Министерство образования и науки РФ

Севастопольский государственный университет

Организации архитектуры «клиент-сервер» в системах баз данных. Построение полной атрибутивной модель базы данных в нотации IDEF1X. Создание программного приложения для работы с базой данных.

Методические указания
к выполнению лабораторной работы №4
по дисциплине **«Управление данными»**

Для студентов, обучающихся по направлению 09.03.02
«Информационные системы и технологии»
по учебному плану подготовки бакалавров
дневной и заочной форм обучения

Севастополь
2018

УДК 004.65 (075.8)

Организации архитектуры «клиент-сервер» в системах баз данных. Построение полной атрибутивной модель базы данных в нотации IDEF1X. Создание программного приложения для работы с базой данных. Методические указания к лабораторным занятиям по дисциплине «Управление данными» / Сост. Ю.В. Доронина, А.В. Волкова – Севастополь: Изд-во СевГУ, 2018 – 27 с.

Методические указания предназначены для проведения лабораторных работ по дисциплине «Управление данными». Целью методических указаний является помощь студентам в построении полной атрибутивной модель базы данных в нотации IDEF1X, в получении навыков по настройке удаленного подключения к серверу баз данных, а также в разработке интерфейса к разработанной базе данных с использованием различного программного обеспечения и изучении механизмов организации взаимодействия с базами данных. Излагаются теоретические и практические сведения необходимые для выполнения лабораторной работы, требования к содержанию отчета.

Методические указания рассмотрены и утверждены на методическом семинаре и заседании кафедры информационных систем.

Допущено учебно-методическим центром СевГУ в качестве методических указаний.

1 ЦЕЛЬ РАБОТЫ

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. Изучение механизмов организации взаимодействия с базами данных и получение практических навыков создания приложений для работы с реляционными базами данных.

2 АРХИТЕКТУРА КЛИЕНТ-СЕРВЕР ДЛЯ БАЗ 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).

2.1 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.

2.2 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.

Таблица1 – 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 программирование	– более сложное программирование

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

2.3 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.

3 СОЗДАНИЕ И НАСТРОЙКА 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.

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.

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.

3.1 Установка VirtualBox

Прежде всего, на официальном сайте https://www.virtualbox.org необходимо загрузить установщик VirtualBox, активировав опцию «Download VirtualBox 5.1» (см. рисунок 1).

