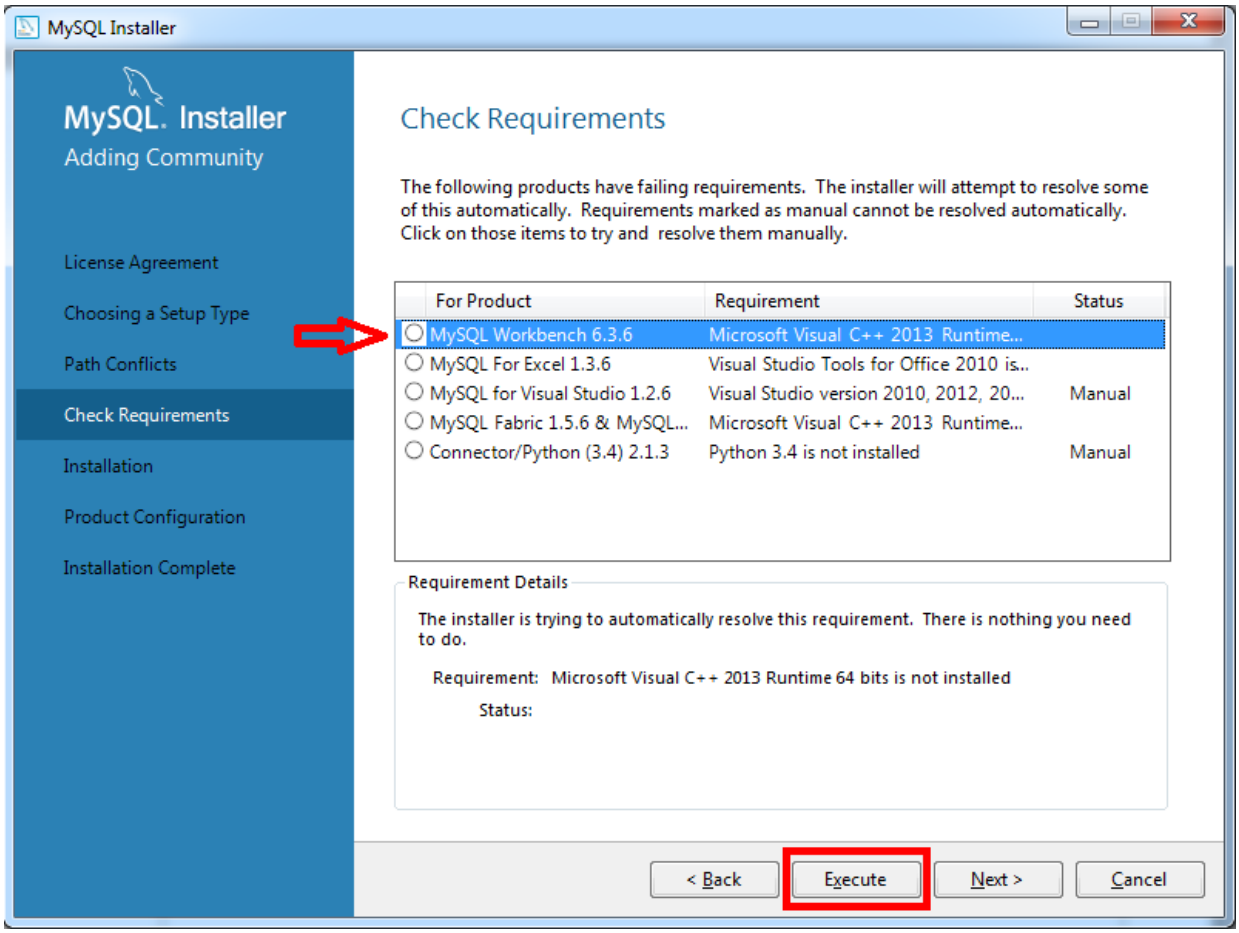


Рис. 1.1. Окно установки компонентов,
необходимых для работы MySQL Workbench

Появится небольшое предупреждение, связанное с отсутствием некоторых компонентов, необходимо активировать опцию «Yes»

4. Затем установщик выведет информацию о тех компонентах, которые будут установлены. Необходимо активировать опцию «Execute», после чего начнется установка всех перечисленных компонентов.

После установки всех компонентов появится опция «Next», которую необходимо активировать.

5. Далее необходимо настроить некоторые установленные компоненты. Для этого также необходимо активировать опцию «Next».

6. В окне, представленном на рис. 1.2 необходимо оставить все по умолчанию, т.е. ничего не менять, а сразу активировать опцию «Next».

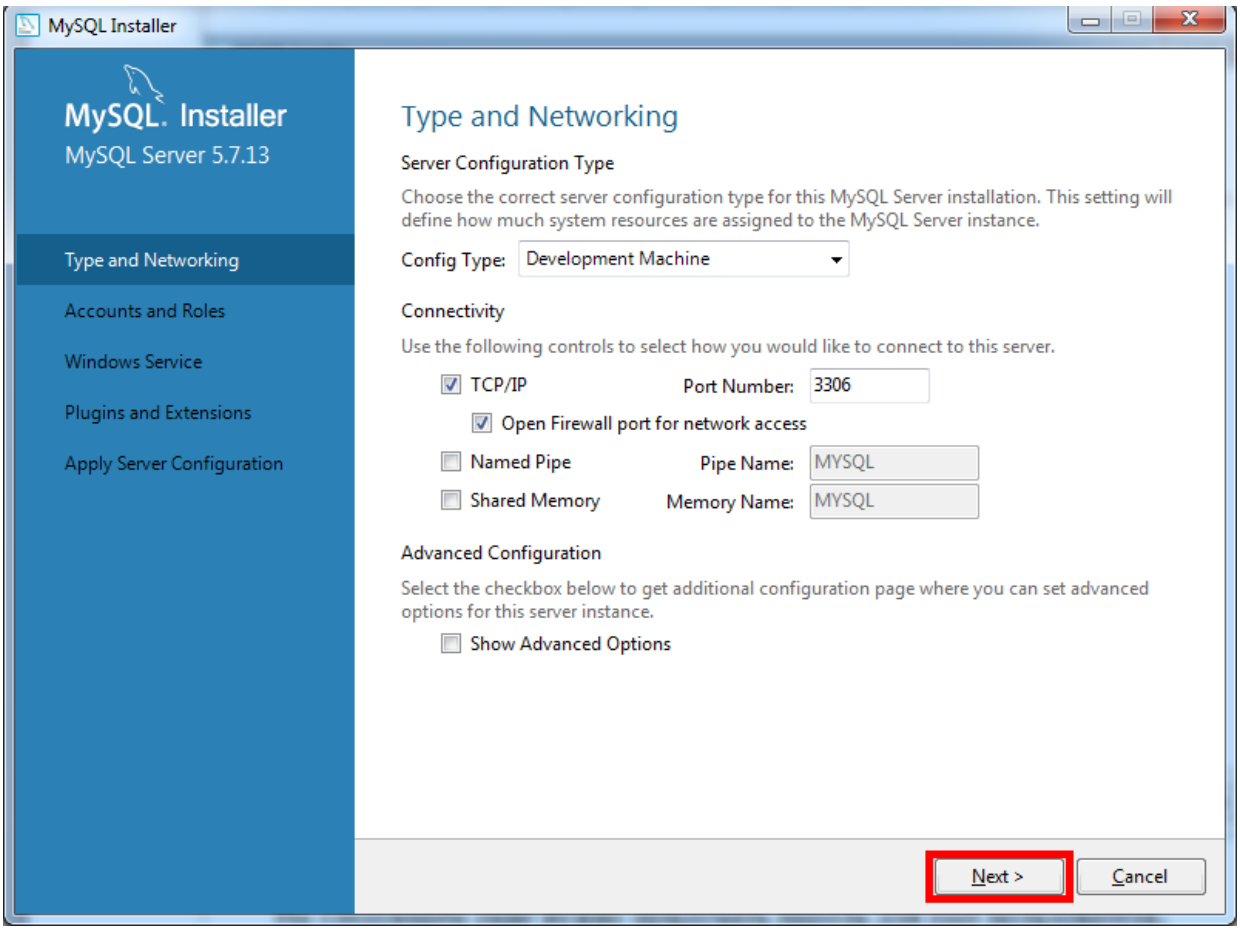


Рис. 1.2. Окно настроек сервера MySQL

7. На следующем окне необходимо будет указать пароль для root-пользователя, т.е. для главного администратора (очень важно его запомнить), также здесь можно сразу добавить и других пользователей, т.е. создать их через опцию «Add User», затем необходимо активировать опцию «Next».

8. В следующем окне необходимо оставить все настройки, указанные по умолчанию и активировать опцию «Next»

9. Далее необходимо применить все настройки, для этого требуется активировать опцию «Execute».

Теперь сервер MySQL сконфигурирован и можно активировать опцию «Finish».

10. Теперь необходимо настроить тестовые данные для MySQL сервера, для этого требуется активировать опцию «Next».

Для проверки подключения к серверу необходимо сначала выбрать опцию «Check», а затем, в случае удачного подключения (появившегося сообщения Connection successful), активировать опцию «Next»

Далее необходимо активировать опции «Execute», а затем опцию «Finish».

11. Установка практически завершена, необходимо активировать опцию «Next», а затем опцию «Finish». Если активировать опцию «Start MySQL Workbench after Setup», то сразу запустится программа MySQL Workbench

1.4 Методические рекомендации по установке MySQL для Ubuntu

Установка и настройка программного обеспечения в Ubuntu производится в консоли. Обязательным условием является наличие Интернет-соединения. Вызов консоли Ubuntu осуществляется комбинацией клавиш Ctrl+Alt+T.

Для установки MySQL-сервера и клиента, в консоли Ubuntu необходимо выполнить команду:

```
sudo apt-get install mysql-server mysql-client
```

В процессе установки, в консоли потребуется дважды ввести пароль администратора сервера MySQL.

Для установки графического клиента Workbench для работы с MySQL в строке поиска менеджера приложений Ubuntu необходимо ввести команду «MySQL Workbench» и активировать опцию «Установить» (рис. 1.3).

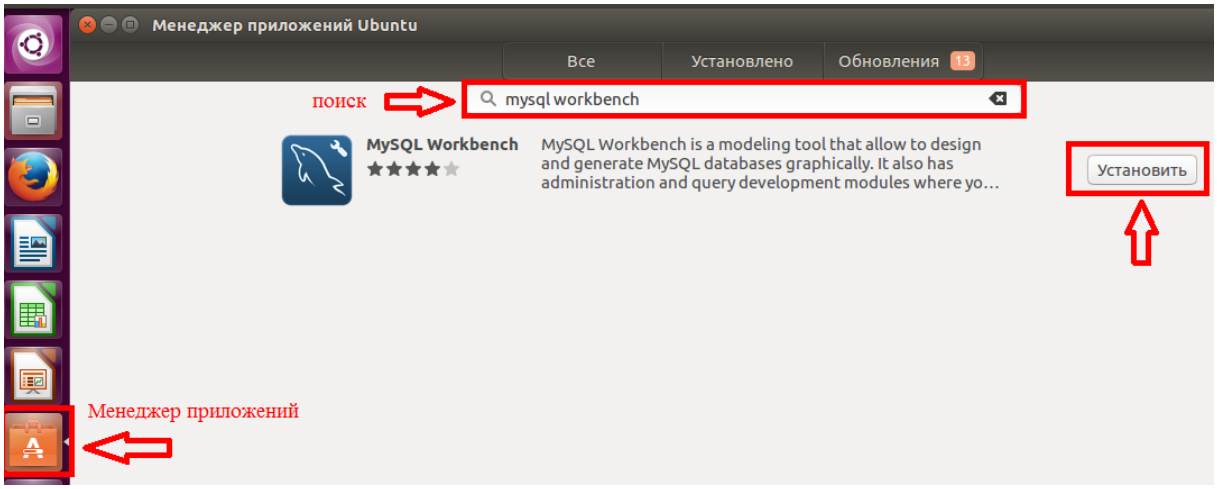


Рис. 1.3. Установка MySQL Workbench
с помощью менеджера приложений Ubuntu

1.5 Методические рекомендации по разработке информационной модели базы данных

1.5.1 Особенности практического проектирования баз данных

Разработанная функциональная модель системы отвечает на вопросы «Что должна делать система?» и «За счет каких действий может быть А тигнут требуемый результат?». Эта модель также позволяет концептуально определить наборы данных, используемых в системе.

В то же время она не отвечает на вопрос: «Каким образом организованы данные в системе?». Для ответа на него необходимо построить информационную модель (спроектировать БД).

Традиционно процедуру проектирования базы данных разбивают на три этапа, каждый из которых завершается созданием соответствующей информационной модели:

1. Концептуальное проектирование – создание представления (схемы, модели) БД, включающего определение важнейших сущностей (таблиц) и

связей между ними, но не зависящего от модели БД (иерархической, сетевой, реляционной и т.д.) и физической реализации (целевой СУБД).

2. Логическое проектирование – развитие концептуального представления БД с учетом принимаемой модели (иерархической, сетевой, реляционной и т.д.).

3. Физическое проектирование – развитие логической модели БД с учетом выбранной целевой СУБД**.**

Концептуальное и логическое проектирование вместе называют также *инфологическим* или *семантическим проектированием*.

В настоящее время для проектирования БД активно используются CASE-средства, в основном ориентированные на использование ERD (Entity – Relationship Diagrams, диаграммы «сущность-связь»), являющейся информационной моделью. С их помощью определяются важные для предметной области объекты (сущности), отношения друг с другом (связи) и их свойства (атрибуты). Средства проектирования ERD в основном ориентированы на реляционные базы данных (РБД), и если существует необходимость проектирования другой системы, например, объектно-ориентированной, то лучше избрать другие методы проектирования.

ERD были впервые предложены П. Ченом в 1976 г. Основными элементами ERD являются:

– сущность (таблица, в РБД – отношение) – реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению. Если выражаться точнее, то это не объект, а набор объектов (класс) с одинаковыми свойствами. Примеры сущностей: работник, деталь, ведомость, результаты сдачи экзамена и т.д.;

– экземпляр сущности (запись, строка, в РБД – кортеж) – уникально идентифицируемый объект;

– связь – некоторая ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Примерами связей могут являться родственные отношения «отец–сын», производственные – «начальник-подчиненный» или произвольные – «иметь в собственности», «обладать свойством»;

– атрибут (столбец, поле) – свойство сущности или связи.

Этапы и правила построения диаграммы «сущность-связь» в нотации П.Чена рассмотрены в п. 3.2.1 методических указаний по выполнению курсовой работы по дисциплине «Технологии баз данных».

Пример построения модели, основанной на ключах, рассмотрены в п. 3.2.1 методических указаний по выполнению курсовой работы по дисциплине «Технологии баз данных».

При использовании любого CASE-средства вначале строится логическая модель БД в виде диаграммы с указанием сущностей и связей между ними, а тем физическая модель.

Логической моделью данных называется универсальное представление структуры данных, независимое от конечной реализации базы данных и аппа-

ратной платформы. Логическая модель позволяет понять суть проектируемой системы, отражая логические взаимосвязи между сущностями. На основании полученной логической модели переходят к физической модели данных.

Физическая модель представляет собой диаграмму, содержащую всю необходимую информацию для генерации БД для конкретной СУБД или даже конкретной версии СУБД, и отражает физические свойства проектируемой БД (типы данных, размер полей, индексы). Параметры физической информационной модели зависят от выбранной СУБД. Если в логической модели не имеет значения, какие идентификаторы носят таблицы и атрибуты, тип данных атрибутов и т.д., то в физической модели должно быть полное описание БД в соответствии с принятым в ней синтаксисом, с указанием типов атрибутов, триггеров, хранимых процедур и т.д. По одной и той же логической модели можно создать несколько физических.

Такой порядок действий называется *прямое проектирование БД (Forward Engineering DB).* CASE-средства позволяют выполнять также *обратное проектирование БД (Reverse Engineering DB)*, т.е. на основании системного каталога БД или DDL-скрипта (Data Definition Language – язык описания данных) построить физическую и, далее, логическую модель данных.

1.5.2 Построение внешней модели БД

Предметная область (ПрО) информационной системы рассматривается как совокупность реальных процессов и объектов (сущностей), представляющих интерес для её пользователей.

Каждый из объектов обладает определённым набором свойств (атрибутов), среди которых можно выделить существенные и малозначительные. Признание какого-либо свойства существенным носит относительный характер. Для упрощения процедуры формализации ПрО в большинстве случаев прибегают к разбиению всего множества объектов ПрО на группы объектов, однородных по структуре и поведению (относительно рамок рассматриваемой ПрО), называемых *типами объектов*. Данные ПрО представлены *экземплярами объектов*. Экземпляры объектов одного типа обладают одинаковыми наборами атрибутов, но должны отличаться значением хотя бы одного атрибута для того, чтобы быть узнаваемыми [3].

Для каждого объекта определяется идентификатор – ключевой атрибут или комбинация атрибутов. Такой идентификатор называется **первичным ключом,** его значение является уникальным и обязательным.

Между объектами ПрО могут существовать связи, имеющие различный содержательный смысл (семантику). Эти связи могут быть **факультативными** или **обязательными** (рис. 1.4). Если вновь порождённый объект одного из типов оказывается по необходимости связанным с объектом другого типа, то между этими типами объектов существует обязательная связь. Иначе связь является факультативной.

Различают типы множественных связей: «один к одному» (1:1), «один ко многим» (1:n) и «многие ко многим» (m:n) (рис. 1.5).

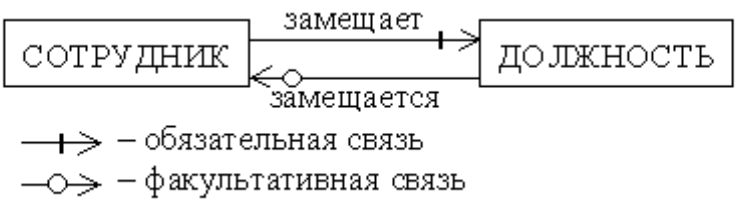


Рис. 1.4. Примеры обязательной и факультативной связей

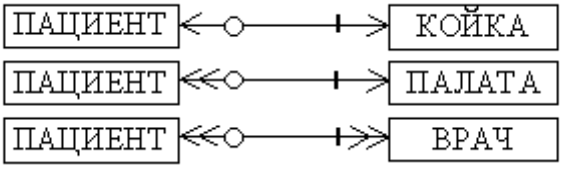


Рис. 1.5. Примеры типов множественных связей

Совокупность типов сущностей и типов связей между ними характеризует структуру предметной области. Собственно, данные представлены экземплярами объектов и связей между ними.

Множества типов объектов ПрО и экземпляров объектов, значения атрибутов объектов и связи между ними могут изменяться во времени. Поэтому каждому моменту времени можно сопоставить некоторое состояние ПрО. Состояния ПрО обладают совокупностью свойств (правил), которые характеризуют семантику ПрО. Эти правила могут быть заданы с помощью так называемых **ограничений целостности**, которые накладываются на типы объектов, типы связей и/или их экземпляры [4].

1.5.3 Логическое проектирование базы данных

Цель логического проектирования – развить концептуальное представление БД с учетом принимаемой модели БД (иерархической, сетевой, реляционной и т.д.) [5].

В качестве модели будет принята реляционная БД в третьей нормальной форме (набор нормализованных отношений с кратностью связей 1:N). Требуется проверить концептуальную модель с помощью методов нормализации. Для этого необходимо:

– удалить связи N:M;

– удалить связи с атрибутами (связи с атрибутами должны быть преобразованы в сущности);

– удалить сложных связей (со степенью участия более 2);

– удалить рекурсивные связи (со степенью участия 1);

– удалить атрибуты, имеющие несколько значений (устраняется путем введения новой сущности и связи 1:N – рис. 1.6);

– удалить избыточные связи (связь является избыточной, если одна и та же информация может быть получена не только через нее, но и с помощью другой связи – рис. 1.7, где одну из связей «Руководит» можно удалить);

– перепроверить связи (в процессе определения сущностей могли быть созданы сущности, которые на самом деле являются одной. В этом случае их следует объединить).

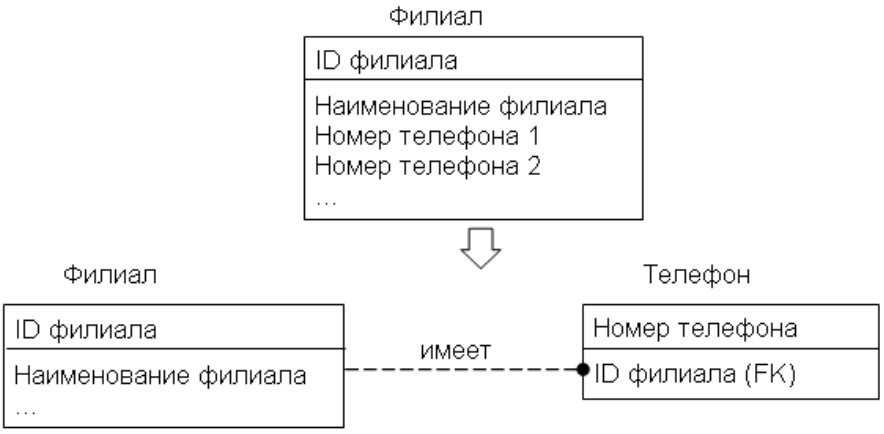


Рис. 1.6. Удаление многозначных атрибутов

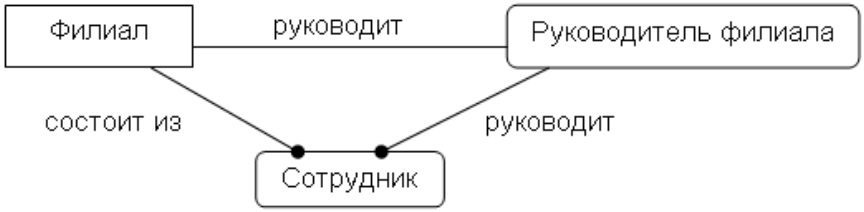


Рис. 1.7. Избыточные связи

1.5.4 Проверка модели с помощью правил нормализации

Основная идея нормализации заключается в том, чтобы каждый факт хранился в одном месте, т.е. чтобы не было дублирования данных. Многие из требований нормализации, как правило, уже учитываются при выполнении предыдущих шагов проектирования.

Проектирование реляционной БД представляет собой пошаговый процесс создания набора отношений (таблиц, сущностей), в которых отсутствуют нежелательные функциональные зависимости.

Функциональная зависимость определяется следующим образом. Пусть A и B – произвольные наборы атрибутов отношения. Тогда B функционально зависит от A (A → B), в том и только в том случае, если каждому значению A соответствует в точности одно значение B. Левая часть функциональной зависимости (A) называется *детерминантом*, а правая (B) – *зависимой частью*. В частности, в отношении A может быть первичным ключом, а B – набором неключевых атрибутов, так как одному значению первичного ключа в точности соответствует одно значение набора неключевых атрибутов.