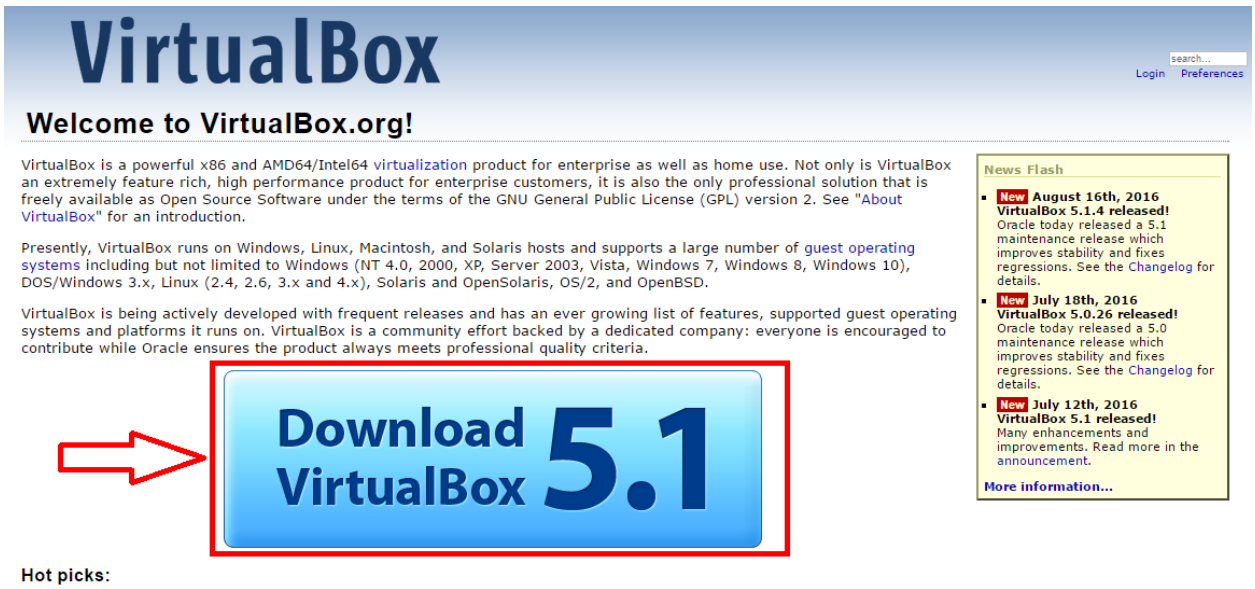


Рисунок 1 – Официальная страница для загрузки установщика VirtualBox

В следующем окне необходимо выбрать соответствующую операционную систему для VirtualBox. В представленном ниже примере выбран установщик для Windows «VirtualBox 5.1 for Windows hosts» (см. рисунок 2).

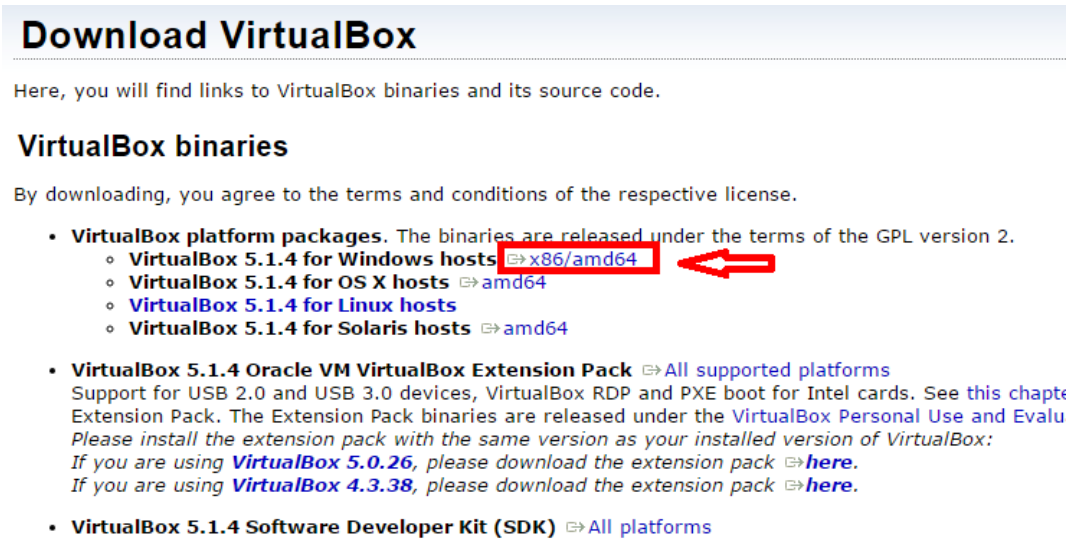


Рисунок 2 – Выбор операционной системы для VirtualBox

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.

3.2 Создание виртуальной машины

Для создания виртуальной машины в VirtualBox необходимо активировать опцию «Создать» (см. рисунок 3).

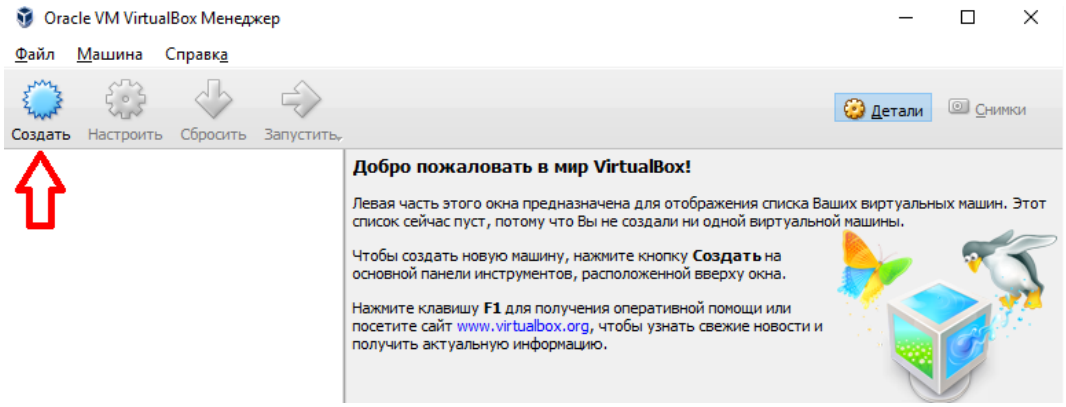


Рисунок 3 – Создание новой виртуальной машины

В открывшемся окне в представленных полях необходимо вписать следующие данные (см. рисунок 4):

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

Затем необходимо активировать опцию «Next».