Если в БД отсутствуют нежелательные функциональные зависимости, то это обеспечивает минимальную избыточность данных, что в свою очередь ведет к уменьшению объема памяти, необходимой для хранения данных. Процесс устранения таких зависимостей получил название *нормализация*. Она выполняется в виде последовательности тестов для некоторого отношения (таблицы, сущности) с целью проверки его соответствия (или несоответствия) набору ограничений для заданной нормальной формы.

Процесс нормализации впервые был предложен Э.Ф. Коддом в 1972 г. Сначала было предложено три вида нормальных форм (1NF, 2NF и 3NF). Затем Р. Бойсом и Э.Ф. Коддом (1974 г.) было сформулировано более строгое

определение третьей нормальной формы, которое получило название нормальная форма Бойса – Кодда (BCNF). Вслед за BCNF появились определения четвертой (4NF) и пятой (5NF или PJNF) нормальных форм (Р. Фагин, 1977 и 1979 г.). На практике нормальные формы более высоких порядков используются крайне редко. При проектировании БД, как правило, ограничиваются третьей нормальной формой, что позволяет предотвратить возможное возникновение избыточности данных и аномалии обновлений [6].

1NF. Отношение находится в 1NF, если на пересечении каждого столбца и строки находятся только элементарные (атомарные, неделимые) значения атрибутов. Степень неделимости (атомарности), т.е. решение о том, следует разбивать неатомарный атрибут на атомарные или оставить его псевдоатомарным, определяется проектировщиком БД исходя из конкретных условий. Если при обработке таблиц нет необходимости различать атомарные составляющие псевдоатомарного атрибута, то его можно не делить (например, атрибуты «Фамилия, имя, отчество», «Адрес» и т. д.).

2NF. Отношение находится во 2NF, если оно находится в 1NF, и каждый неключевой атрибут характеризуется полной функциональной зависимостью от первичного ключа.

Полная функциональная зависимость определяется следующим образом. В некотором отношении атрибут В полностью зависит от атрибута А, если атрибут В функционально зависит от полного значения атрибута А и не зависит от какого-либо подмножества полного значения атрибута А.

Например, таблица «Оценки по экзаменам» характеризуется следующим набором атрибутов {Номер зачетной книжки, Дисциплина, Дата сдачи, ФИО студента, № группы, Оценка}. Очевидно, что первичным ключом А клется набор {Номер зачетной книжки, Дисциплина, Дата сдачи}. Полной функциональной зависимостью обладает только один неключевой атрибут «Оценка». Атрибуты «ФИО студента» и «№ группы» могут быть однозначно определены по части первичного ключа – «Номер зачетной книжки». Таким образом, требуется разбиение исходной таблицы на две (рис. 1.8).

3NF. Отношение находится в 3NF, если оно находится во 2NF и никакой неключевой атрибут функционально не зависит от другого неключевого атрибута, т.е. нет транзитивных зависимостей.

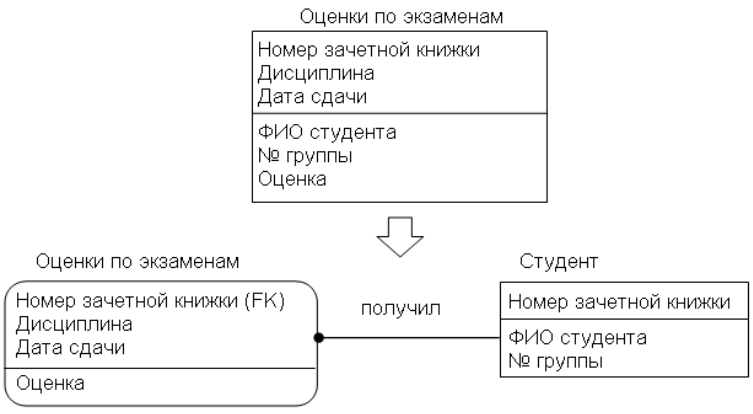


Рис. 1.8. Обеспечение полной функциональной зависимости

Транзитивная зависимость. Если для атрибутов A, B и C некоторого отношения существуют зависимости вида A → B и B → C, то атрибут C транзитивно зависит от атрибута A через атрибут B.

Например, таблица «Работник» характеризуется набором атрибутов {Табельный номер, Фамилия, Имя, Отчество, Должность, Зарплата, …}, первичный ключ – {Табельный номер}. В этой таблице от первичного ключа («Табельный номер») зависит неключевой атрибут «Должность», а от «Должности» другой неключевой атрибут «Зарплата». Для приведения к 3NF необходимо добавить новую таблицу (рис. 1.9).

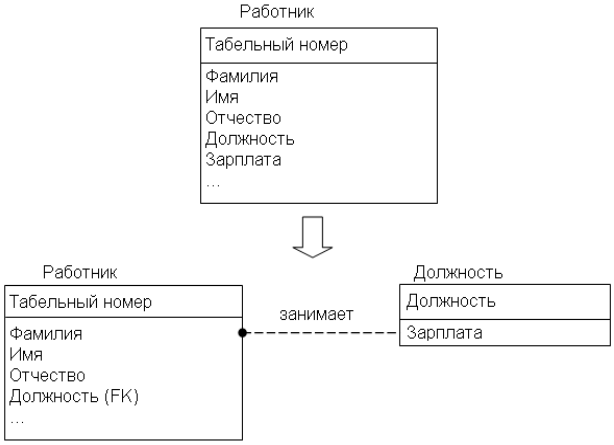


Рис. 1.9. Устранение транзитивной зависимости

1.5.5 Проверка выполнения транзакций

Наиболее частыми операциями по работе с БД являются ввод, удаление и модификация записей, а также выборка данных. На основе текущей ERD необходимо проверить выполнимость и, далее, корректность выполнения всех требуемых операций по работе с БД.

Примеры транзакций:

- изменение наименования раздельного пункта;
- удаление задания на расчет;
- выборка данных по раздельным пунктам расчетного пути участка и т.д.

1.5.6 Определение требований поддержки целостности данных

Ограничения целостности данных представляют собой ограничения, которые вводятся с целью предотвращения помещения в базу А клюю ечиивых данных.

К этим ограничениям относятся:

1) обязательные данные – атрибуты, которые всегда должны содержать одно из допустимых значений (NOT NULL). Например, поворот кривой (влево или вправо) должен быть обязательно задан. Обязательными также А клются все атрибуты, входящие в первичный ключ сущности;

2) домены – наборы допустимых значений для атрибута. Например, радиус кривой должен быть положительным числом не более 4 цифр или поворот кривой может принимать одно из двух допустимых значений – «Л» (влево) или «П» (вправо);

3) бизнес-правила (бизнес-ограничения) – ограничения, принятые в рассматриваемой предметной области. Например, сумма длин переходных кривых не должна быть более длины всей кривой, километраж начала или конца кривой должен быть в пределах общего километража пути и т.д.;

4) ссылочная целостность – набор ограничений, определяющих действия при вставке, обновлении и удалении записей (экземпляров сущности). Например:

– при наличии обязательной связи вставка записи в дочернюю сущность требует обязательного заполнения атрибутов внешнего ключа, и введенному значению должна соответствовать запись родительской сущности;

– аналогичное требование выдвигается при обновлении внешнего ключа в дочерней сущности;

– удаление записи из дочерней сущности или вставка записи в родительскую не вызывают нарушения ссылочной целостности;

– удаление записи в родительской сущности может требовать удаления всех связанных записей в дочерней сущности.

Автоматическая поддержка всех видов ограничений целостности возможна за счет использования операторов SQL.

1.6 Описание лабораторной установки

В качестве лабораторной установки используется персональный компьютер, на котором установлена **СУБД MySQL Workbench**.

Запустить MySQL Workbench можно активировав меню Пуск → MySQL → MySQL Workbench 6.3 CE.

В верхней части экрана находится список подключений к MySQL серверам разрабатываемых проектов (рис. 1.10), а список последних открытых моделей данных – в нижней части экрана.

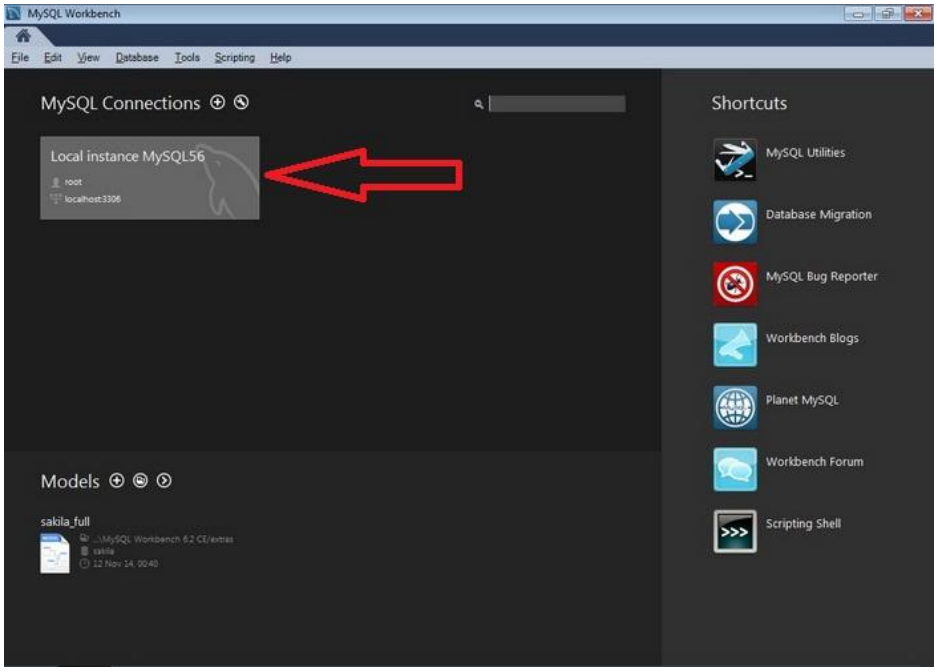


Рис. 1.10. Выбор сервера MySQL для подключения

Работа начинается с создания схемы данных или загрузки существующей структуры в MySQL Workbench.

После запуска необходимо выбрать экземпляр сервера MySQL для того чтобы к нему подключится, по умолчанию он всего один (локальный), необходимо просто его активировать и ввести пароль root-пользователя (который был указан при настройке сервера).

Подключение к серверу MySQL с помощью MySQL Workbench закончено и доступны все тестовые базы, к которым можно писать SQL-запросы.

1.6.1 Создание и редактирование модели данных

В начале работы необходимо создать схему, в которой будет храниться физическая модель. Для добавления модели необходимо активировать опцию ⊕ рядом с заголовком «Models» или воспользоваться пунктом меню «File → New Model» (Ctrl + N).

При двойном щелчке на имени схемы (по умолчанию схеме присвоено имя **mydb**) появится закладка со свойствами схемы, на которой можно изменить её имя по умолчанию, кодировку, добавить комментарии (см. рис. 1.11). Необходимо изменить имя схемы с **mydb** на **book**.

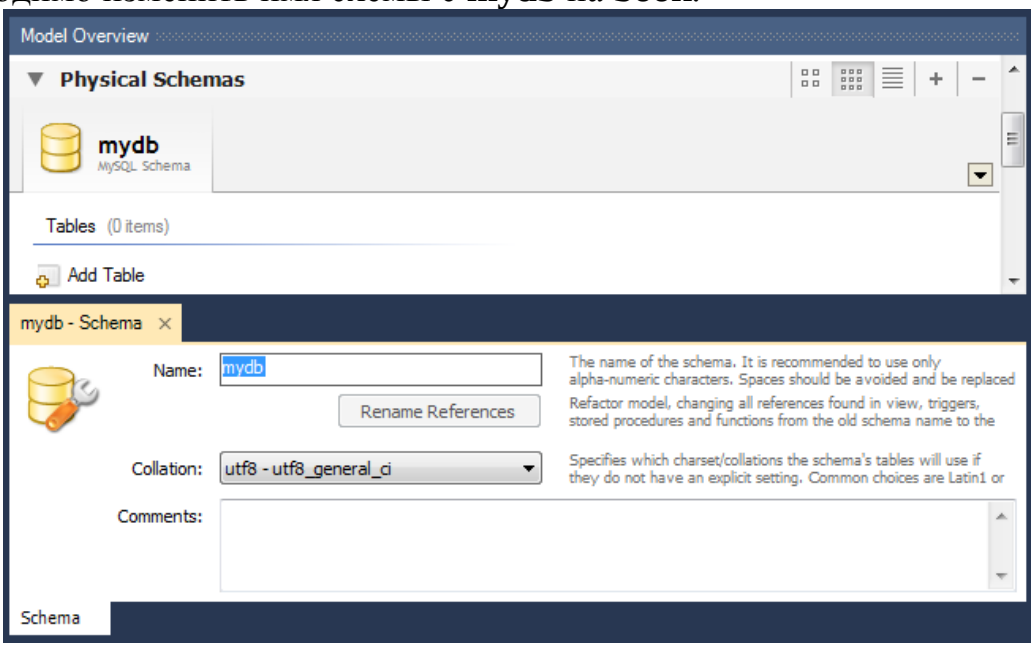


Рис. 1.11. Создание модели данных

На рис. 1.11 необходимо ввести имя базы данных, выбрать кодировку по умолчанию и, по необходимости, заполнить поле комментария. Теперь можно приступать к созданию таблиц.

Добавление и редактирование таблицы. Список баз данных проекта и список таблиц в пределах базы данных располагается во вкладке «Physical Schemas». Чтобы создать таблицу, необходимо двойным кликом активировать опцию «+Add Table» (рис. 1.12).

Откроется удобный интерфейс для редактирования списка полей и их свойств. Здесь можно задать название поля, тип данных (см. табл. 1.1), а также установить для полей различные атрибуты.

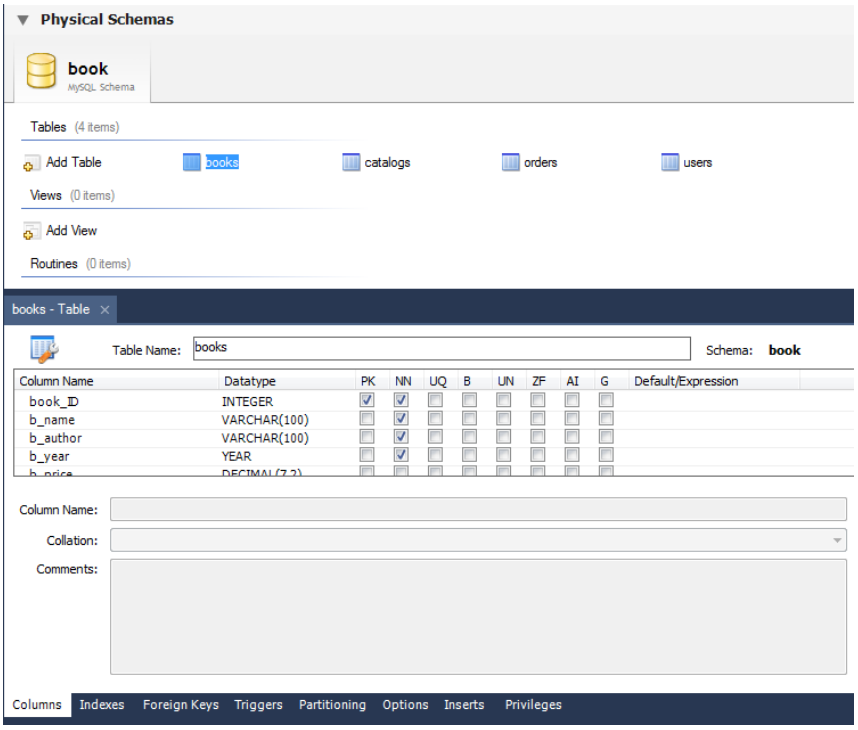


Рис. 1.12. Добавление таблицы

Таблица 1.1 – Типы данных MySQL

Тип данных	Объем памяти	Диапазон	Пример
Числовые типы данных			
TINYINT(M)	1 байт	от -128 до 127 или от 0 до 255	MEDIUMINT UNSIGNED – хранит любое число в диапазоне от 0 до 16777215 SMALLINT (4) ZEROFILL – свободные позиции слева заполнит нулями. Например, величина 2 будет отображаться, как 0002. TINYINT (2) – предполагается, что значения будут двузначными, но по факту будет хранить и трехзначные
SMALLINT(M)	2 байта	от -32768 до 32767 или от 0 до 65535	
MEDIUMINT(M)	3 байта	от -8388608 до 8388608 или от 0 до 16777215	
INT(M) INTEGER(M) INTEGER	4 байта	от -2147683648 до 2147683648 или от 0 до 4294967295	
BIGINT(M)	8 байт	от -2^{63} до $2^{63}-1$ или от 0 до 2^{64}	
BOOL B0OOLEAN BIT	1 байт	либо 0, либо 1	
FLOAT(M,D)	4 байта	от +(-) $1.175494351 \times 10^{-39}$ до +(-) $3.402823466 \times 10^{38}$	FLOAT (5,2) – будет хранить числа из 5 символов, 2 из которых будут идти после запятой (например: 46,58)
DOUBLE(M,D) DOUBLE REAL	8 байт	от +(-) $2.2250738585072015\times\times10^{-308}$ до +(-) $1.797693134862315\times\times10^{308}$	
DECIMAL(M,D) DECIMAL DEC(M,D) NUMERIC(M,D) FIXED	M + 2 байт	зависят от параметров M и D	DECIMAL (5,2) – будет хранить числа от -99,99 до 99,99

Продолжение табл. 1.1

Тип данных	Объем памяти	Диапазон	Пример
Строковые типы данных			
CHAR(M) BINARY(M)	M символов	Значение M от 0 до 65535	CHAR (8) – хранит строки из 8 символов и занимает 8 байтов.
VARCHAR(L) VARBINARY(L)	L+1 символов	Значение M от 0 до 65535	VARCHAR (3) – хранит строки максимум из 3 символов, но пустая строка '' занимает 1 байт памяти, строка 'a' – 2 байта, строк 'aa' – 3 байта, строка 'aaa' – 4 байта. Значение более 3 символов будет усечено до 3
TINYBLOB TINYTEXT	L+1 символов	2^8-1 символов	
BLOB(L) LONGVARBINARY TEXT(L) LONGVARCHAR	L+2 символов	2^{16}-1 символов	
MEDIUMBLOB MEDIUMTEXT	L+3 символов	2^{24}-1 символов	
LONGBLOB LONGTEXT	L+4 символов	2^{32}-1 символов	
ENUM()	1 или 2 байта	65535 элементов	ENUM ('да', 'нет') – в столбце с таким типом может храниться только одно из имеющихся значений. Удобно использовать, если в столбце должен храниться ответ на вопрос.
SET()	до 8 символов	64 элемента	SET ('первый', 'второй') – в столбце с таким типом может храниться одно из перечисленных значений, оба сразу или значение может отсутствовать вовсе.
Календарные типы данных			
DATE	3 байта	от '1000-01-01' до '9999-12-31'	
DATATIME	8 байт	от '1000-01-01 00:00:00' до '9999-12-31 23:59:59'	
TIMESTAMP	4 байта	от '1970-01-01 00:00:00' до '2037-12-31 23:59:59'	
YEAR YEAR(M)	1 байт	от 1970 до 2069 для M=2 от 1901 до 2155 для M=4	YEAR (2) – 70, YEAR (4) – 1970. Если параметр M не указан, то по умолчанию считается, что он равен 4

Возможные атрибуты для полей:

– PK – Primary key (первичный ключ);

– NN – Not null (не null, т.е. значение должно быть определено);
– UQ – Unique (уникальное значение);
– B – Binary (двоичное значение);
– UN – Unsigned (беззнаковое значение);
– ZF – Zero fill (заполнено нулями);
– AI – Autoincrement (автоинкремент – автоматическое увеличение на 1).

DEFAULT – значение по умолчанию, т.е., значение, которое при добавлении новой строки в таблицу автоматически вставляется в поле сервером, если пользователь оставил значение поля пустым.

Управление индексами. Каждая сущность должна обладать атрибутом или комбинацией атрибутов, чьи значения однозначно определяют каждый экземпляр сущности. Эти атрибуты образуют первичный ключ сущности (Primary Key, PK).

Созданные таблицы на диаграмме представлены на рис. 1.13.

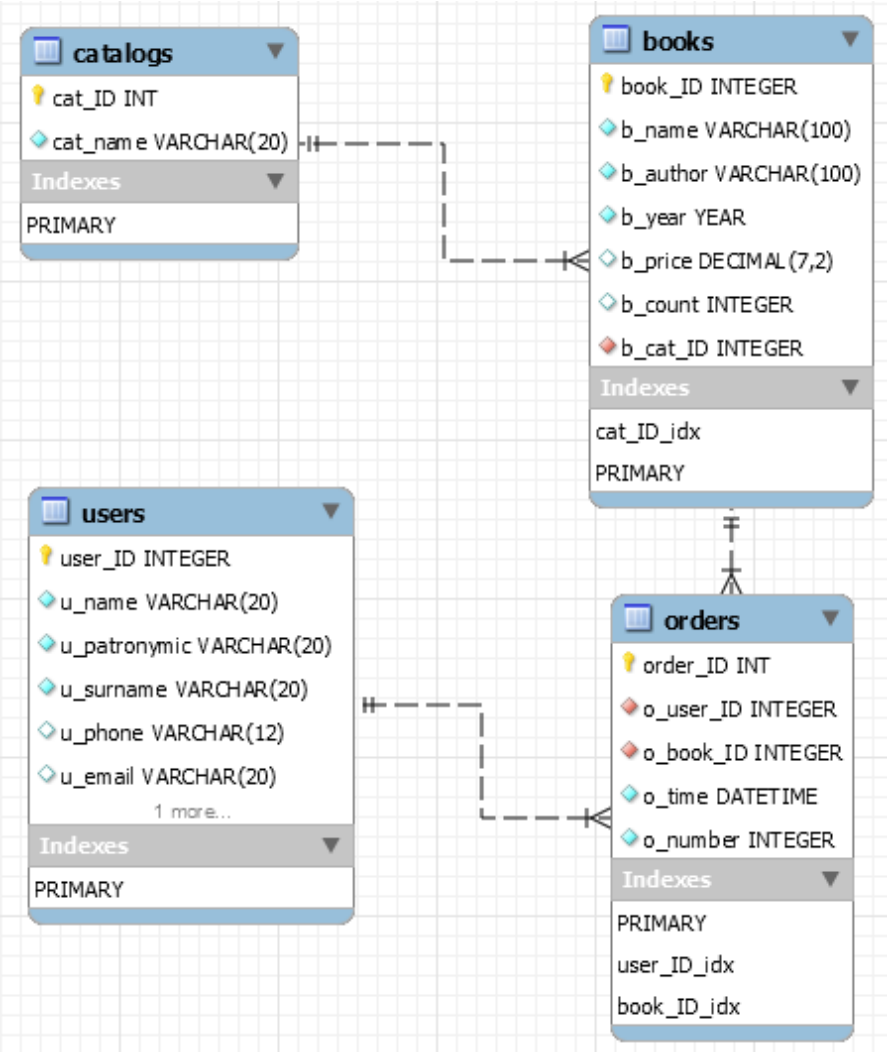


Рис. 1.13. Модель базы данных в MySQL Workbench

При создании первичного ключа автоматически создается *индекс* по этому первичному ключу. Индекс представляет собой вспомогательную

структуру, которая обеспечивает, прежде всего, повышение скорости доступа к данным по значению индекса.