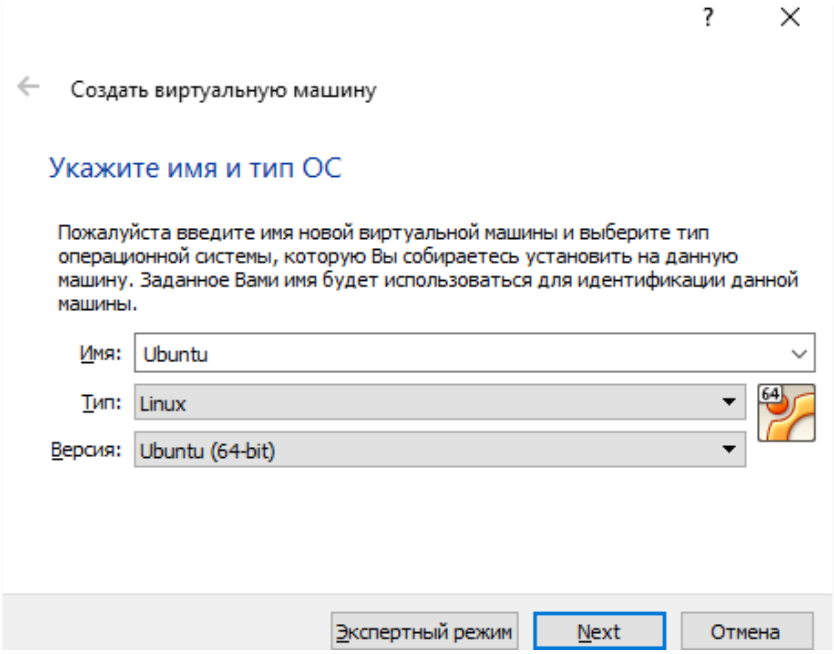


Рисунок 4 – Выбор типа виртальнй машины

В следующем окне необходимо установить выделяемый объем оперативной памяти для виртуальной машины. В представленном ниже примере (см. рисунок 5). Активировать опцию «Next».

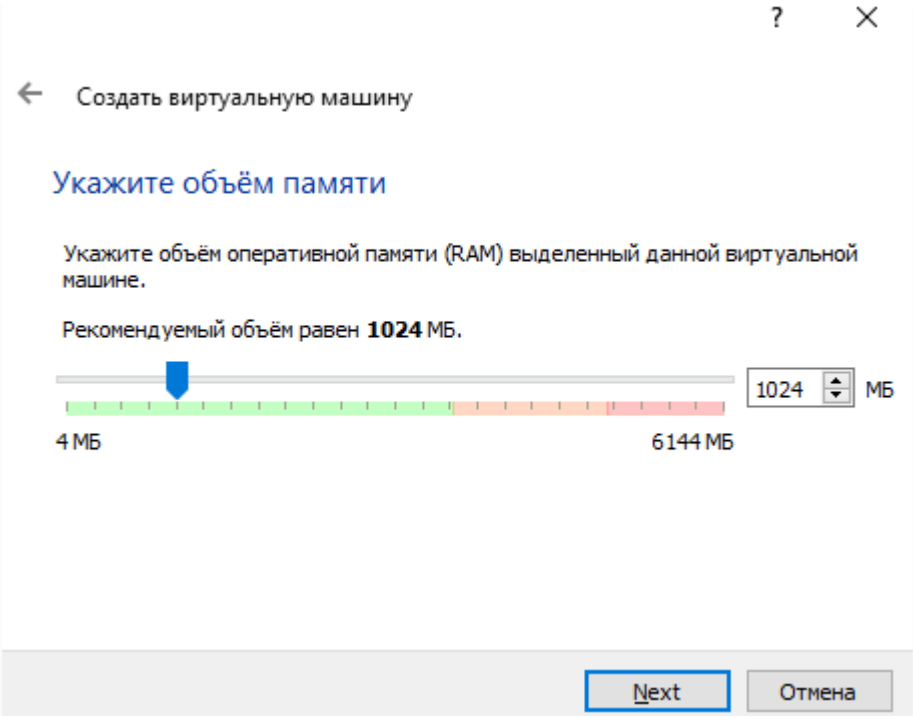


Рисунок 5 – Установка объема оперативной памяти для виртуальной машины

В следующем окне из предложенного списка необходимо выбрать вариант «Создать новый виртуальный жесткий диск» и активировать опцию «Создать» (см. рисунок 6).