Связи между таблицами. Если между двумя сущностями имеется специфическое отношение связи или категоризации, то атрибуты, входящие в первичный ключ родительской или общей сущности, наследуются в качестве атрибутов сущностью-потомком или категориальной сущностью соответственно. Эти атрибуты и называются внешними ключами (Foreign Key, FK).

Установка внешних ключей и связывание таблиц возможно только для таблиц InnoDB (эта система хранения данных выбирается по умолчанию). Для управления связями в каждой таблице находится вкладка «Foreign Keys» (рис. 1.14).

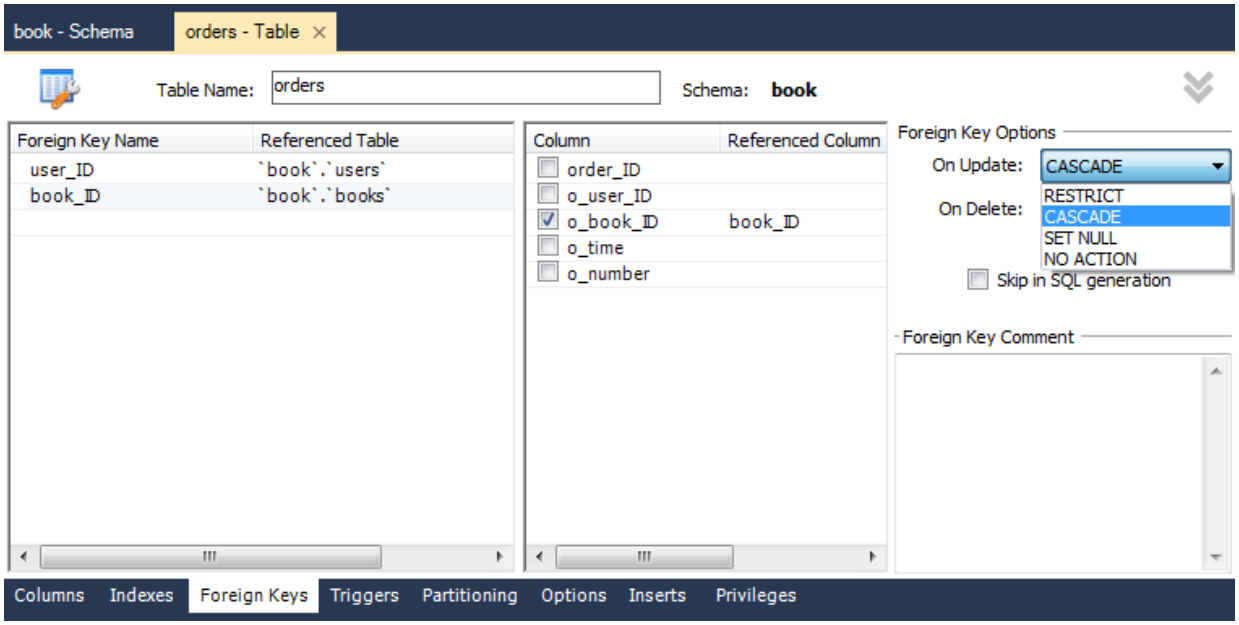


Рис. 1.14. Управление связями в таблице на вкладке Foreign Keys

Для добавления связи необходимо открыть вкладку «Foreign Keys» дочерней таблицы, ввести имя внешнего ключа и выбрать таблицу-родителя. Далее в средней части вкладки в графе «Column» выбир необходимо выбрать поле-ключ из дочерней таблицы, а в графе «Referenced Column» – соответствующее поле из родительской таблицы (тип полей должен совпадать). При создании внешних ключей в дочерней таблице автоматически создаются соответствующие индексы.

В разделе «Foreign Key Options» настраивается поведение внешнего ключа при изменении соответствующего поля (ON UPDATE) и удалении (ON DELETE) родительской записи:

– RESTRICT – выдавать ошибку при изменении/удалении родительской записи;

– CASCADE – обновлять внешний ключ при изменении родительской записи, удалять дочернюю запись при удалении родителя;

– SET NULL – устанавливать значение внешнего ключа NULL при изменении / удалении родителя (неприемлемо для полей, у которых установлен флаг NOT NULL!);

– NO ACTION – не делать ничего, однако по факту эффект аналогичен RESTRICT.

В приведённом примере были добавлены к дочерней таблице *orders* внешние ключи для связи с родительскими таблицами *users* и *books*. При редактировании поля *user_ID* и удалении позиций из таблицы *users* или редактировании поля *book_ID* и удалении позиций из таблицы books аналогичные изменения будут автоматически происходить и со связанными записями из таблиц *users* или *books* соответственно.

После того, как все таблицы связаны, необходимо сохранить схему. Для этого необходимо выбрать пункт меню File → Save Model As …

Наполнение таблицы базовыми данными. При создании проекта в базу данных часто нужно добавлять стартовые данные. Это могут быть корневые категории, пользователи-администраторы и т.д. В управлении таблицами MySQL Workbench для этого существует вкладка «Inserts» (рис. 1.15).

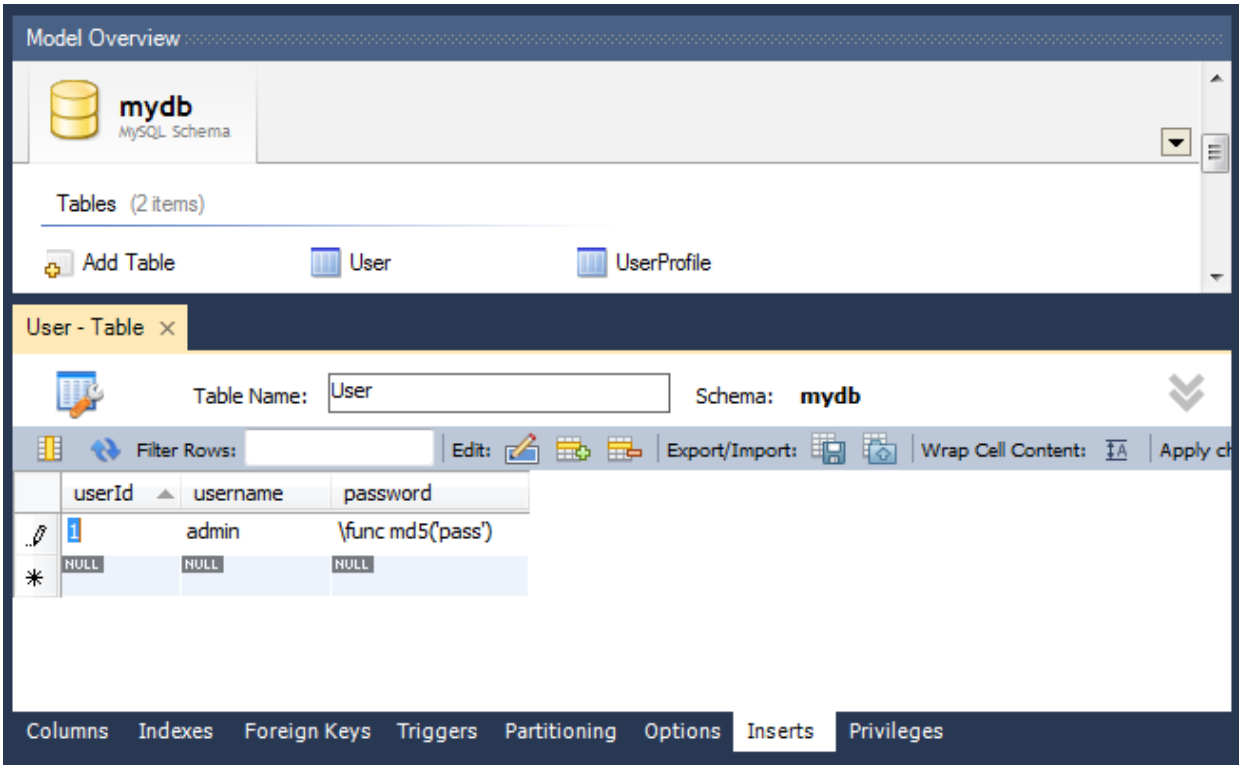


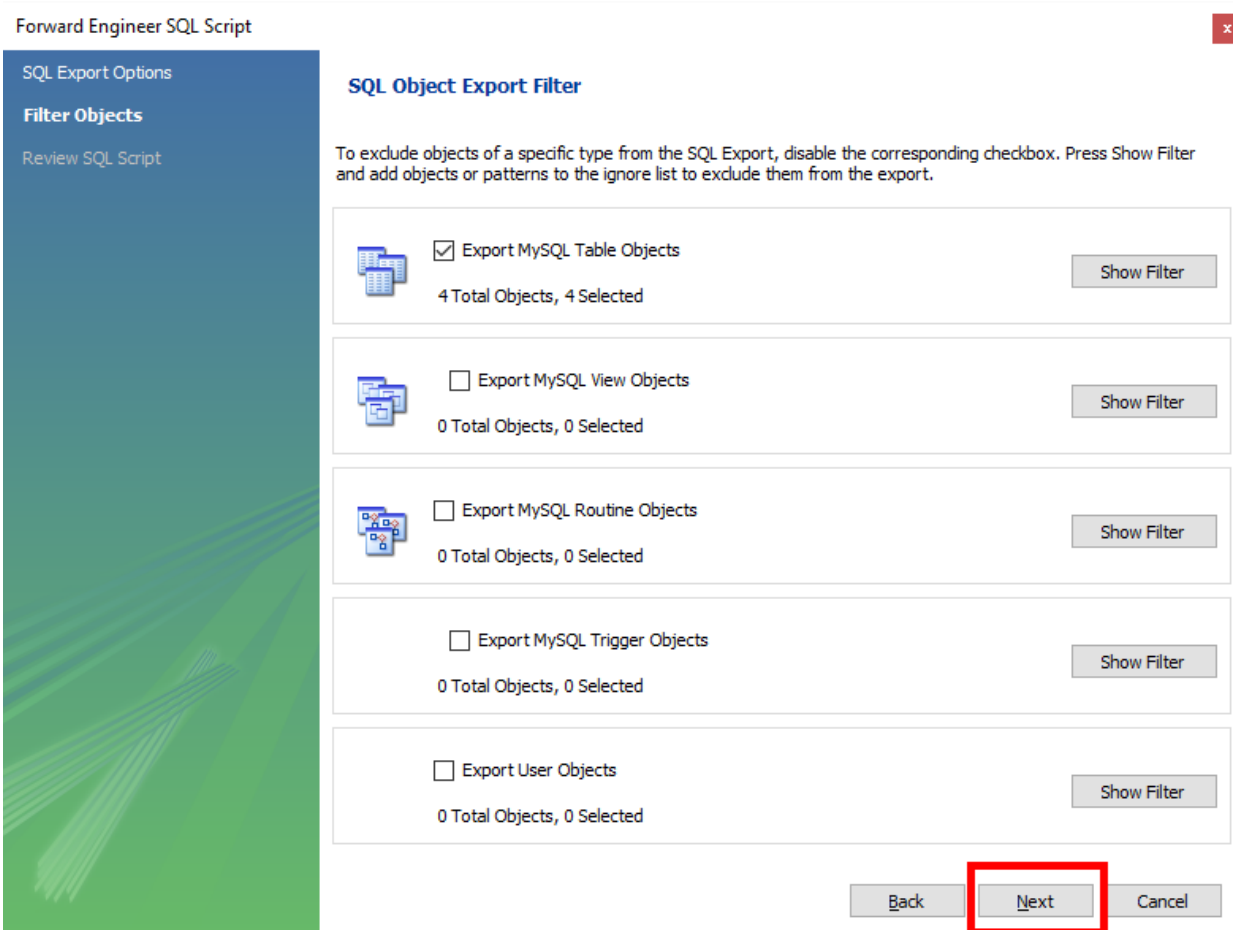
Рис. 1.15. Добавление стартовых данных в таблицу на вкладке Inserts

В случае если перед записью в базу данных к данным нужно применить какую-то функцию MySQL, это делается с помощью синтаксиса \func functionName('data'), например, \func md5('password').

После ввода данных необходимо сохранить их в локальную базу данных, активировав опцию «Apply Changes».

Создание SQL-скрипта. Последним этапом моделирования данных, является трансформация разработанной модели в базу данных MySQL. Для этого необходимо создать SQL-скрипт.

Для этого необходимо воспользоваться пунктом меню File → Export → Forward Engineer SQL CREATE Script (рис. 1.16).



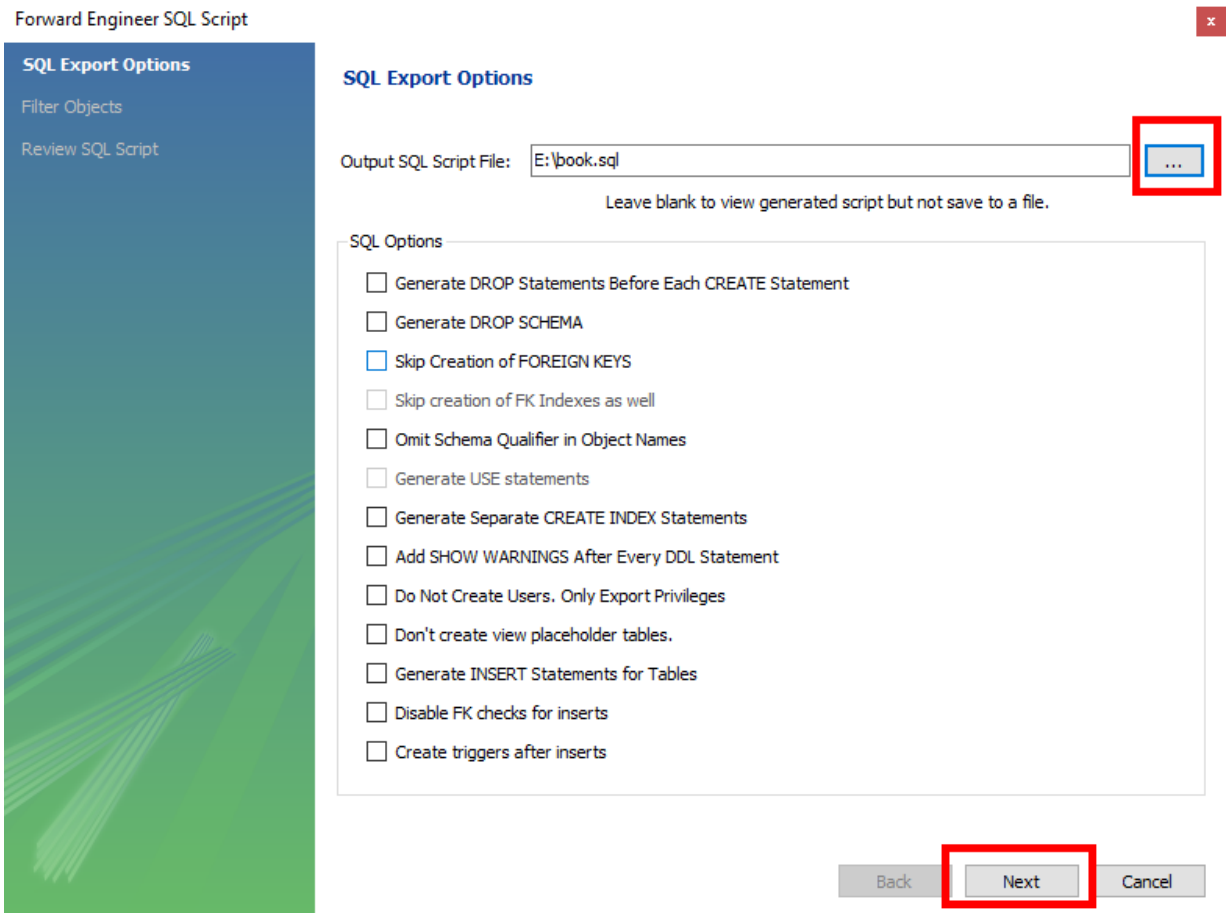


Рис. 1.16. Создание SQL-скрипта

В появившемся окне в поле Output file при помощи опции «Browse» необходимо выбрать путь хранения скрипта и задать его имя. Затем активировать опцию «Next» и в следующем окне выбрать опцию «Finish» (рис. 1.17).

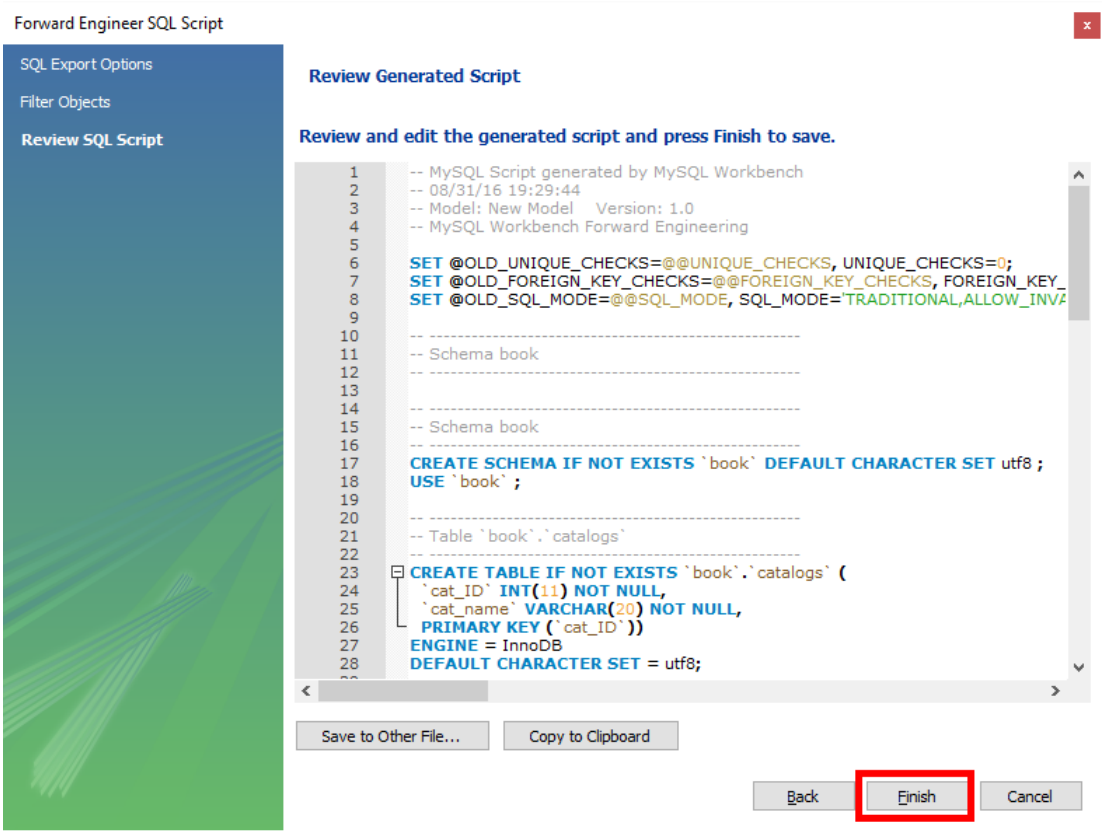


Рис. 1.17. Содержимое SQL-скрипта

В результате после данной операции будет создан SQL-файл.

1.6.2 Импорт существующей схемы данных (из SQL дампа)

Если есть уже готовая схема данных, ее можно импортировать в MySQL Workbench для дальнейшей работы. Для импорта модели из SQL файла необходимо выбрать пункт меню File → Import → Reverse Engineer MySQL Create Script…, после чего выбрать нужный SQL-файл и активировать опцию «Execute».

В MySQL Workbench так же предусмотрен импорт и синхронизация модели данных напрямую с удалённым сервером. Для этого потребуется создать подключение удалённого доступа к MySQL (данная опция будет рассмотрена в следующей лабораторной работе).

1.7 Программа работы

1. Произвести краткое описание предметной области (предметная область лабораторной работы соответствует варианту предметной области из курсового проекта). Подробное описание предметной области включить в

раздел «Анализ предметной области» курсового проекта. Для выполнения этого этапа необходимо:

- проанализировать информационные потребности пользователей;
- сформировать состав документов, подлежащих включению в БД;
- разработать состав и форму представления информации по каждому документу;
- создать кодификаторы для упорядочения данных в БД;
- определить задачи и функции системы.

2. Построить внешнюю модель разрабатываемой БД.
3. Разработать диаграмму сущность-связь (ERD) в нотации П.Чена.
4. Установить MySQL Workbench. Изучить возможности СУБД.

1.8 Содержание отчета

Отчет по лабораторной работе должен содержать:

1) титульный лист;
2) цель работы;
3) краткое описание предметной области;
4) внешняя модель разрабатываемой БД;
5) простая сетевая модель разрабатываемой БД;
6) диаграмма сущность-связь (ERD) в нотации П.Чена;
7) выводы по работе.

1.9 Контрольные вопросы

1. Этапы проектирования баз данных.
2. Какова суть концептуального проектирования БД.
3. В чем суть логического проектирования БД?
4. В чем суть физического проектирования БД?
5. Модель «сущность-связь». Основные компоненты.
6. Как представляется диаграмма «сущность-связь» в нотации П. Чена?
7. Какие существуют типы связей между сущностями и чем они отличаются?
8. Какова цель модели, основанной на ключах. В чем ее особенность.
9. Процесс построения информационной модели БД.
10. Суть прямого и реверсного проектирования БД.

Лабораторная работа № 2

ОРГАНИЗАЦИИ АРХИТЕКТУРЫ «КЛИЕНТ-СЕРВЕР» В СИСТЕМАХ БАЗ ДАННЫХ. ПОСТРОЕНИЕ ПОЛНОЙ АТРИБУТИВНОЙ МОДЕЛЬ БАЗЫ ДАННЫХ В НОТАЦИИ IDEF1X

2.1 Цель работы

Углубление теоретических знаний и построение полной атрибутивной модель базы данных в нотации IDEF1X. Изучение принципов настройка SQL-сервера на виртуальной машине под управлением операционной системы Ubuntu и по созданию сетевого взаимодействия между реальной машиной, выступающей в роли SQL-клиента и виртуальной, играющей роль SQL-сервера.

2.2 Методические указания по настройке виртуальной машины

Под понятием виртуальная машина (Virtual Machine) понимают программную или аппаратную систему, которая эмулирует аппаратное обеспечение некой платформы (гостевая платформа), исполняющая программы для гостевой платформы средствами хост-платформы. Виртуальная машина может виртуализировать некую платформу, создавая на ней независимые, изолированные среды для работы операционных систем и программ [7].