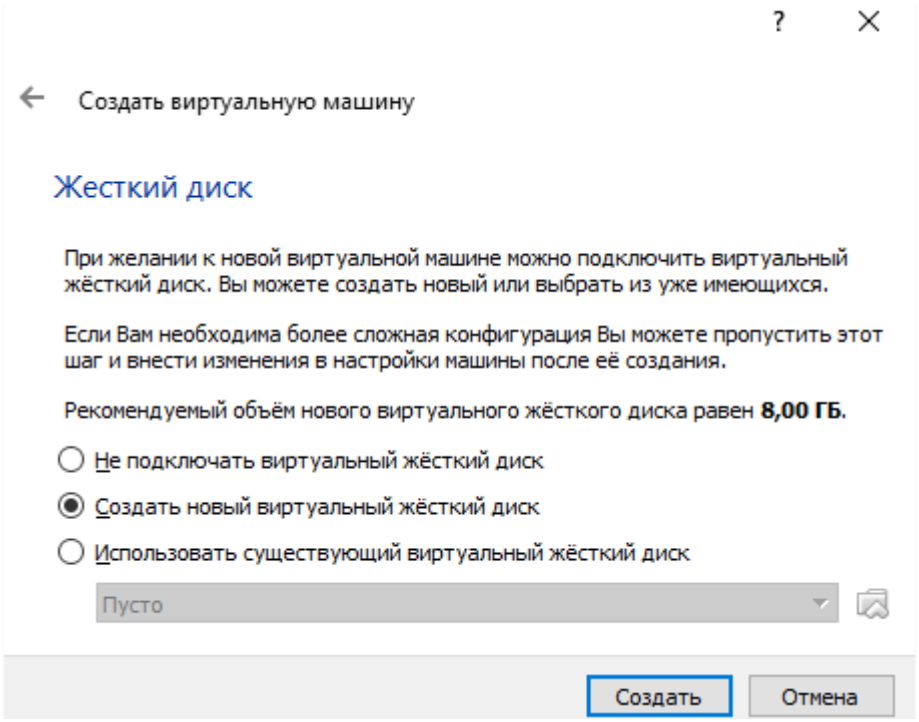


Рисунок 6 – Создание нового виртуального диска

В следующем окне необходимо указать тип файла, определяющего формат жесткого диска – VDI, которые выбран по умолчанию и активировать опцию «Next» (см. рисунок 7).