Таким образом, виртуальная машина предоставляет возможность на одном реальном, физическом компьютере, создавать несколько виртуальных компьютеров, устанавливать на них различные операционные системы, программы и пр.

Технология виртуализации пришла из серверной инфраструктуры, где виртуальные машины используются с целью создания максимальной загрузки сервера и уменьшения простоев оборудования [8].

Виртуальные машины используют для решения следующих задач:

- оптимизация использования серверных ресурсов;
- информационная защита, а также ограничение возможностей некоторых программ;
- исследования новой компьютерной архитектуры или программного обеспечения;
- эмуляция различных компьютерных архитектур;
- создание вредоносного кода (предоставляющего, например, удаленный контроль над виртуальной машиной злоумышленнику).
- моделирование компьютерных сетей;
- тестирование и отладка программного обеспечения.

Одной из самых популярных программ виртуализации является VirtualBox.

VirtualBox (Oracle VM VirtualBox) – бесплатный программный продукт виртуализации для операционных систем Microsoft Windows, Linux, FreeBSD, Mac OS X, Solaris/OpenSolaris, ReactOS, DOS и других. Свободно распространяется под универсальной общественной лицензией GNU. При помощи

общих сетевых ресурсов VirtualBox позволяет осуществлять обмена файлами между «внешней» и «внутренней» операционными системами.

Перед тем, как начать установку, необходимо выделить новой системе некоторые количество системных ресурсов – максимально доступное количество оперативной памяти и ядер процессора, а также назначить максимальный уровень их загрузки. Также для настройки доступны более тонкие параметры аппаратной конфигурации.

2.2.1 Установка VirtualBox

Прежде всего, на официальном сайте https://www.virtualbox.org необходимо загрузить установщик VirtualBox, активировав опцию «Download VirtualBox 5.1» (рис. 2.1).

Рис. 2.1. Официальная страница для загрузки установщика VirtualBox

В следующем окне необходимо выбрать соответствующую операционную систему для VirtualBox. В представленном ниже примере выбран установщик для Windows «VirtualBox 5.1 for Windows hosts» (рис. 2.2).

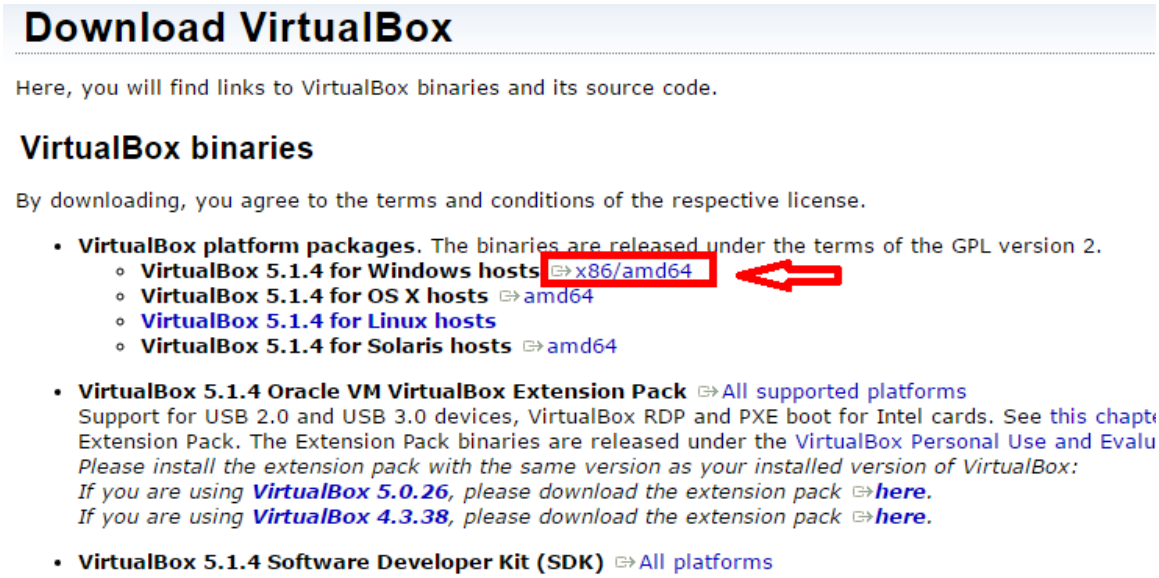


Рис. 2.2. Выбор операционной системы для VirtualBox

После полной загрузки установщика можно преступить к установке VirtualBox. Никаких дополнительных настроек во время установки VirtualBox производить не надо, необходимо несколько раз активировать опцию «Next», утвердительно ответить на предупреждение о временном отключении компьютера от сети и запустить процесс установки.

По завершении установки необходимо перезагрузить компьютер.

2.2.2 Создание виртуальной машины

Для создания виртуальной машины в VirtualBox необходимо активировать опцию «Создать» (рис. 2.3).

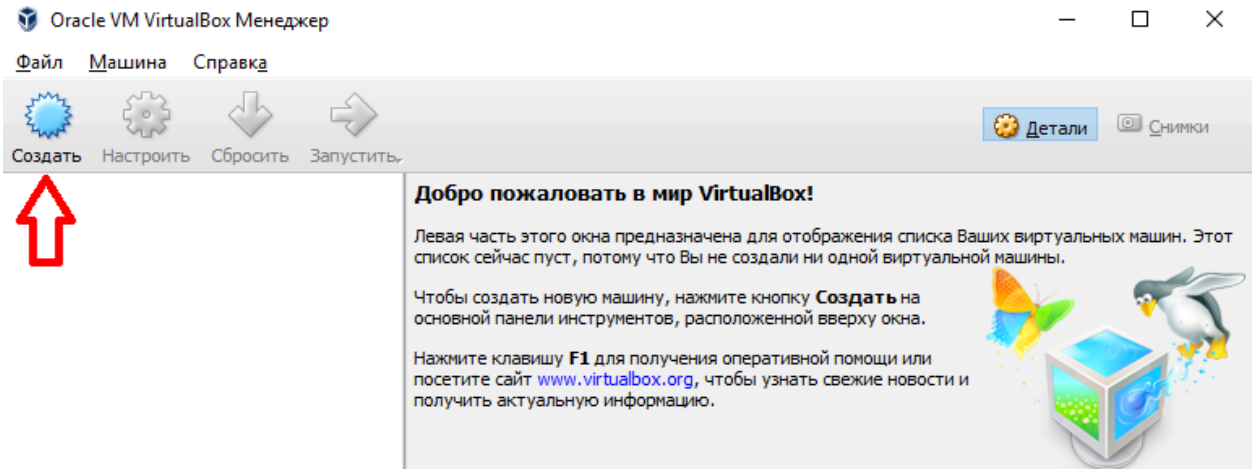


Рис. 2.3. Создание новой виртуальной машины

В открывшемся окне в представленных полях необходимо вписать следующие данные (рис. 2.4):

– имя – вписать название создаваемой виртуальной машины;
– тип – выбрать Linux;
– версия – выбрать Ubuntu (64-bit);

Затем необходимо активировать опцию «Next».

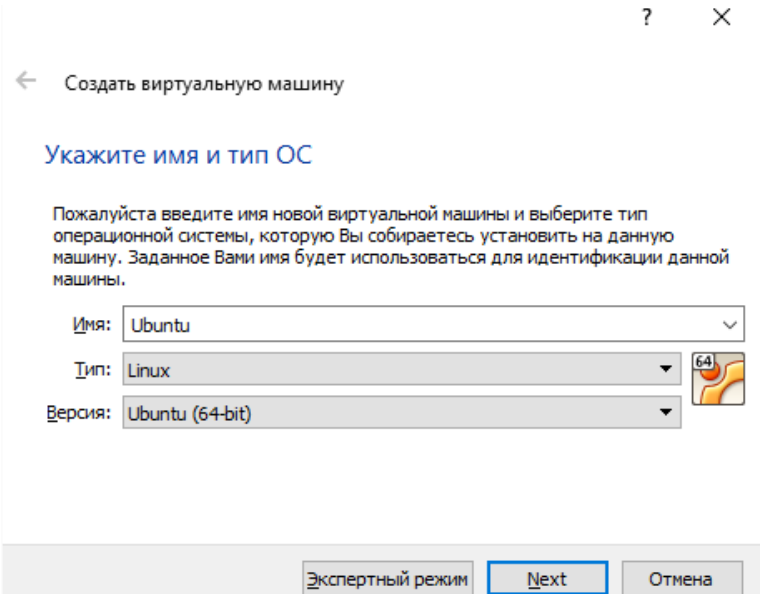


Рис. 2.4. Выбор типа виртальнй машины

В следующем окне необходимо установить выделяемый объем оперативной памяти для виртуальной машины. В представленном ниже примере (рис. 2.5) выделен 1ГБ. Активировать опцию «Next».

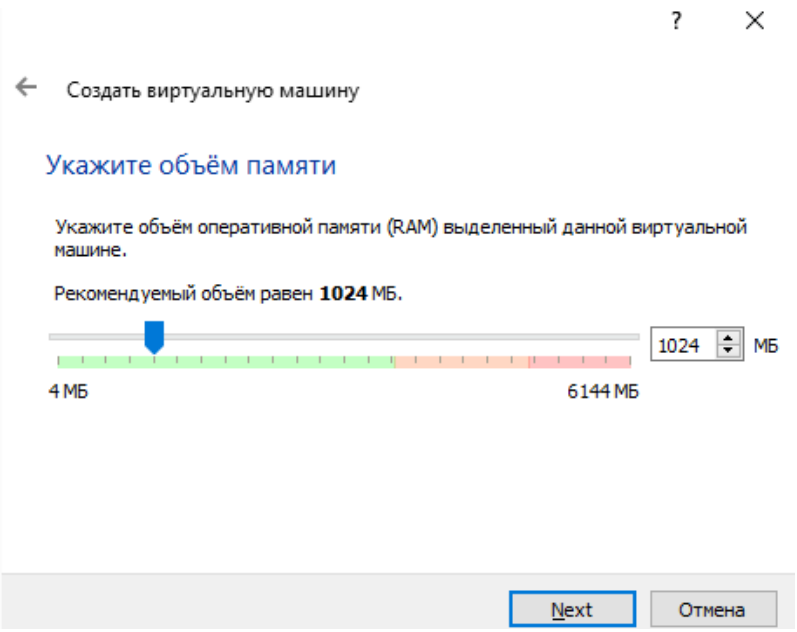


Рис. 2.5. Установка объема оперативной памяти
для виртуальной машины

В следующем окне из предложенного списка необходимо выбрать вариант «Создать новый виртуальный жесткий диск» и активировать опцию «Создать» (рис. 2.6).

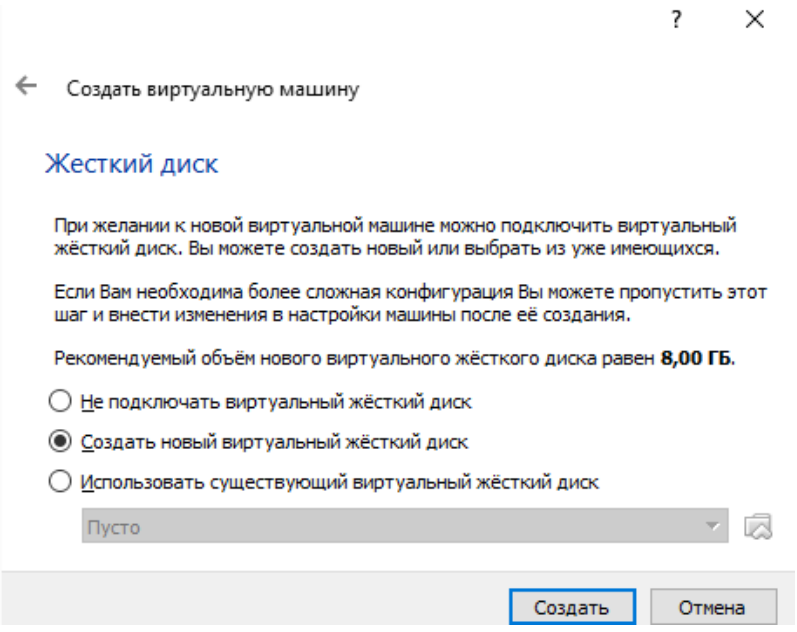


Рис. 2.6. Создание нового виртуального диска

В следующем окне необходимо указать тип файла, определяющего формат жесткого диска – VDI, которые выбран по умолчанию и активировать опцию «Next» (рис. 2.7).

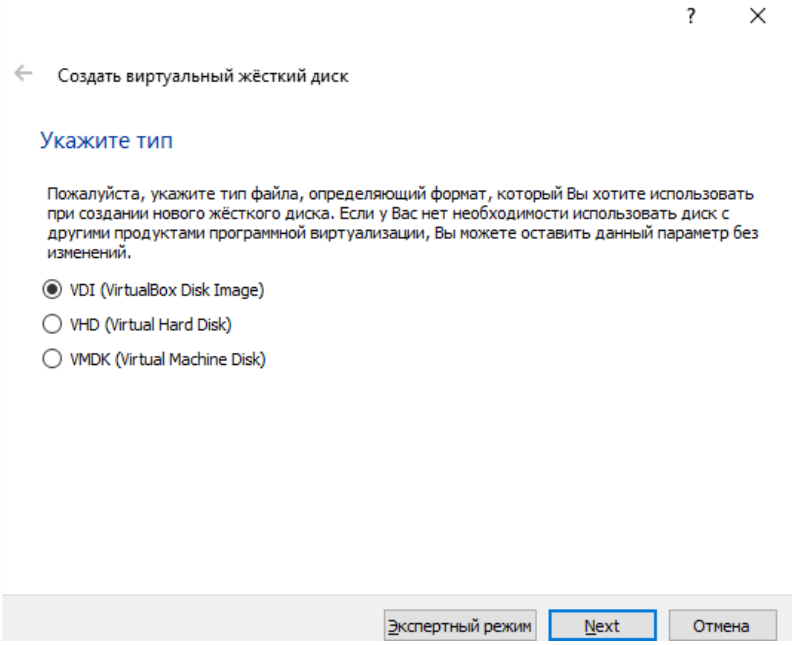


Рис. 2.7. Выбор типа файла, определяющего формат жесткого диска

В следующем окне необходимо выбрать вариант хранения данных «Динамический виртуальный жесткий диск» и активировать опцию «Next».

Далее необходимо указать объем виртуального жесткого диска. Система рекомендует 8ГБ, можно оставить без изменений и активировать опцию «Создать».

Теперь виртуальная машина создана, и ее можно увидеть в списке виртуальных машин (рис. 2.8).

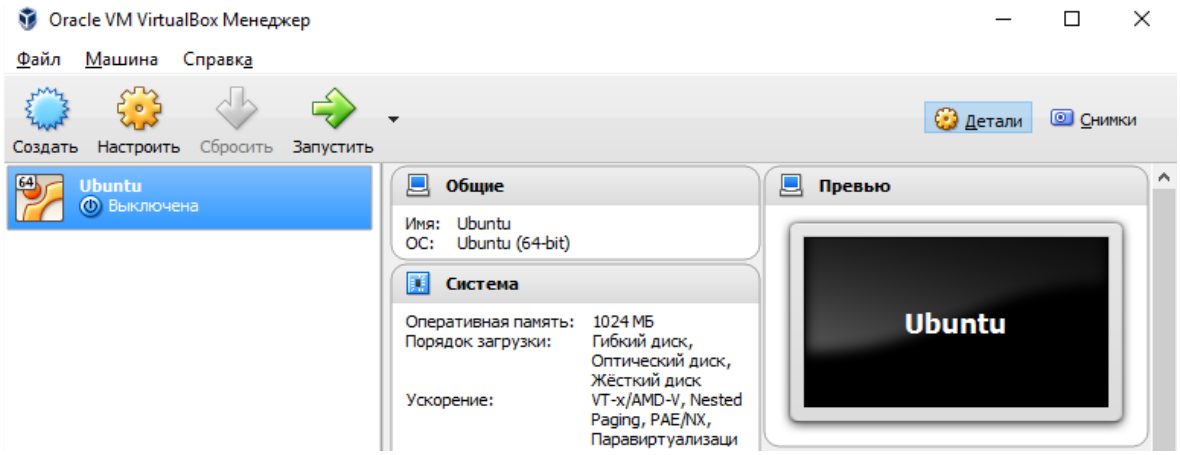


Рис. 2.8. Список виртуальных машин, созданных в VirtualBox

Теперь можно приступать к установке Ubuntu Desktop на созданную виртуальную машину.

2.2.3 Установка операционной системы Ubuntu Desktop на виртуальную машину

Прежде чем приступить к установке Ubuntu, необходимо скачать сам дистрибутив. Для этого надо зайти на официальный сайт Ubuntu http://www.ubuntu.com и в меню «Download» выбрать пункт «Desktop» и активировать опцию «Alternative downloads and torrents» (рис. 2.9).

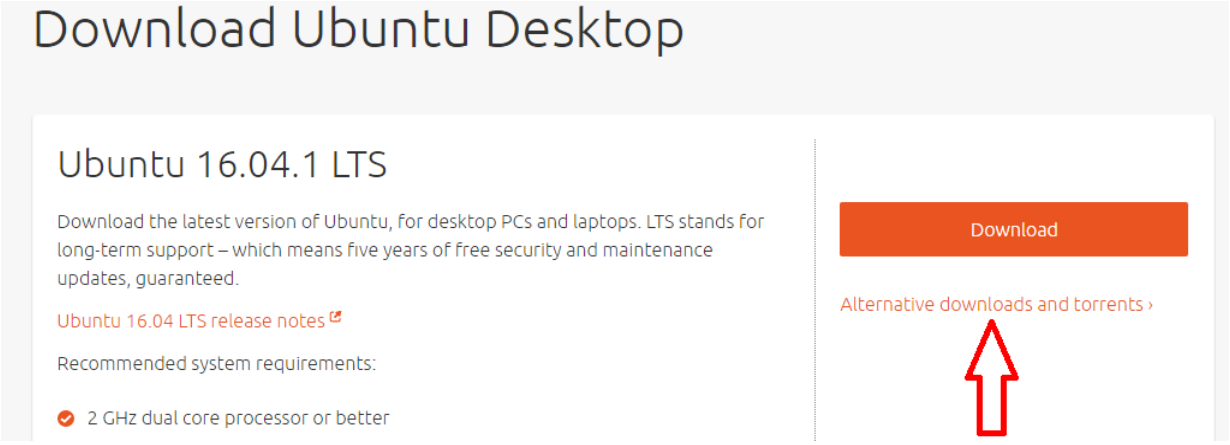


Рис. 2.9. Страница официального сайта для загрузки установщика Ubuntu

В появившемся окне необходимо выбрать раздел «Ubuntu 16.04.1 Desktop (32 bit)» и активировать загрузку torrent-файла установщика (подходящего под тип процессора реальной машины), с помощью которого затем можно загрузить сам установщик операционной системы Ubuntu (рис. 2.10).

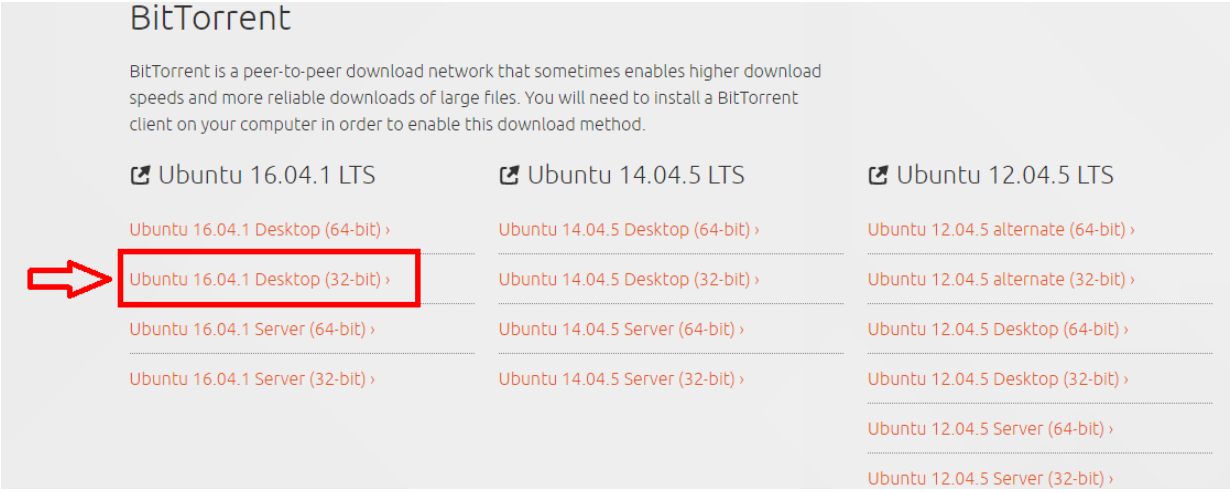


Рис. 2.10. Выбор версии операционной системы Ubuntu

После закачки установщика операционной системы можно приступать к установке Ubuntu на виртуальную машину.

Для этого необходимо поместить только что скачанный образ ISO в привод виртуальной машины. В разделе «Носители» в опции «Оптический привод» необходимо выбрать только что скачанный образ дистрибутива Ubuntu Desktop (рис. 2.11) и запустить виртуальную машину, активировав опцию «Запустить».

Автоматически начнется установка операционной системы Ubuntu 16.04.1 Desktop (32 bit) на виртуальную машину.

1. Необходимо выбрать язык операционной системы и активировать опцию «Установить Ubuntu».

2. Необходимо активировать опцию «Загрузить обновления во время установки Ubuntu», но для того, чтобы обновления загрузились необходимо установить соединение с Интернет и активировать опцию «Продолжить».

3. На следующем шаге установки Ubuntu необходимо выбрать пункт «Стереть диск и установить Ubuntu», активировать опцию «Установить сейчас» и принять появившееся предупреждение.

4. На следующем шаге установки Ubuntu необходимо задать имя компьютера и учетные данные пользователя: логин, пароль. Также необходимо активировать опцию «Входить в систему автоматически» для того, чтобы при каждом запуске виртуальной машины не вводить пароль пользователя. После активации опции «Продолжить» начнется установка операционной системы Ubuntu на выбранную виртуальную машину.