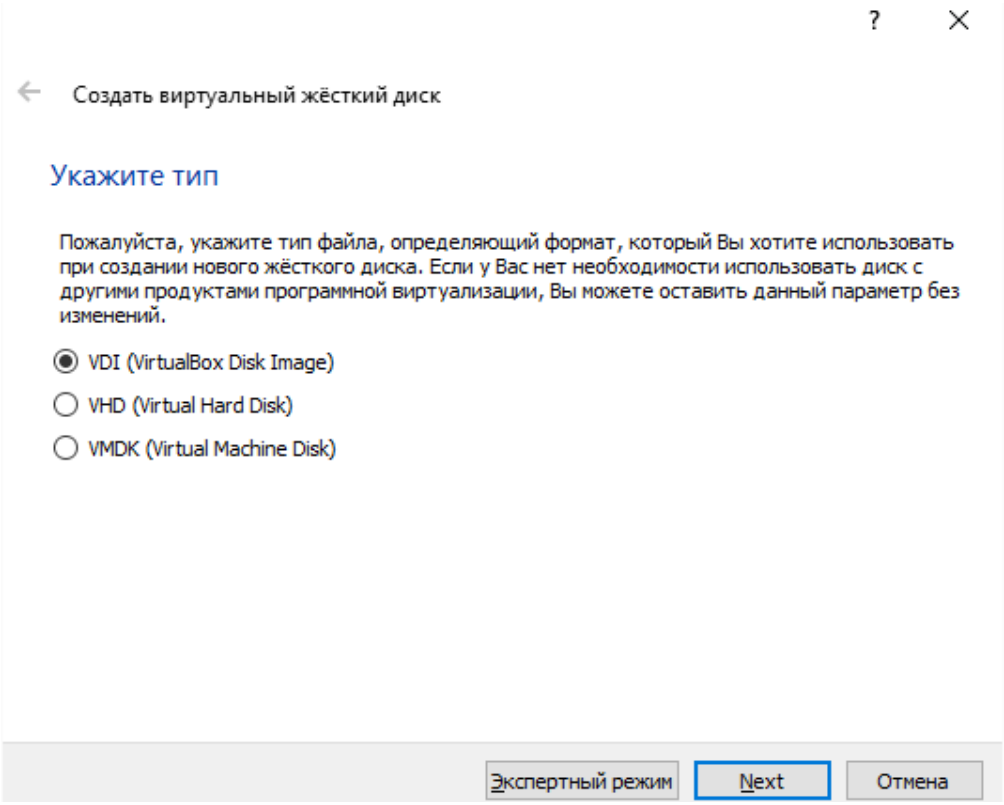


Рисунок 7 – Выбор типа файла, определяющего формат жесткого диска

В следующем окне необходимо выбрать вариант хранения данных «Динамический виртуальный жесткий диск» и активировать опцию «Next» (см. рисунок 8).

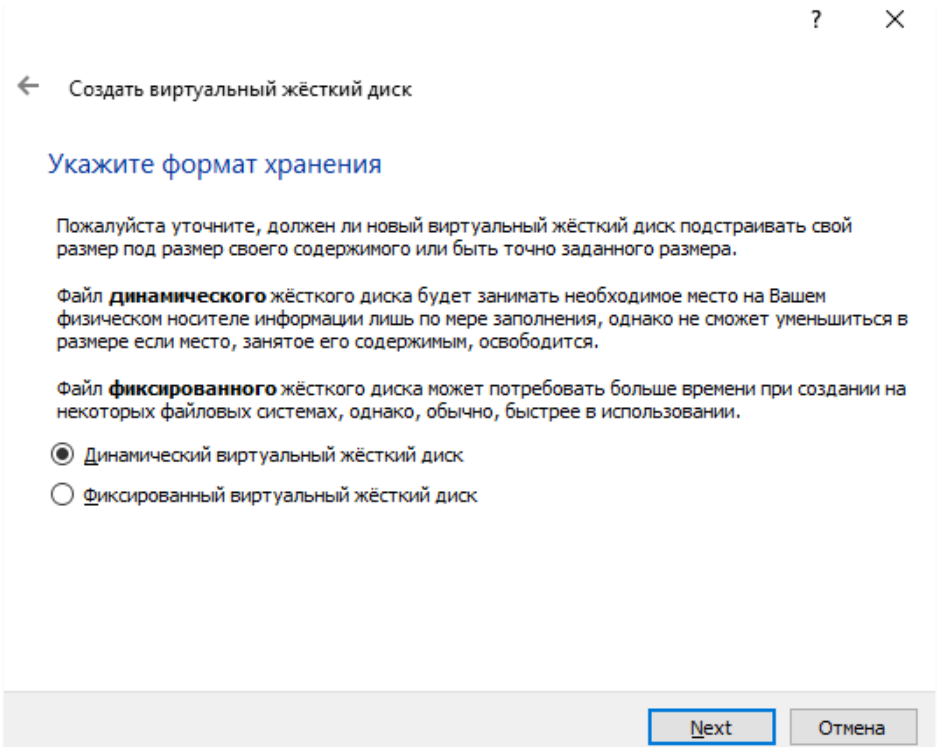


Рисунок 8 – Выбор формата хранения данных
на виртуальном жестком диске

В следующем окне необходимо указать объем виртуального жесткого диска. Система рекомендует 8ГБ, можно оставить без изменений и активировать опцию «Создать» (см. рисунок 9).