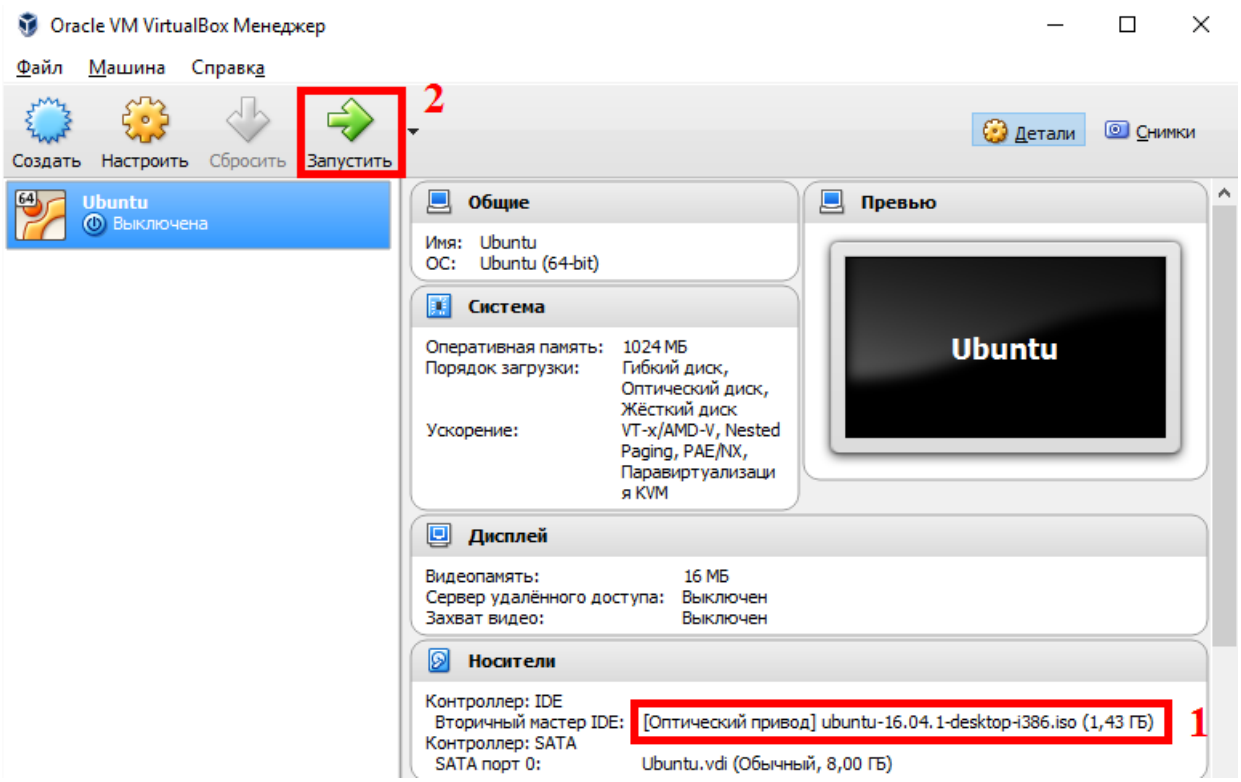


Рис. 2.11. Выбор образа для установки операционной системы

По завершении установки операционной системы будет выведено соответствующее сообщение с предложением перезапустить компьютер.

После перезапуска компьютера и активации опции «Enter», на виртуальной машине отобразится рабочий стол операционной системы Ubuntu.

Последовательность действий по установке MySQL и MySQL Workbench на Ubuntu описаны в методических указаниях к лабораторной работе № 1 по дисциплине «Технологии баз данных».

2.2.4 Настройка сети между реальной и виртуальной машинами

Для создания и настройки виртуальной компьютерной сети между реальной и виртуальной машинами средствами программы VirtualBox в списке виртуальных машин необходимо выбрать ту машину, с которой необходимо будет настроить сетевое подключение, и активировать опцию «Настроить» (рис. 2.12).

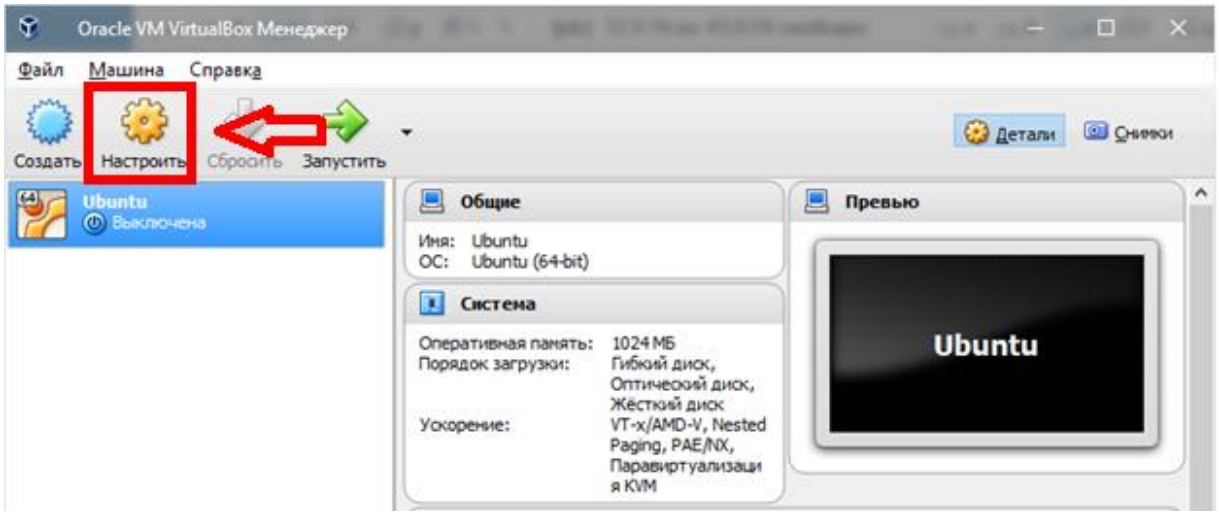


Рис. 2.12 . Меню настройки сетевой машины

В левой части меню необходимо перейти в раздел «Сеть».

Каждая виртуальная машина имеет 4 условных адаптера, каждый из адаптеров имеет 5 профилей настройки.

Настройку виртуальной компьютерной сети между реальной и виртуальной машинами средствами программы VirtualBox можно реализовать двумя способами:

– используя тип соединения «Сетевой мост» – в этом случае условный сетевой адаптер подключается и работает напрямую с физическим адаптером минуя хост-машину. Данный тип работы адаптера есть смысл использовать, когда необходимо предоставить доступ к виртуальной машине другим участникам локальной физической сети;

– используя тип соединения «Виртуальный адаптер хоста» – при таком режиме работы есть возможность взаимодействия как между виртуальными машинами, а также виртуальной машиной и хостом. При использовании адаптера хоста отсутствует возможность взаимодействия с другими участниками физической локальной сети.

Настройка сетевого моста (1 способ). Для данного типа соединения необходимо выбрать тип подключения «Сетевой мост (Bridge)» и в дополнительных настройках активировать неразборчивый режим, выбрав надстройке «Разрешить все» (рис. 2.13).

Используя этот тип соединения, виртуальная машина ничем не отличается от хост-машины для других участников сети, сетевой адаптер при такой настройке служит мостом между виртуальной и физической сетью. Таким образом, данный тип работы адаптера используется, когда необходимо предоставить доступ к виртуальной машине другим участникам локальной физической сети.

Условный сетевой адаптер подключается и работает напрямую с
А клческим адаптером минуя хост-машину.

Если компьютер имеет несколько сетевых интерфейсов, то в поле «Имя» необходимо указать через какой из них будет осуществляться взаимодействие.

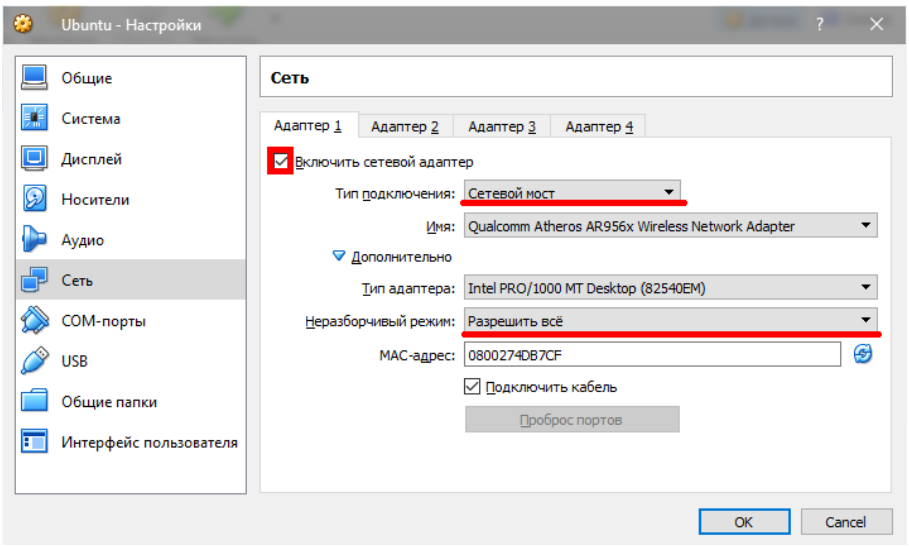


Рис. 2.13. Настройка сетевого подключения между между реальной и виртуальной машинами

Настройка виртуального адаптера хоста (2 способ). Для данного типа соединения необходимо выбрать тип подключения «Виртуальный адаптер хоста» и в дополнительных настройках активировать неразборчивый режим, выбрав надстройке «Разрешить все» (рис. 2.14). В этом случае используется специальное устройство – VirtualBox Host-Only Ethernet Adapter, которое создает подсети и назначает IP-адреса гостевым операционным системам.

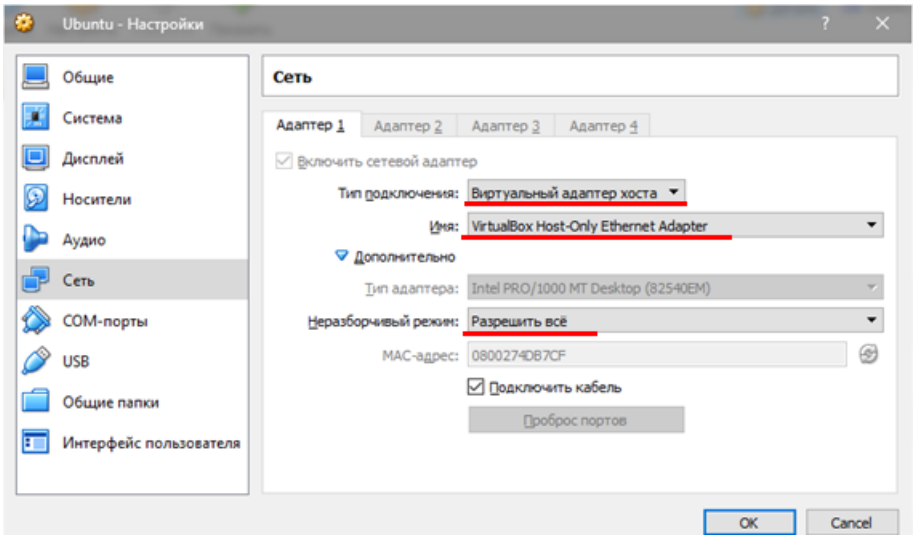


Рис. 2.14. Настройка сетевого подключения между между реальной и виртуальной машинами

2.2.5 Установка статического IP-адреса на виртуальной машине

По умолчанию виртуальной машине автоматически присваивается динамический IP-адрес, который при каждом новом запуске может быть разным. Поэтому, для того, чтобы обеспечить постоянное стабильное А клюючение к серверу MySQL, тип IP-адреса, выделяемого виртуальной машине, необходимо изменить с динамического на статический.

Для этого в операционной системе Ubuntu на виртуальной машине необходимо зайти в меню «Настройки» и выбрать раздел «Сеть» (рис. 2.15).

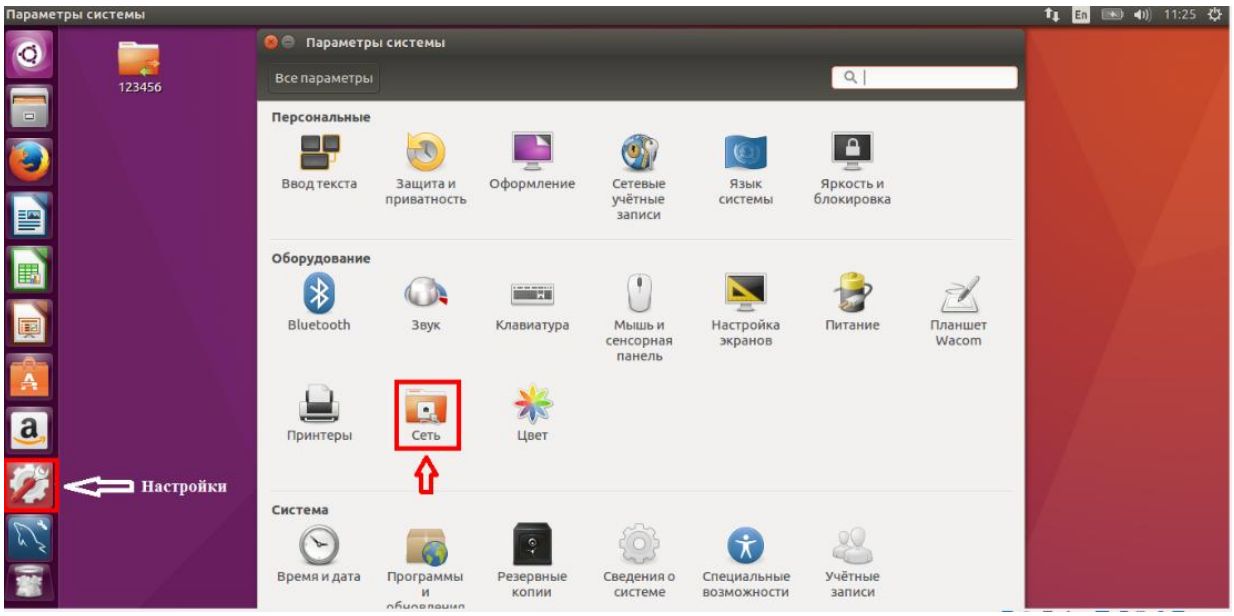


Рис. 2.15. Вход в меню сетевых настроек виртуальной машины

В появившемся окне необходим запомнить текущий IP-адреса виртуальной машины (на рис. 2.16 это адрес Ipv4 – 192.168.0.105) и шлюз по умолчанию, если указан (на рисунке 2.16 это маршрут по умолчанию – 192.168.0.1). Затем необходимо выбрать тип соединения – «Проводное» и активировать опцию «Параметры…» (см. рис. 2.16).

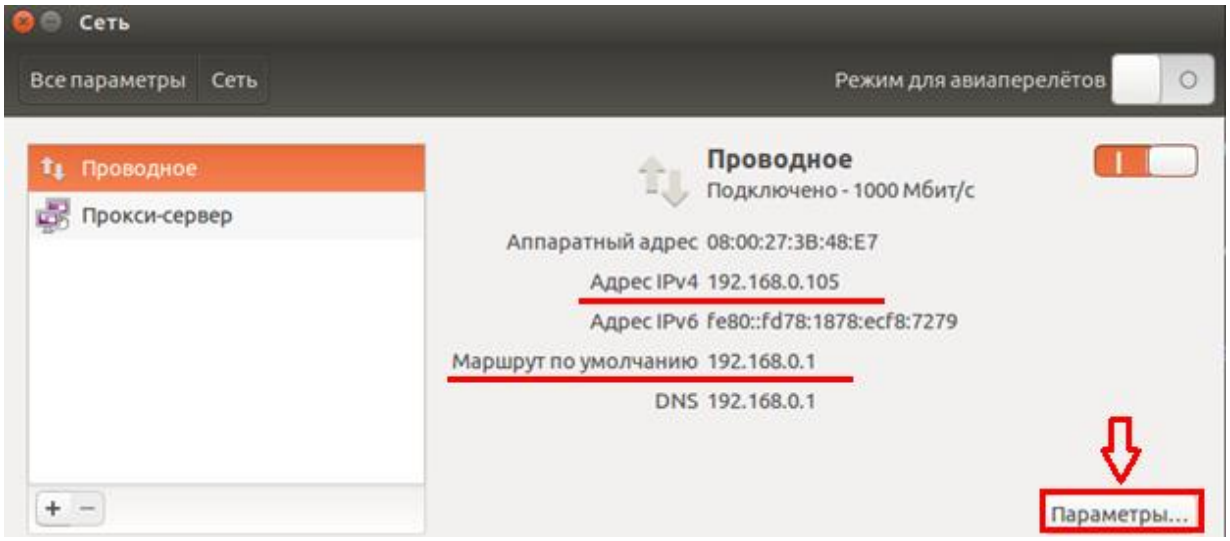


Рис. 2.16. Меню сетевых настроек виртуальной машины

Затем, в окне изменения настроек проводного соединения необходимо выбрать раздел «Параметры Ipv4» и в меню «Способ настройки» активировать опцию «Вручную» вместо предложенной «Автоматически (DHCP)» (рис. 2.17).

Затем необходимо сделать динамически выделенный виртуальной Ашине IP-адрес статическим. Для этого необходимо активировать опцию «Добавить» и в разделе «Адрес» в столбце «Адрес» вручную прописать IP-адрес, который был присвоен динамически (см. рис. 2.16, поле «Адрес Ipv4»), в столбце «Маска подсети» – указать стандартное значение маски –

255.255.255.0, а в столбце «Шлюз» – значение маршрута по умолчанию (см. рис. 2.16), и затем активировать опцию «Сохранить» (см. рис. 2.17).

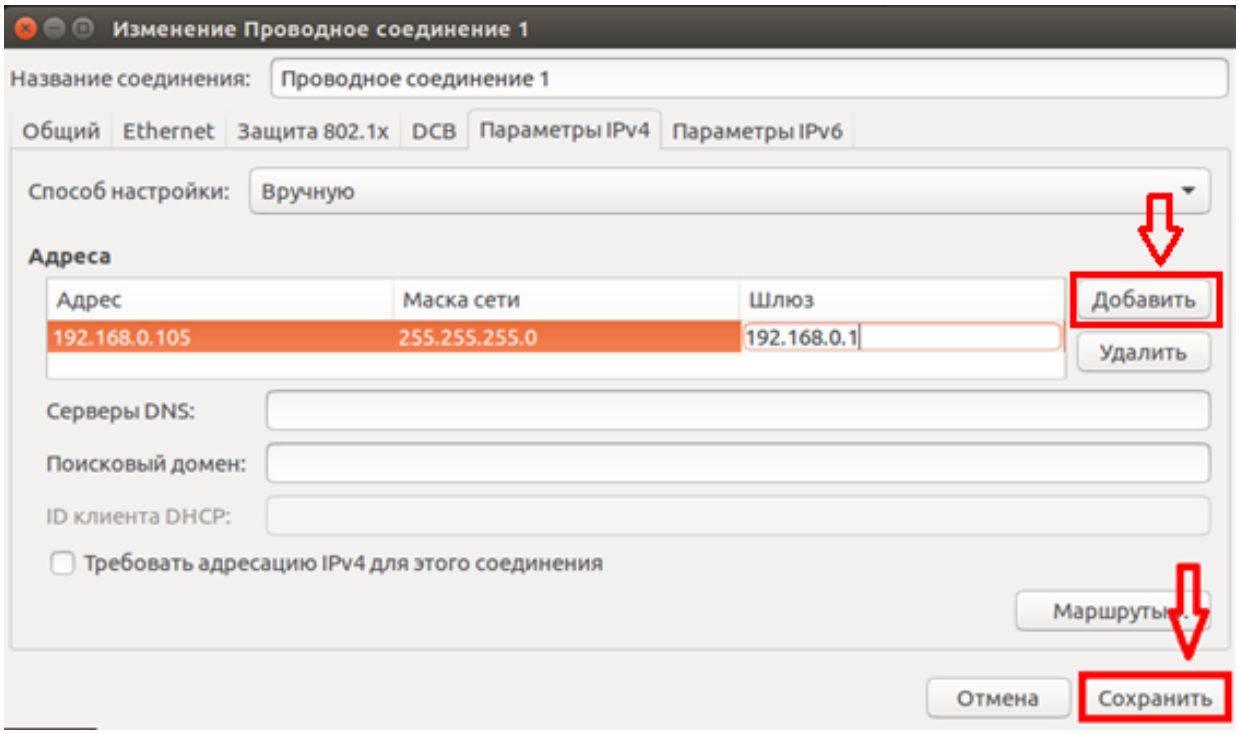


Рис. 2.17. Изменение настроек проводного соединения

Настройка статического IP-адреса для виртуальной машины завершена.

2.2.6 Настройка удаленного соединения с сервером MySQL

По умолчанию сервер MySQL принимает соединения только с локальной машины. Для того, чтобы разрешить подключаться к нему с других Ашин, необходимо открыть конфигурационный файл **mysqld.cnf**, с возможностью редактирования, с помощью следующей команды консоли Ubuntu:

```
sudo gedit /etc/mysql/mysql.conf.d/mysqld.cnf
```

И в появившемся окне найти и замените строку:

```
bind-address            = 127.0.0.1
```

на

```
bind-address            = 0.0.0.0
```

Данная команда позволяет настроить MySQL на ожидание подключений от компьютеров в сети, если в параметре «bind-address» указать IP-адрес 0.0.0.0, то подключение будет разрешено с любого компьютера.

После сохранения произведенных изменений к серверу MySQL можно подключаться извне.

Осталось создать пользователей, которым разрешено удаленное подключение. Для этого необходимо запустить консоль MySQL командой:

```
mysql -u root -p
```

И создать пользователя командой:

```
GRANT ALL PRIVILEGES ON *.* TO username@"remote_addr"
        IDENTIFIED BY 'password' WITH GRANT OPTION;
```

Вместо значения «**remote_addr**» необходимо указать адрес хоста с которого планируется подключение или можно указать значение «**%**» тогда подключение будет разрешено с любого адреса, в поле «**password**» необходимо указать пароль пользователя. Например, командой, представленной ниже, можно добавить пользователя с именем **root** и паролем **12345** для удаленного подключения к серверу MySQL:

```
GRANT ALL PRIVILEGES ON *.* TO
root@"%" IDENTIFIED BY '12345' WITH GRANT OPTION;
```

Затем необходимо выйти из консоли MySQL (командой exit или комбинацией клавиш Ctrl + Z) и выполнить команду для обновления таблиц MySQL, в которых хранится информация о пользователях:

```
mysqladmin -u root -p flush-privileges
```

2.2.7 Создание удаленного подключения к серверу MySQL

Теперь можно настроить удаленное подключение к серверу MySQL с клиентской (реальной) машины.

Для этого на стартовом окне приложения MySQL Workbench в разделе подключений к серверу MySQL «MySQL Connections» необходимо добавить новое подключение, активировав опцию ⊕.

В появившемся окне необходимо указать IP-адрес машины, на которой развернут MySQL-сервер (например, 192.168.0.105), имя пользователя, который подключается, и пароль (рис. 2.18).

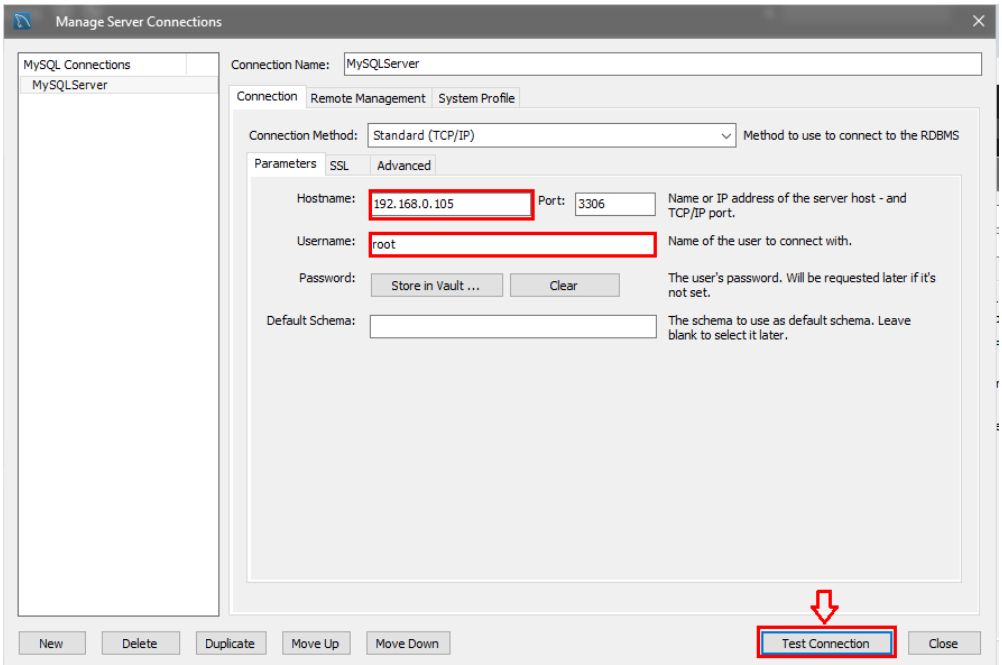


Рис. 2.18. Настройка подключения к удаленному серверу MySQL

Активировав опцию «Test connection» необходимо проверить статус подключения. Если все сделано правильно, появится сообщение об успешном установлении соединения с сервером MySQL (рис. 2.19).

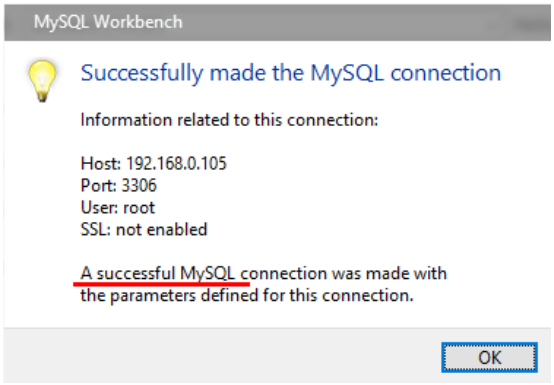


Рис. 2.19. Проверка подключения к удаленному серверу MySQL

2.2.8 Экспорт модели MySQL Workbench на SQL-сервер

Самый быстрый путь для того, чтобы схема данных из MySQL Workbench попала на сервер – создание SQL дампа mwb-модели. Для этого не потребуется создавать удаленное подключение в программе, однако этот способ хорош лишь в случае, если требуется однократная заливка структуры и базовых данных на сервер. Дальнейшая поддержка, обновление и синхронизация модели данных в этом случае будет весьма проблематична.

Для экспорта существующей модели MySQL Workbench на SQL-сервер необходимо открыть модель и выбрать пункт меню «File → Export → Forward Engineer SQL CREATE Script…». На рис. 2.20 представлено окно настроек экспорта.

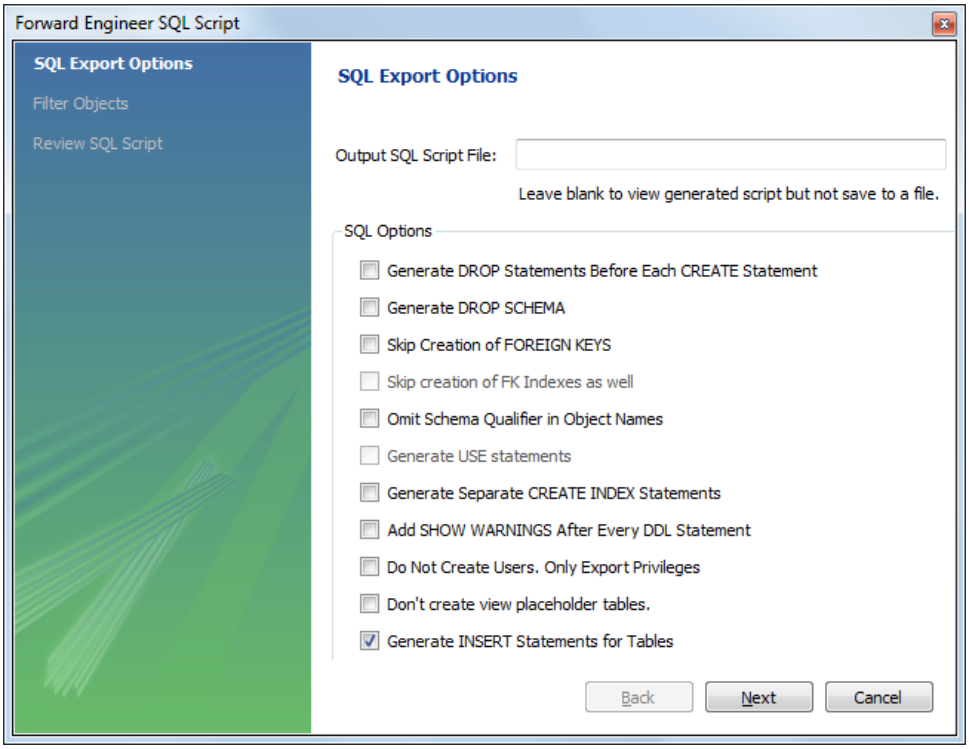


Рис. 2.20. Окно настроек экспорта модели MySQL Workbench на сервер

Если требуется записать SQL-скрипт в файл, необходимо указать путь до файла в поле «Output SQL Script File» (если оставить поле пустым, SQL-скрипт можно будет скопировать на последнем шаге в буфер обмена).

Настройки стандартные, чтобы понять их суть, достаточно перевести их названия. Если активировать опцию «Generate INSERT Statements for Tables» в дамп будут включены базовые данные, располагающиеся во вкладке «Inserts» интерфейса редактирования таблиц модели. После активации опции «Next» появится список объектов, которые можно экспортировать. Для экспорта таблиц необходимо выбрать опцию «Export MySQL Table Objects», а чтобы экспортировать их выборочно, можно дополнительно активировать опцию «Show Filter» и выбираем нужные таблицы (рис. 2.21).

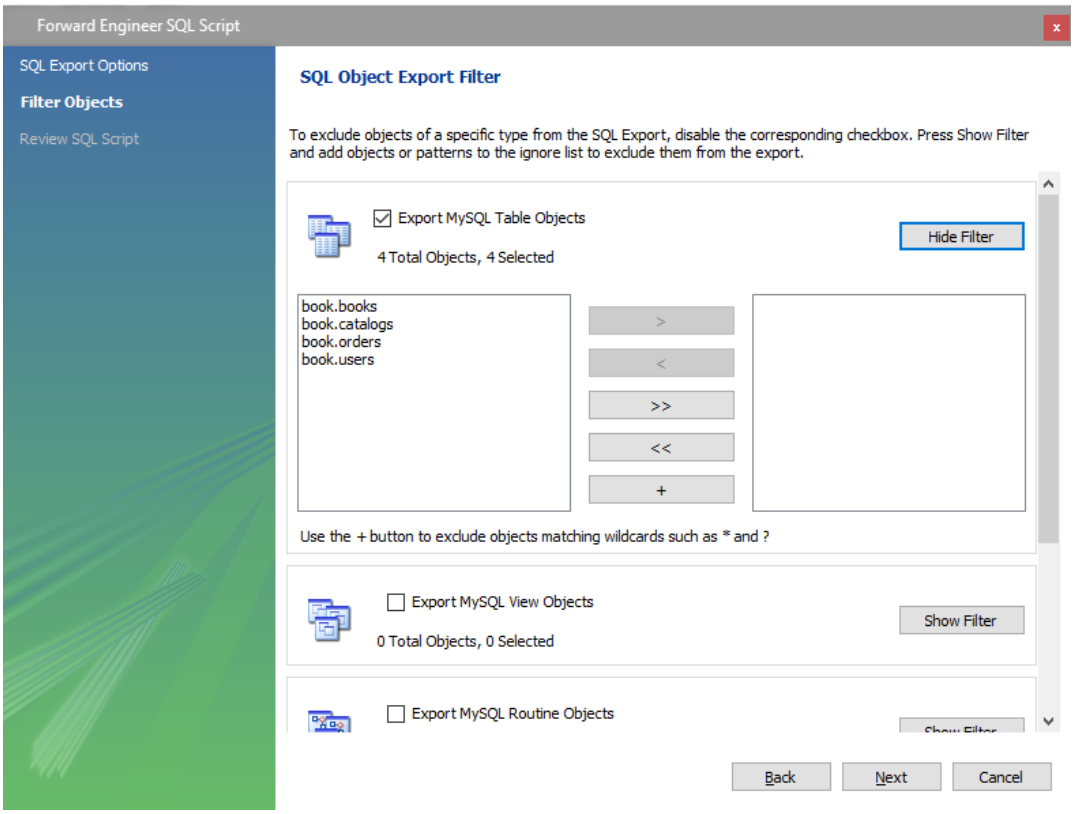


Рис. 2.21. Выбор объектов для экспорта

Активировав опцию «Next», в следующем окне появится готовый SQL-скрипт, который, при необходимости, можно скопировать его буфер обмена или записать в файл.

2.2.9 Синхронизация структуры данных

Для синхронизации структуры базы данных и локальной модели в MySQL Workbench существует специальный инструмент. Необходимо открыть нужную модель и выбрать пункт меню «Database → Synchronize Model…», после чего необходимо выбрать сохраненное в п. 3.2 удаленное подключение и отредактировать его параметры.

Для подключения к базе данных необходимо активировать опцию «Next». В новом окне появится список моделей (в левой колонке) и баз данных (в правой колонке), доступные для синхронизации (рис. 2.22).

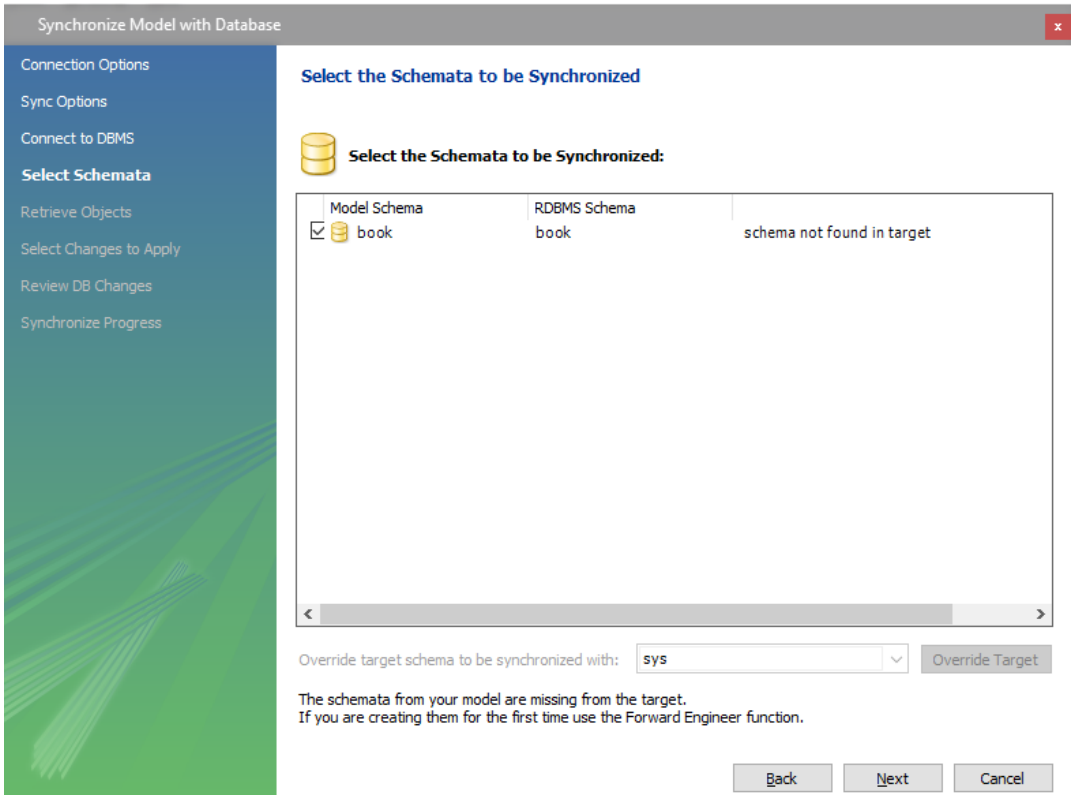


Рис. 2.22. Доступные модели и базы данных для синхронизации

Выбрав нужную базу и схему, необходимо снова активировать опцию «Next», запустив тем самым процедуру сравнения структур удаленной базы данных и локальной модели. После завершения процедуры появится список различий между локальной схемой данных и удалённой базой (рис. 2.23).

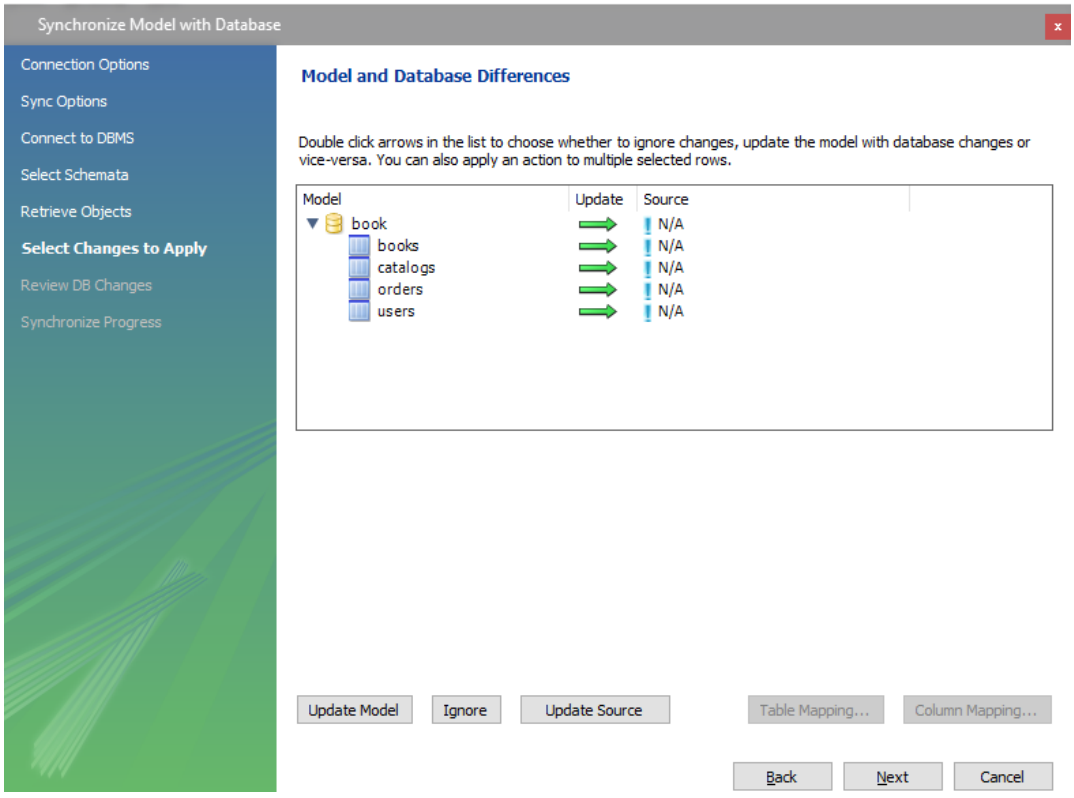


Рис. 2.23. Список различий между локальной и удаленной схемами данных

На данном этапе можно настроить объединение таблиц: протолкнуть изменения на сервер («Update Source»), втянуть в локальную модель конфигурацию с сервера («Update Model») или игнорировать отличия («Ignore»). Кроме того, доступен как вариант настройки для всей базы, так и отдельно для каждой таблицы. При выделении одной из таблиц и выборе способа объединения можно увидеть SQL-запросы, которые выполнятся в процессе синхронизации, а активировав опцию «Next» – увидеть полный стек этих запросов.

Просмотрев SQL-запросы, активировав опцию «Execute >», начнется выполнение синхронизации. Если все пройдет успешно, появится сообщение, представленное на рис. 2.24.

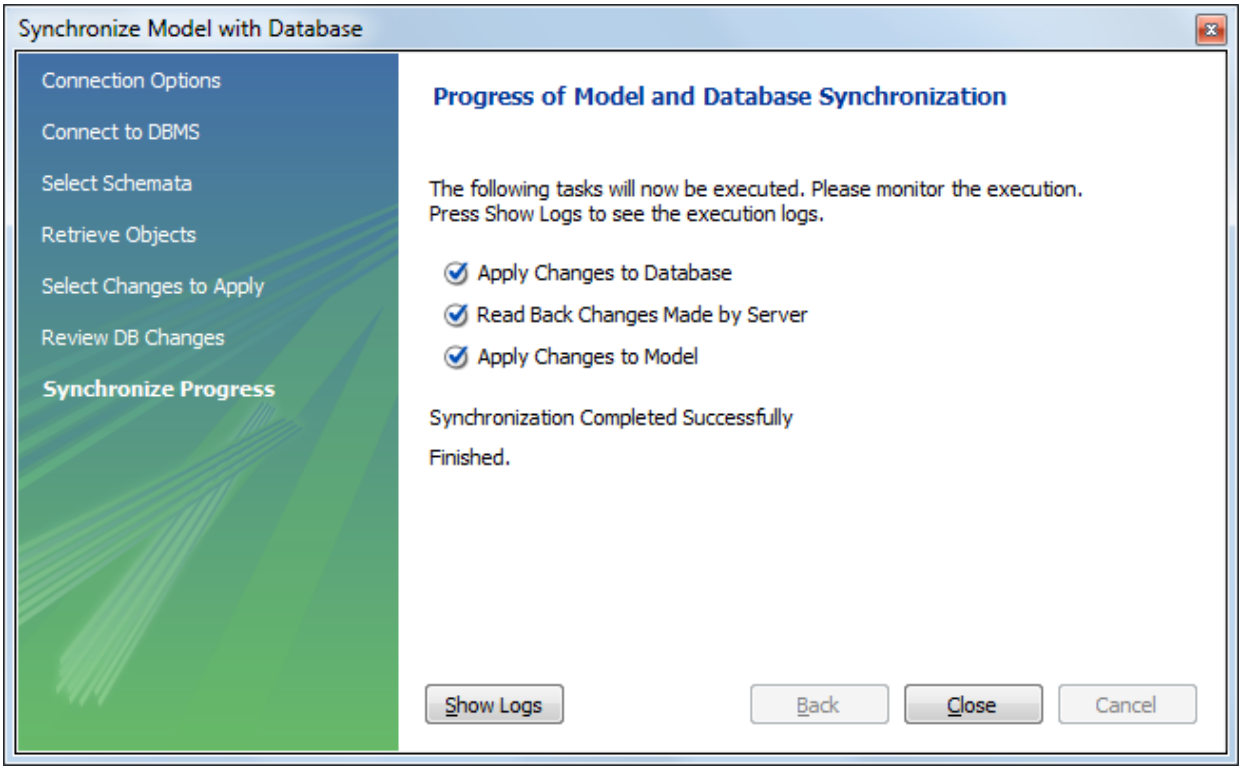


Рис. 2.24. Отчет об успешно выполненной синхронизации

В случае возникновения ошибок их лог отобразится в этом же диалоговом окне.

2.2.10 Выгрузка на сервер схемы и стартовых данных

Описанная выше синхронизация осуществляет лишь объединение структуры схемы данных удаленной базы и локальной модели, но никак не затрагивает стартовые данные, внесённые в модель («Inserts»). Если требуется выгрузить их, необходимо выбрать пункт меню «Database → Forward Engineer…», затем выбрать одно из сохраненных ранее подключений (или создать новое) и активировать опцию «Next». В остальном механизм выгрузки аналогичен механизму экспорта mwb-модели. Его можно так же использовать, если требуется простая выгрузка схемы данных на сервер без синхронизации.

2.3 Программа работы

1. Нормализовать отношения в базе данных, построенной в лабораторной работе №1, до третьей и четвертой нормальных форм.
2. Построить полную атрибутивную модель базы данных в нотации IDEF1X.
3. Создать виртуальную машину, установить на ней операционную систему Ubuntu и настроить SQL-сервер.
4. Настроить удаленное подключение к виртуальной (серверной) Ашине с реальной (клиентской) и выгрузить на сервер разработанную в MySQL Workbench схему.

2.4 Содержание отчета

Отчет по лабораторной работе должен содержать:

1) титульный лист;
2) цель работы;
3) приведенная в лабораторной работе № 1 простая сетевая модель к древовидной;
4) модель данных, основанную на ключах (KB) по методологии IDEF1 для каждого полученного дерева;
5) этапы нормализации для каждого полученного дерева;
6) полная атрибутивная модель базы данных в нотации IDEF1X;
7) скриншоты результатов настройки сетевого подключения между виртуальной (серверной) и реальной (клиентской) машинами;
8) выводы по работе.

2.5 Порядок защиты работы

1. Показать работу программы.
2. Объяснить:
   - принципы методологии IDEF1X;
   - нормализация до 4НФ и 3НФ;
   - принципами организации архитектуры «клиент-сервер» в системах баз данных;
   - каково назначение сервера баз данных;
   - функции клиентского приложения баз данных.

2.6 Контрольные вопросы

1. Принципы работы в многопользовательских системах?
2. Расскажите об особенностях работы в системах типа «клиент-сервер».
3. Что такое распределенная база данных?
4. Какие задачи распределенных систем?
5. Какие существуют формы прозрачности распределенных систем?

6. Что такое распределение вычислений?
7. Что такое балансировка, и какие методы балансировки существуют?
8. Что такое распределение данных?
9. Сформулировать необходимость процесса нормализации БД.
10. Сформулировать способы повышения надежности данных.

Лабораторная работа № 3

ИССЛЕДОВАНИЕ ПРОЦЕССОВ ПОСТРОЕНИЯ ИНФОРМАЦИОННЫХ МОДЕЛЕЙ В СИСТЕМЕ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ POSTGRESQL

3.1 Цель работы

Углубление теоретических знаний и изучение возможностей и условий применения СУБД, реализующих поддержку пространственных данных на примере PostgreSQL. Получение практических навыков в построении физический модели БД.

3.2 Описание лабораторной установки

3.2.1 Возможности PostgreSQL

СУБД PostgreSQL – это кроссплатформенная свободно распространяемая объектно-реляционная СУБД, наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных.

PostgreSQL считается самой совершенной СУБД, распространяемой на условиях открытых исходных текстов. В PostgreSQL реализованы многие возможности, традиционно встречавшиеся только в масштабных коммерческих продуктах [1].

По данным многих тестов проводимых для СУБД PostgreSQL не значительно уступает по своей производительности Oracle (около 15 %). При этом одна из самых больших БД в мире – БД Yahoo работает на модифицированной версии СУБД PostgreSQL.

К основным возможностям PostgreSQL относятсяс [10]:

1) поддержка объектно-реляционной модели – работа с данными в PostgreSQL основана на объектно-реляционной модели, что позволяет задействовать сложные процедуры и системы правил;

2) простота расширения – в PostgreSQL поддерживаются пользовательские операторы, функции, методы доступа и типы данных;

3) полноценная поддержка SQL – PostgreSQL соответствует базовой спецификации SQL99;

4) гибкость API – позволяет легко создавать интерфейсы к реляционной СУБД PostgreSQL. В настоящее время существуют программные интерфейсы для Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/ C+ и Pike;

5) в PostgreSQL предусмотрена поддержка внутренних процедурных языков, в том числе специализированного языка PL/pgSQL, являющегося аналогом PL/SQL, процедурного языка Oracle. Одним из преимуществ PostgreSQL является возможность использования Perl, Python и TCL в качестве внутренних процедурных языков;

6) технология MVCC (Multiversion Concurrency Control) – используется в PostgreSQL для управления конкурентным доступом к данным на многовариантной основе. Эта технология позволяет предотвращать лишние блокировки (locking) операций чтения операциями, производящими обновление записей. PostgreSQL отслеживает все транзакции, выполняемые пользовате-

лями базы данных, что позволяет работать с записями без ожидания их освобождения. При запросе к БД каждая транзакция «видит» как бы снимок данных (версию) на момент этого снимка, а не текущее состояние данных. Таким образом, транзакции защищаются от просмотра незафиксированных данных, которые в данный момент могут только формироваться конкурентными транзакциями в тех же самых строках таблицы. Основное преимущество MMVC состоит в том, что чтение данных никогда не блокирует запись, а запись никогда не блокирует чтение. MMVC позволяет избегать явного блокирования на уровне таблиц и отдельных записей, которое используется в традиционных СУБД, и, таким образом, минимизирует блокирование данных и увеличивает производительность в многопользовательских системах БД. Также реализовано отслеживание взаимных блокировок;

7) клиент-серверная архитектура – в PostgreSQL используется архитектура «клиент-сервер» с распределением процессов между пользователями. Главный (master) процесс создает дополнительные подключения для каждого клиента, пытающегося установить соединение с PostgreSQL;

8) полнотекстовый поиск – начиная с версии 8.3 в ядро PostgreSQL включена функция полнотекстового поиска, который позволяет создавать такие запросы к текстовым документам, которые могут гибко настраиваться в зависимости от конкретных потребностей;

9) сохраняющая регистрация WAL (Write-Ahead Logging) – является стандартным методом для обеспечения целостности данных и представляет метод регистрации (журналирования) транзакций, при котором запись в журнале делается до записи данных. Суть WAL заключается в том, что изменения в файлах данных (таблицы и индексы) должны быть внесены только после записи в журнал (log), в котором фиксируются эти изменения. Эта процедура позволяет не переписывать страницы данных на диске при каждой транзакции, так как в случае аварии можно восстановить базу данных с помощью журнала. Механизм WAL обеспечивает следующие преимущества:

– повышение производительности работы СУБД за счет того, что записываются только внесенные изменения без переписывания всех данных в таблицах;

– повышение надежности хранения данных за счет предыдущего сохранения буферизованных данных в WAL;

– возможность отката состояния БД на любой момент времени, путем применения WAL к существующей резервной копии.

9) репликация и технология Hot Standby – начиная с версии 9.0, на основе WAL введена репликация по технологии Hot Standby, позволяющей получить на сервере вторую базу данных, которая является актуальной копией оригинальной базы данных, доступной лишь для чтения.

10) наследование – могут наследовать характеристики и наборы полей от других таблиц (родительских). При этом данные, которые добавляются в порожденную таблицу, автоматически будут принимать участие в запросах к родительской таблице;

11) гибкая настройка сервера – основной конфигурационный файл postgresql.conf включает более 150 настраиваемых параметров по разделам: файлы и пути к ним, авторизация и безопасность, выделение ресурсов и т.д. Дополнительный конфигурационный файл pg_hba.conf включает в себя настройку

доступа к отдельным БД, такие как указание конкретных IP-адресов и (или) сетей, из которых разрешен доступ, а также метод авторизации для доступа в БД и возможность включения безопасных (зашифрованных) соединений.

3.3.2 Типы данных PostgreSQL

С любым объектом данных, представленным в PostgreSQL, как и в любой СУБД, связывается определенный тип. Тип данных одновременно определяет и ограничивает разновидности операций, которые могут выполняться с этими данными [10].

Большинство типов данных PostgreSQL взято непосредственно из стандартов SQL, но существуют и другие, нестандартные типы данных (например, геометрические и сетевые типы). В таблице 3.1 перечислены основные базовые типы данных PostgreSQL, а также их синонимы (альтернативные имена).

Таблица 3.1 – Типы данных PostgreSQL

Тип данных	Описание	Стандарт
Логические и двоичные типы данных		
boolean, bool	Отдельная логическая величина (true или false)	SQL99
bit(n)	Битовая последовательность фиксированной длины	SQL92
bit varying(n), varbit(n)	Битовая последовательность переменной длины (до n бит)	SQL92
Символьные типы		
character(n), char(n)	Символьная строка фиксированной длины (ровно n символов)	SQL89
character varying(n), varchar(n)	Символьная строка переменной длины (до n символов)	SQL92
text	Символьная строка переменной или неограниченной длины	PostgreSQL
Числовые типы		
small int, int2	2-байтовое целое со знаком	SQL89
integer, int, int4	4-байтовое целое со знаком	SQL92
bigint, int8	8-байтовое целое со знаком, до 18 цифр	PostgreSQL
real, float4	4-байтовое вещественное число	SQL89
double precision, floats, float	8-байтовое вещественное число	SQL89
numeric(p.s), decimal (p.s)	Число из p цифр, содержащее 5 цифр в дробной части	SQL99
money	Фиксированная точность, представление денежных величин, считается устаревшим	PostgreSQL
serial	4-байтовое целое с автоматическим приращением	PostgreSQL
Время и дата		
date	Календарная дата (день, месяц и год)	SQL92
time	Время суток	SQL92
time with time zone	Время суток с информацией о часовом поясе	SQL92
timestamp	Дата и время	SQL92
interval	Произвольный интервал времени	SQL92
Геометрические типы		
box	Прямоугольник на плоскости	PostgreSQL
line	Бесконечная линия на плоскости	PostgreSQL

Продолжение табл. 3.1

Тип данных	Описание	Стандарт
Iseg	Отрезок на плоскости	PostgreSQL
circle	Круг с заданным центром и радиусом	PostgreSQL
Path	Замкнутая или разомкнутая геометрическая фигура на плоскости	PostgreSQL
point	Точка на плоскости	PostgreSQL
polygon	Замкнутый многоугольник на плоскости	PostgreSQL
Сетевые типы		
cidr	Спецификация сети IP	PostgreSQL
inet	Сетевой IP-адрес с необязательными битами подсети	PostgreSQL
macaddr	MAC-адрес	PostgreSQL
Системные типы		
oid	Идентификатор объекта (записи)	PostgreSQL
xid	Идентификатор транзакции	PostgreSQL

Помимо встроенных типов данных, пользователь может самостоятельно создавать новые необходимы ему типы и программировать для них механизмы индексирования с помощью методов GiST.

3.2.3 Установка и настройка программное обеспечение PostgreSQL

С официального сайта PostgreSQL http://www.postgresql.org/ необходимо осуществить загрузку стабильной версии программного обеспечения PostgreSQL http://www.enterprisedb.com/products-services-training/pgdownload для Вашей операционной системы.

PostgreSQL является свободным сервером баз данных и также, как и MySQL имеет оболочку проектирования pgAdmin, с использованием которой и будет осуществляться работа в среде PostgreSQL.

При установке сервера PostgreSQL и его дополнительного программного обеспечения будут запрошены параметры учетной записи пользователя. По умолчанию создается запись с логином postgres, пароль к которой устанавливает пользователь в момент установки. Аналогично MySQL, PostgreSQL идентифицируется хостом и номером порта (по умолчанию, 5432).

3.2.4 Установка PostGIS

PostGIS можно скачать и установить, как самостоятельно, так и полуавтоматически через Stack Builder, скачав пакет PostGIS. Способы установки отличаются лишь методом получения установщика PostGIS. В первом случае Stack Builder запустится сам: Пуск → PostgreSQL→ Application Stack Builder. Последнюю версию PostGIS можно скачать на сайте http://postgis.refractions.net.

Можно установить PostGIS сразу после завершения установки PostgreSQL, активировав опцию Stack Builder по завершении установки.

Необходимо выбрать установленный сервер.

Из пункта Spatial Extensions выбрать PostGIS 1.4 (рис. 3.1).

Далее необходимо выбрать зеркало загрузки в временную папку, в которую будет загружен дистрибутив, который автоматически начнет установку.

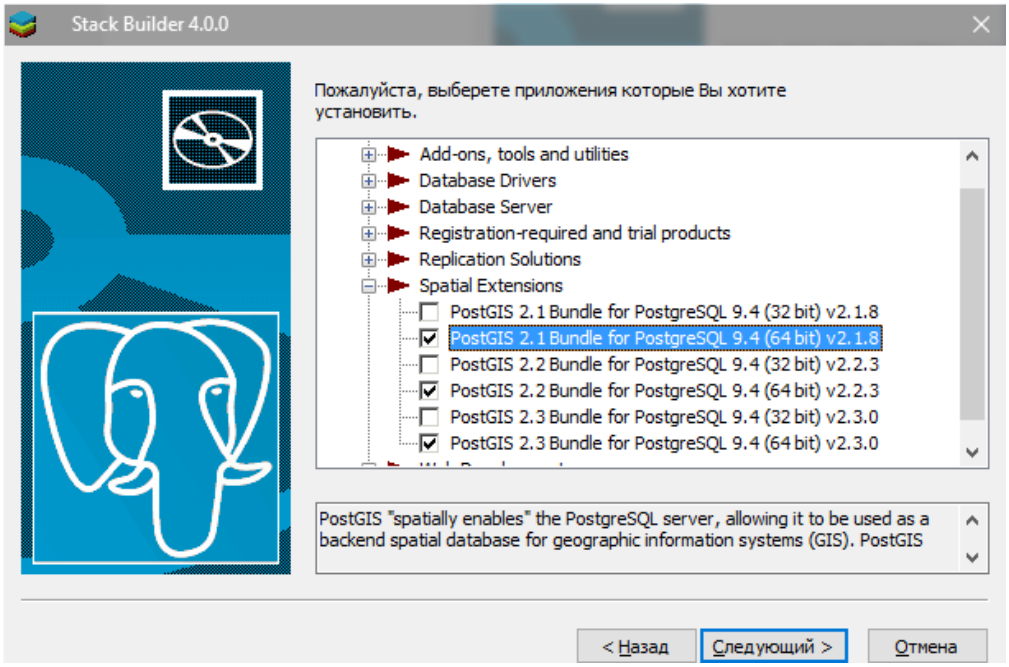


Рис. 3.1. Установка PostGIS

Необходимо снять отметку с пункта «Create spatial database» (рис. 3.2). «Create spatial database» позволяет создать пространственную базу данных автоматически. Далее мы будем рассматривать процесс создания такой базы данных вручную, не прибегая к услугам установщика PostGIS, поэтому на данном этапе мы отметку с этого пункта снимем. Затем необходимо выбрать путь установки.

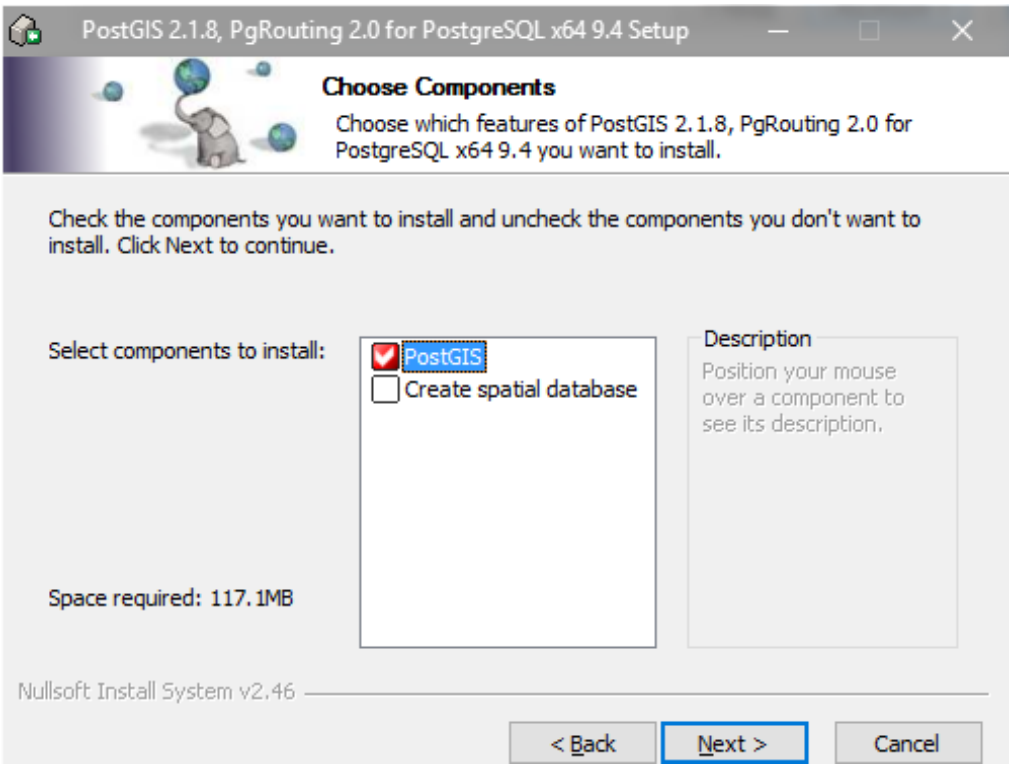


Рис. 3.2. Завершение установки PostGIS

На этом установка PostGIS завершена!

3.2.5 Работа в среде pgAdmin

Графический интерфейс pgAdmin поддерживает все возможности PostgreSQL и значительно облегчает администрирование.

Ввод инструкций SQL. Для ввода и тестирования инструкций и операторов языка plpgsql можно использовать приложение Инструмент запросов, входящее в комплект поставки PosgreSQL. Для запуска «Инструмент запросов» необходимо выбрать одноименную команду из меню «Инструменты» командного меню pgAdmin или нажать комбинацию клавиш Ctrl+E. В открывшемся окне выбрать базу данных из списка и создать новый запрос или загрузить сохраненный ранее SQL-скрипт.

Query Tool – многооконное графическое средство, которое позволяет:

– вводить (New query, Load SQL script), редактировать, выполнять (Execute query), и сохранять (Save query/result) запросы и их результаты в текстовом (text) и графическом виде (image);

– визуально строить запросы (Graphical Query Builder);

– получать пояснения, как запросы анализируются, оптимизируются и выполняются, и представлять в графическом виде схему выполнения запроса (Explain query);

– получать статистическую информацию о запросе (Message);

– сохранять результаты выполнения запроса в файле (Export).

Создание базы данных. В меню Пуск необходимо найти директория PostgreSQL 9.4 и запустить утилиту pgAdmin III.

В браузере объектов после двойного щелчка на объекте PostgreSQL 9.4 необходимо активировать пароль суперпользователя для подключения к выбранному серверу.

Затем в браузере объектов необходимо выбрать раздел PostgreSQL 9.4 → Базы данных → Новая база данных… и установить имя новой базы данных (например, mydb), владельца – postgres, шаблон – template_postgis. Используя шаблон базы данных «template_postgis», мы тем самым создаем базу данных с пространственным расширением.

Примечание: *если при установке, был оставлен флаг на опции, отвечающей за создание базы по шаблону, то необходимо в поле «шаблон» указать – «template_postgis». Который указывает на, создание базу данных с пространственным расширением по шаблону.*

Если база данных не содержит шаблон «template_postgis», то можно загрузить PostGIS вручную, вызвав два скрипта SQL, которые установят функции и типы PostGIS. Для этого необходимо запустить редактора SQL командой меню «Инструменты → Инструмент запросов» или нажать комбинацию клавиш Ctrl+E.

Выбрать меню «Файл → Открыть» и открыть файл: C:\Program Files\PostgreSQL\9.4\share\contrib\postgis.sql

Активировать опцию «Выполнить запрос» (пиктограмма с изображением зеленого треугольника). Файл postgis.sql будет исполнен – функции и объекты PostGIS будут загружены в базу данных.

Выбрать меню «Файл → Открыть» и открыть файл: C:\Program Files\PostgreSQL\9.4\share\contrib\spatial_ref_sys.sql

Активировать опцию «Выполнить запрос». Файл spatial_ref_sys.sql будет исполнен, загрузив параметры систем координат в формате EPSG в таблицу БД.

Процесс создания пространственной базы данных без использования шаблона окончен. Таким образом, на данный момент времени создана пространственная база данных «postgis», готовая к наполнению данными.

Для создания новых таблиц необходимо выбрать раздел Базы данных → mydb → Схемы → Public → Таблицы → Новая таблица …

Таблицы метаданных OGC. При создании пространственной базы данных автоматически создаются две таблицы метаданных – SPATIAL_REF_SYS и GEOMETRY_COLUMNS. Они создаются в соответствии со спецификацией «Open Geospatial Consortium Simple Features for SQL specification», выпущенной OGC и описывающей стандартные типы объектов ГИС, функции для манипуляции ими и набор таблиц метаданных.

Таблица GEOMETRY_COLUMNS хранит информацию о таблицах базы данных, содержащих пространственную информацию. Её заполнение осуществляется вручную, либо как следствие выполнения специальной процедуры OGC AddGeometryColumn().

Таблица SPATIAL_REF_SYS содержит числовые идентификаторы и текстовые описания систем координат, используемых в пространственной базе данных. Одним из полей этой таблицы является поле SRID – уникальный идентификатор, однозначно определяющий систему координат. SRID представляет из себя числовой код, которому соответствует некоторая система координат. Например, распространенный код EPSG 4326 соответствует географической системе координат WGS84.

Более подробную информацию по таблицами метаданных можно найти в руководстве по PostGIS http://gis-lab.info/docs/postgis/manual.

Для созданной базы данных «postgis» необходимо создать таблицу, содержащую пространственные данные и применить к ним функции PostGIS. Для этого в окно запросов SQL необходимо вставить нижеприведенный запрос:

```
create table points (pt geometry, name varchar);
```

и активировать опцию: «Выполнить запрос».

Тем самым будет создана таблица `points`, содержащая два поля: поле `pt` типа `geometry` и поле `name` типа `varchar`.

Команды:

```
insert into points values ( 'POINT(0 0)', 'Origin' );
insert into points values ( 'POINT(4 0)', 'X Axis' );
```

добавили (`insert`) в таблицу points три записи, содержащие информацию о точках.

Запрос:

```
select name, ST_AsText(pt), ST_Distance(pt, 'POINT(4 3)')
from points;
```

осуществляет выборку с использованием функций PostGIS: `ST_Distance()` и `ST_AsText()` (рис. 3.3).

В качестве параметра обе функции используют объект типа geometry. Функция `ST_Distance()` рассчитывает расстояние между двумя указанными точками плоскости, а `ST_AsText()` возвращает геометрию объекта в текстовом формате WKT (Well-Known Text). Формат WKT включает информацию о типе объекта и координаты, составляющие объект.

Для наполнения пространственной базы данными импортированными из shape-файла необходимо сначала преобразовать данные из shape-файла в формат понятный PostgreSQL с помощью утилиты shp2pgsql, а затем загрузить их в базу данных (см. рис. 3.3).

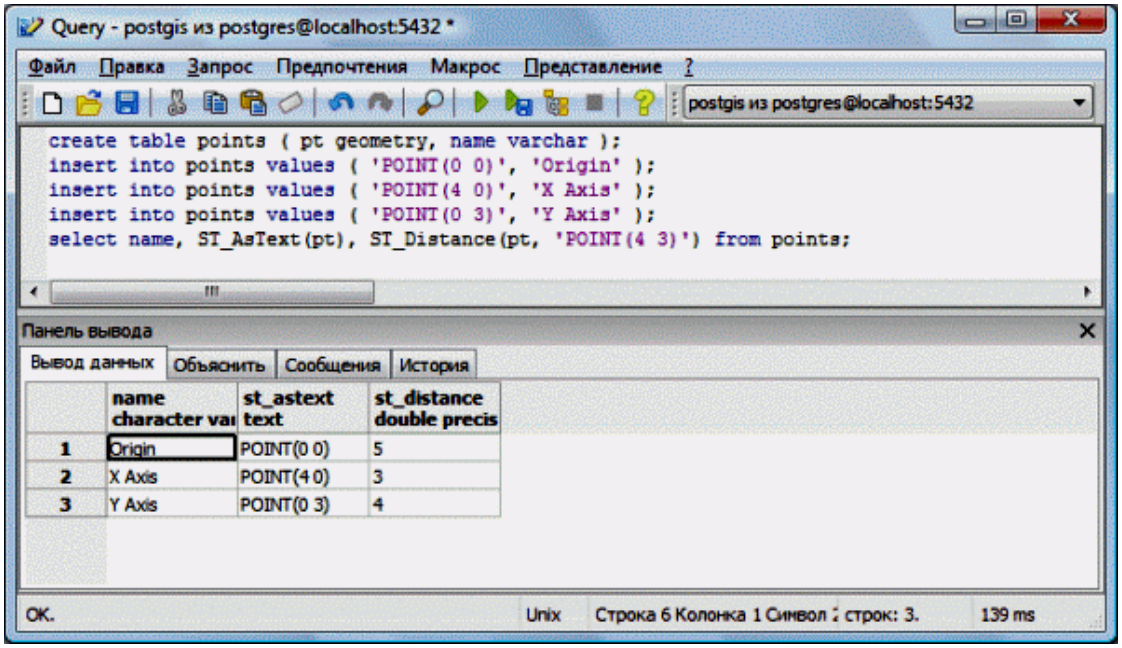


Рис. 3.3. Создание таблицы, добавление записей и выполнение запроса

Данные хранятся в географической системе координат WGS84, для которой SRID имеет значение 4326. При переводе данных из shape-файлы необходимо указать это значение в качестве параметра утилиты shp2pgsql.

Загружать в базу данных необходимо файл, содержащий границы административного деления с расширением .shp (часть набора данных).

Командный bat-файл, содержащий следующие строки (в качестве примера, здесь и далее, осуществляется загрузка слоя с данными об административных границах территории России `bnd-political-boundary-a.shp`):

```
SET PATH="C:\Program Files\PostgreSQL\X.X\bin"
shp2pgsql -i -D -s 4326 bnd-political-boundary-a.shp bnd-political-boundary-a > bnd-political-boundary-a.sql
psql -U postgres -f bnd-political-boundary-a.sql -d postgis
```

осуществляет конвертацию (shp2pgsql создает файл sql) и загрузку данных в БД (psql).

Для того чтобы добавить данные к существующей БД, можно использовать ключ -a, при этом схемы существующей БД и загружаемых данных должны совпадать.

Для того чтобы убедится, что все данные были успешно загружены необходимо ввести пароль для пользователя postgres, открыть базу данных, через pgAdmin зайти в: «Базы → postgis → схемы → public→ Таблицы», вызвать правой кнопкой мыши контекстное меню, выбрать «Обновить», после чего в списке таблиц должна появиться загруженная таблица имеющая имя `bnd-political-boundary-a` (рис. 3.4).

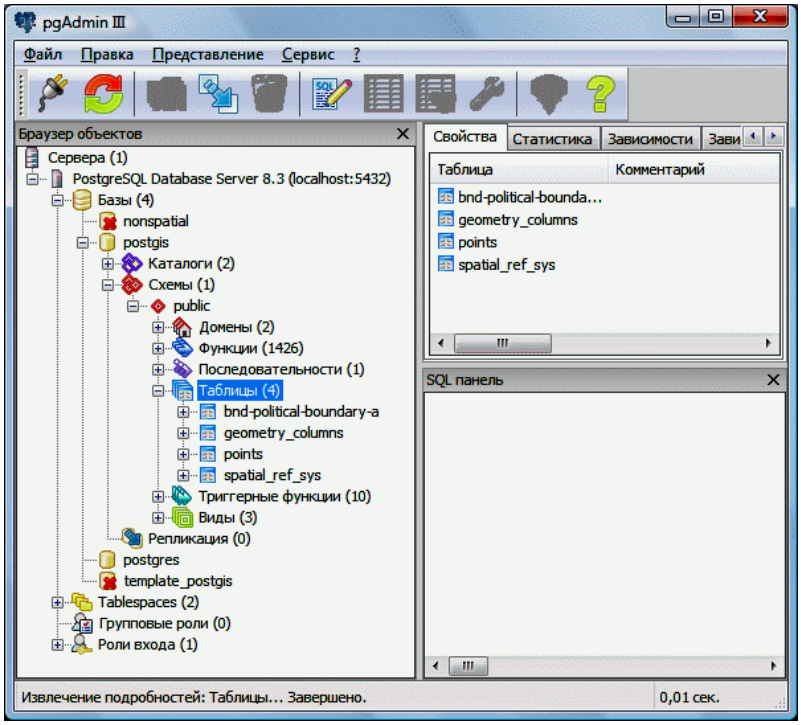


Рис. 3.4. Загрузка данных в существующую базу данных

Утилита shp2pgsql имеет множество ключей (табл. 3.2).

Таблица 3.2 – Ключи утилиты shp2pgsq

Ключ	Описание
-D	Использовать формат «database dump». По умолчанию используется формат «insert», заставляющий базу данных анализировать каждую строку загружаемого файла. Формат «database dump» не требует подобного анализа, вследствие чего загрузка файла осуществляется значительно быстрее.
-s <#>	Использовать SRID. Параметр SRID необходим при осуществлении перепроецирования внутри базы данных.
-i	Использовать 32-битные целые числа для всех значений типа integer. Необходимость этого параметра объясняется тем, что, иногда, в заголовках dbf-файлов указан размер поля гораздо больше, чем используется на самом деле.
-W <encoding>	Кодировка
-a	Использовать режим добавления. Применяется в случае добавления данных в существующую таблицу.

Ключи клиента PostgreSQL psql, которые использовались при загрузке данных, представлены в табл. 3.3.

Таблица 3.3 – Ключи клиента PostgreSQL psql

Ключ	Описание
-U <username>	Учетная запись, от имени которой идет подключение к базе данных
-f <filename>	Имя файла, который требуется выполнить.
-d <database>	Имя базы данных, в которую следует загрузить результат выполнения файла

3.3 Методические указания по построению физической модели базы данных

Физическое проектирование – создание схемы БД для конкретной СУБД. Специфика конкретной СУБД может включать в себя ограничения на именование объектов БД, ограничения на поддерживаемые типы данных и т.п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т.д.

На уровне физической модели сущности соответствует таблица в реальной СУБД, атрибуту сущности − столбец таблицы, связи − внешний ключ, первичным ключам − уникальные значения.

Если для идентификаторов объектов логической модели можно использовать национальный или латинский алфавит, то идентификаторы объектов физической модели (имена таблиц, столбцов и индексов) необходимо указывать на латинице. Для каждого атрибута необходимо указать тип данных, возможность пустых значений, значения по умолчанию и т.п. в зависимости от используемой СУБД.

Пример физической модели БД представлен на рис. 3.5.

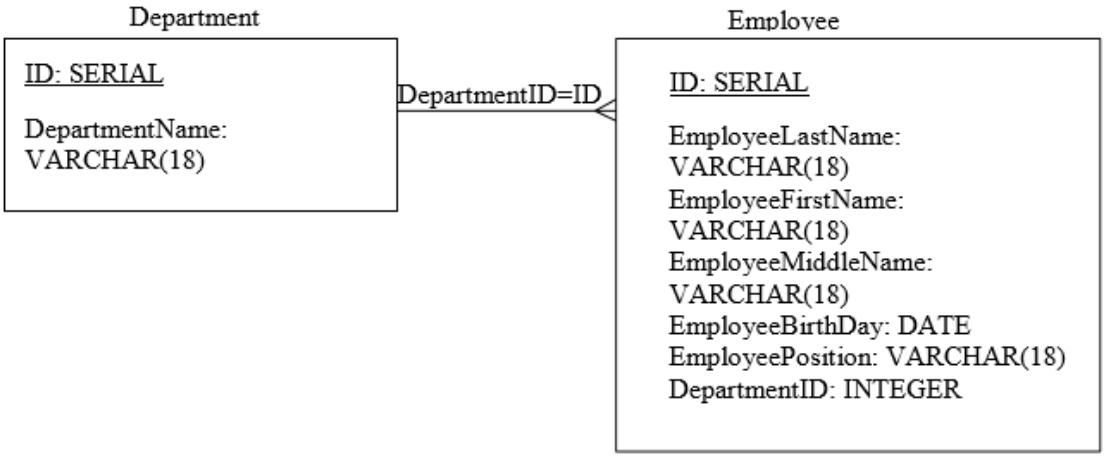


Рис. 3.5. Физическая модель БД

Схема базы анных генерируется на основе физической модели в виде SQL-скрипта (Script File − файла с расширением sql).

3.4 Программа работы

1. Преобразовать/создать разработанную в MySQL базу данных в СУБД PostgreSQL.
2. Дополнить разработанную БД пространственными данными.
3. Произвести сравнительный анализ СУБД MySQL и PostgreSQL.

3.5 Содержание отчета

Отчет по лабораторной работе должен содержать:

1) титульный лист;
2) цель работы;
3) этапы создания БД в PostgreSQL;
4) скриншот модели БД, построенной в PostgreSQL;
5) сравнительный анализ СУБД MySQL и PostgreSQL;
6) выводы по работе.

3.6 Контрольные вопросы

1. Перечислить типы данных СУБД PostgreSQL.
2. Что такое агрегатные функции?
3. Что такое ограничения целостности? Какие ограничения целостности поддерживает PostgreSQL?
4. В чем заключается концепция табличных пространств, используемая в PostgreSQL?
5. Что такое схемы в PostgreSQL? Для чего они предназначены?
6. Какие типы индексов поддерживает PostgreSQL?
7. Как выполнить пользовательскую функцию? Синтаксис команды создания функции.
8. Используются ли ограничения, наложенные на таблицу, при добавлении в нее записей через созданную функцию?
9. Можно ли использовать созданные функции за пределами базы данных, в которой они созданы? Почему?
10. Перечислить возможности PostgreSQL, в том числе связанные с реализацией ГИС.

Лабораторная работа № 4

СОЗДАНИЕ ПРОГРАММНОГО ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ. ТЕСТИРОВАНИЕ БАЗЫ ДАННЫХ

4.1 Цель работы

Углубление теоретических знаний и изучение механизмов организации взаимодействия с базами данных, получение практических навыков создания приложений для работы с реляционными базами данных.

4.2 Основные теоретические положения

4.2.1 Архитектура клиент-сервер в базах данных

В настоящее время большинство СУБД поддерживают режим работы клиент-сервер. Технология клиент-сервер обеспечивает прикладным программам – клиентам – доступ к данным, которыми управляет сервер, и позволяет нескольким клиентам работать с одним сервером.

Сервер будем рассматривать в качестве процесса, который выполняет запросы от других процессов. Сервер базы данных будем рассматривать в качестве логического процесса, отвечающего за обработку запросов к базе данных. Клиента будем рассматривать в качестве процесса, отправляющего серверу запрос на обслуживание. К функциям клиента относятся: установление связи с сервером базы данных; запрос конкретного вида обслуживания; получение результатов запроса; подтверждение окончания обслуживания.

При использовании технологии клиент-сервер клиент посылает запрос серверу, который в соответствии с запросом выбирает данные из базы данных, возможно, подвергает их предварительной обработке и отправляет результаты клиенту. Таким образом, основную работу с базой данных выполняет сервер, что позволяет уменьшить сетевой трафик.

В качестве языка, на котором формулируются запросы к базе данных, выступает язык SQL. Основной принцип технологии клиент-сервер – разделение функций стандартного интерактивного приложения на четыре группы:

1) функции ввода и отображения данных (интерфейс);
2) прикладные функции;
3) функции хранения данных и управления информационными ресурсами;
4) служебные функции.

Прикладные функции зависят от предметной области, например, для системы продажи авиабилетов такими функциями являются поиск мест на рейс, продажа и бронирование билетов и т.д.

Служебные функции играют роль связок между функциями групп 1–3. В соответствие с этими группами выделяют три логических компонента:

1) компонент представления (для ввода и отображения данных);
2) прикладной компонент (для реализации прикладных функций);
3) компонент доступа к информационным ресурсам (для управления данными).

Технологии доступа к базе данных. Доступ к базе данных осуществляется с помощью интерфейса, который реализован как приложение к базе данных. Приложение может быть написано на различных языках программирования высокого уровня и с использованием различных программных средств. Для того чтобы упростить процесс разработки приложений баз данных, были созданы различные технологии, скрывающие от разработчика специфику работы с конкретной базой данных. К таким технологиям можно отнести:

– ADO (Active Data Objects) – модель программирования, которая предназначена для создания на web-серверах динамических интерактивных web-страниц для организации подключения к базам данных. Является наиболее современной технологией разработки приложений для работы с распределенными БД по технологии клиент-сервер;

– BDE (Borland Database Engine) – интерфейс между приложением и базой данных, реализованный в рамках продуктов фирмы Borland;

– ODBC (Open Database Connectivity) – технология открытого доступа к данным, предусматривает использование единого интерфейса для доступа к базам данных, поддерживающим язык SQL;

– OLE DB (Object Linking and Embedding Data Base) – технология, предоставляющее решение обеспечения COM-приложениям доступ данным независимо от типа источника данных;

– JDBC (Java Data Base Connectivity) – мобильный интерфейс к базам данных на платформе Java. Это интерфейс прикладного программирования для выполнения SQL-запросов к базам данных из программ, написанных на платформенно-независимом языке Java, позволяющем создавать как самостоятельные приложения, так и аплеты, встраиваемые в web-страницы.

Модель ADO базируется на технологии OLE DB (Object Linking and Embedding Databases – связывание и внедрение объектов баз данных). OLE является объектно-ориентированной технологией разработки повторно используемых программных компонентов. Она позволяет приложениям совместно использовать объекты, которые обладают специальными функциями. В качестве источника данных OLE-приложений могут выступать таблицы баз данных, представления, а также текстовые файлы, электронные таблицы, диаграммы и другие графические изображения.

Механизм BDE действует как интерфейс между приложением и базой данных, обеспечивая работу компонентов доступа. Он реализован как набор системных файлов DLL (Dynamic Link Library). Через BDE из приложений можно непосредственно обращаться к базам данных фирмы Borland (Paradox, dBASE), а обращение к другим базам данных BDE перенаправляет менеджеру драйверов ODBC или SQL-серверам. Технология BDE применяется в таких широко распростран енных продуктах, как C++ Builder, Borland C++, Delphi, IntraBuilder и JBuilder. Для доступа к базе данных через BDE достаточно знать алиас базы данных (т.е. имя, по которому к ней обращаются).

В стандарте ODBC язык SQL рассматривается как базовое средство доступа к данным. Этот интерфейс встраивается непосредственно в язык Си и обеспечивает высокий уровень универсальности. В результате одно и то же приложение может получить доступ к базам данных разных СУБД без внесения изменений в текст программы. Для связи приложения с любой выбранной пользователем СУБД достаточно иметь соответствующий ODBC-драйвер.

В интерфейс ODBC включены следующие элементы:

– библиотека функций, вызов которых позволяет приложению подключаться к базе данных, выполнять SQL-операторы и извлекать информацию из результирующих наборов данных;

– стандартный метод подключения и регистрации СУБД;

– стандартное представление для различных типов данных;

– стандартный набор кодов ошибок;

– типовой синтаксис SQL-операторов, построенный на использовании спецификаций стандартов X/Open и ISO CLI.

Архитектура ODBC включает четыре элемента:

1) приложение: выполняет вызов функций библиотеки ODBC для оправки SQL-операторов в СУБД и обработку возвращаемых СУБД данных;

2) менеджер драйверов: выполняет загрузку драйверов по требованию приложений. Менеджер драйверов представляет собой библиотеку DLL;

3) драйверы баз данных: обрабатывают вызовы функций ODBC и направляют SQL-запросы в конкретные источники данных, а также возвращают полученные от источников данные приложению. При необходимости драйверы выполняют модификацию запросов с целью приведения их в соответствие с синтаксическими требованиями конкретной СУБД. Драйверы предоставляют только те возможности, которые реализованы в целевой СУБД.

4) источники данных: являются базами данных, электронными таблицами и т.п. Данные в базе данных контролируются СУБД и операционной системой.

Объекты связи. Объект связи – объект языка программирования, осуществляющий связь между файлом данных и интерфейсом информационной системы.

Объекты связи – это объекты проекта, осуществляющие обмен информацией между интерфейсом БД и файлом данных.

Объекты связи всегда находятся на клиентской машине. Они осуществляют доступ к файлам данных, передавая информацию в интерфейс БД, и содержат внутри себя запросы, выполнения на стороне клиента.

Объекты связи также могут ограничивать доступ к информации и осуществлять защиту информации, хотя для защиты информации и ограничения доступа лучше использовать сам сервер.

Существует три технологии используемых в объектах связи:

– технология ADO;

– технология RDC;

– технология ADO.Net.

ADO является более старой технологией. Ее суть заключается в следующем: подключение к конкретной таблице или запросу, осуществляется через отдельный объект связи, т.е. все настройки и средства для работы с данными хранятся внутри конкретного объекта связи и были заложены туда при его проектировании.

Согласно технологии RDC, файлы данных рассматриваются в качестве устройств, т.е. для работ с БД нам необходим драйвер. Объект связи, работающий по технологии RDC, при работе с файлом данных сначала обращается к драйверу БД, который в свою очередь обращается к файлу данных.

Технология ADO.Net является смесью технологий ADO и RDC. Объекты связи работающие по этой технологии работают аналогично объектам, работающим по технологии ADO, однако, объекты связи входят в состав пакета Microsoft Net Framework, и автоматически обновляются вместе с этим пакетом.

Таблица 4.1 – Плюсы и минусы технологий, используемых в объектах связи

ADO	RDC	ADO.Net
+ независимость от драйверов БД, установленных в операционной системе	+ возможность работать с современными БД	+ возможность работать с современными БД
+ простое программирование	+ возможность добавлять новые виды БД	+ возможность добавлять новые виды БД
– невозможность работать с новыми типами БД	– зависимость от драйверов, установленных в системе	– зависимость от пакета Microsoft Net Framework
– невозможность обновлять список поддерживаемых БД	– более сложное рограмммирование	– более сложное рограмммирование

Динамические запросы и запросы, выполненные на стороне сервера можно создавать только в технологиях RDC и ADO.Net.

4.2.2 Интерфейс информационных систем

В системах, построенных по технологии клиент-сервер, существует два вида интерфейса:

– интерфейс, реализуемый при помощи клиентского приложения;

– web-интерфейс.

Интерфейс, реализуемый при помощи клиентского приложения – это компьютерная программа, устанавливаемая на клиентские компьютеры, предназначенная для работы с файлами данных через сеть. Основными элементами клиентских приложений являются формы (окно программы) и отчеты.

Элементы управления на форме называется объектами. Каждый объект обладает своим набором свойств, событий и методов.

В БД все объекты форм делятся на два класса:

– объекты управления – объекты, осуществляющие управление БД (например, кнопка или выпадающий список);

– объекты для отображения информации – элементы, отображающие содержимое таблиц, запросов или фильтров, позволяющие добавлять изменять и удалять информацию, и проводить ее анализ.

Все формы в клиентском приложении делятся на три группы:

1) формы для работы с данными – формы, содержащие как объекты управления, так и объекты просмотра данных. Такие формы предназначены для отображения, изменения, удаления и анализа данных;

2) кнопочные формы – формы, содержащие только объекты управления, предназначаются для открытия всех других форм;

3) информационные и служебные формы – формы, содержащие только элементы управления, предназначены для отображения служебной информации (справки), несвязанной с таблицами, запросами и фильтрами, либо для выполнения служебных операций, не связанных с данными (например, форма с калькулятором)

Существует два вида дизайна форм:

– ленточные формы – формы, выводящие информацию по одной записи;

– табличные формы – формы, выводящие информацию в виде таблицы.

Отчет – это объект базы данных, который используется для вывода на экран, в печать или файл структурированной информации. Отчеты позволяют извлечь из таблиц или запросов базы данных необходимую информацию и представить ее в виде удобном для восприятия. Отчеты содержит заголовок, область данных, верхний и нижний колонтитулы, примечание и разбит на страницы.

Основой web-интерфейса являются страницы (файл с расширенным htm или html). Работа со страницами осуществляется с помощью программы – браузера. Изначально страницы находятся на сервере, пользователь сначала загружает их на свой компьютер с сервера, а затем с помощью страниц пользователь работает с файлом данных.

Объекты связи используются только в клиентском интерфейсе. В web-интерфейса функции объекта связи выполняет сервер. В web-интерфейсе отсутствуют отчеты, их роль выполняют сами страницы.

4.3 Программа работы

1. Разработать приложение для организации доступа к данным, хранящимся в БД, которая была разработана в лабораторных работах № 1, 2. БД необходимо создать под управлением выбранной Вами открытой СУБД. Приложение должно содержать кнопки и выпадающие списки. Каждая кнопка или каждый из элементов выпадающего списка обеспечивает просмотр одной формы с данными или выполнение одного запроса. Приложение должно обеспечивать просмотр всех данных БД и выполнение всех необходимых запросов, удовлетворяющих требованиям технического задания (соответствующих описанию предметной области, реализованному в лабораторной работе № 1).

2. Разрабатываемое программное приложение должно:

– заносить информацию в созданную базу данных;

– выполнять необходимые действия по модификации и удалению информации в базе данных, причем все операции по занесению, модификации и удалению данных должны выполняться в терминах предметной области, а не базы данных;

– поддерживать целостность базы данных, не допуская появления некорректных данных;

– выполнять все действия над базой данных в рамках транзакций;

– содержать достаточное количество данных, позволяющих показать результаты выполнения запросов;

– контролировать все вводимые данные.

3. Отразить в отчете к лабораторной работе:

– постановку задачи, решаемой разработанным программным приложением;

– описать технологии, используемые для доступа к базе данных;

– обосновать необходимость использования выбранной СУБД и языка программирования для реализации поставленных задач;

– указать последовательность действий (использованные команды, подключаемые библиотеки) для создания в приложении связи с внешней базой данных;

– указать команды, используемые для создания и завершения сеанса между клиентом и сервером для передачи запросов в СУБД;

– указать команды, используемые для выполнения SQL-запросов и обработка их результатов;

– привести руководство пользователя разработанным программным приложением, содержащее описание интерфейсов всех функций приложения.

4.4 Содержание отчета

Отчет по лабораторной работе должен содержать:

1) титульный лист;
2) цель работы;
3) скриншоты основных функций разработанного приложения;
4) выводы по работе.

4.5 Контрольные вопросы

1. Пояснить смысл программной навигации по записям таблицы.
2. Привести примеры методов для работы с записями таблицы (пример использования).
3. Привести пример доступа к значениям полей данных в программе.
4. Как происходит осуществления ввода данных пользователем с помощью компоненты Edit (пример)?

5. Что такое транзакция к БД? Каковы функции и задачи транзакции?

6. Перечислить функции преобразования типов данных.

7. Как осуществляется вывод рассчитанных значений на форму приложения?

8. Может ли сервер рассматриваться в качестве процесса, который выполняет запросы от других процессов?

9. Перечислить функции интерфейса БД (учесть манипуляционную и административную, пояснить их смысл).

10. Есть ли в web-интерфейсе отчеты?

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация сопровождение. – М.: Издательский дом «Вильямс», 2003. – 1114 с.

2. Дейт К.Дж. Введение в системы баз данных. – 8-е изд. – М.: «Вильямс», 2006. – С. 1328.

3. Карпова Т.С. Базы данных: модели, разработка, реализация. – СПб.: Питер, 2001. – 304 с.

4. Советов Б. Я. Базы данных: теория и практика: учебник для бакалавров / Б.Я. Советов, В.В. Цехановский. – 2-е изд. – М.: Издательство Юрайт, 2013. – 463 с.

5. Кренке Д. Теория и практика построения баз данных. – СПб.: Питер, 2003. – 800 с.

6. Ульман Д. Основы реляционных баз данных. – М.: Вильямс, 2006. – 382 с.

7. Кузнецов М. MySQL 5 в подлиннике. – СПб.: БХВ-Петербург, 2010. – 1007 с.

8. Уорсли Д. PostgreSQL. – СПб.: Питер, 2003. – 496 с.

9. Моор П. К., Моор А. П. Базы данных: учеб. пособие. - Тюмень: Изд-во ТюмГУ, 2010. – 288 с.

Изд. № 121/18. Объем 3,75 п.л.
РИИЦМ ФГАОУ ВО «Севастопольский государственный университет»