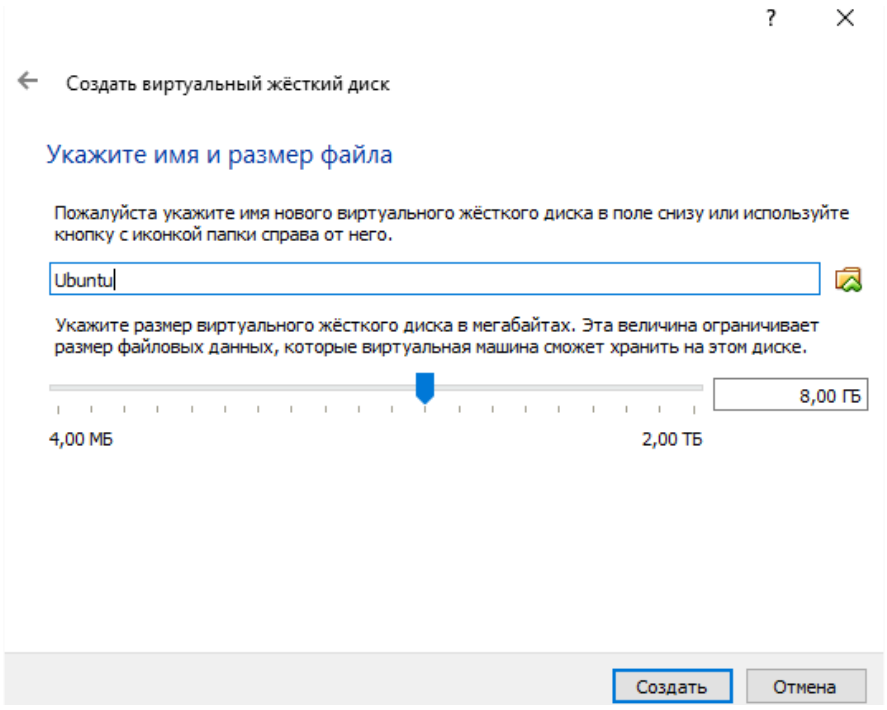


Рисунок 9 – Установка объем виртуального жесткого диска

Теперь виртуальная машина создана и ее можно увидеть в списке виртуальных машин (см. рисунок 10).

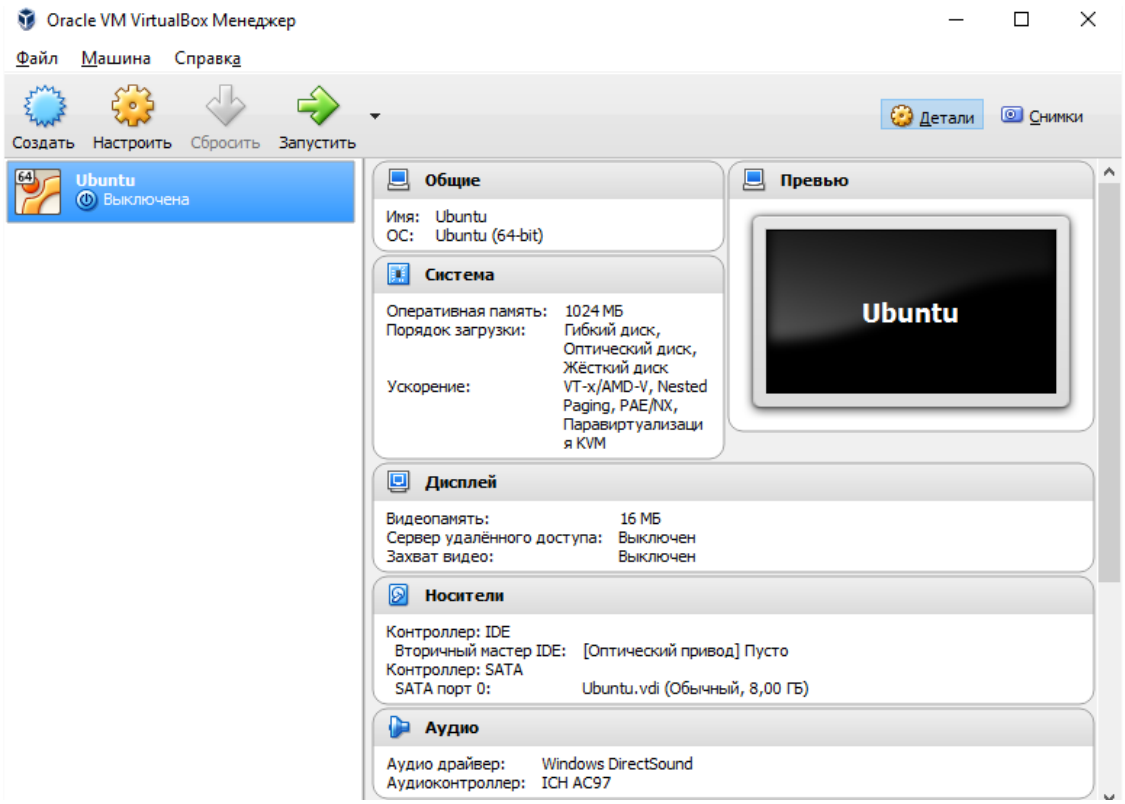


Рисунок 10 – Список виртуальных машин, созданных в VirtualBox

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

3.3 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» (см. рисунок 11).

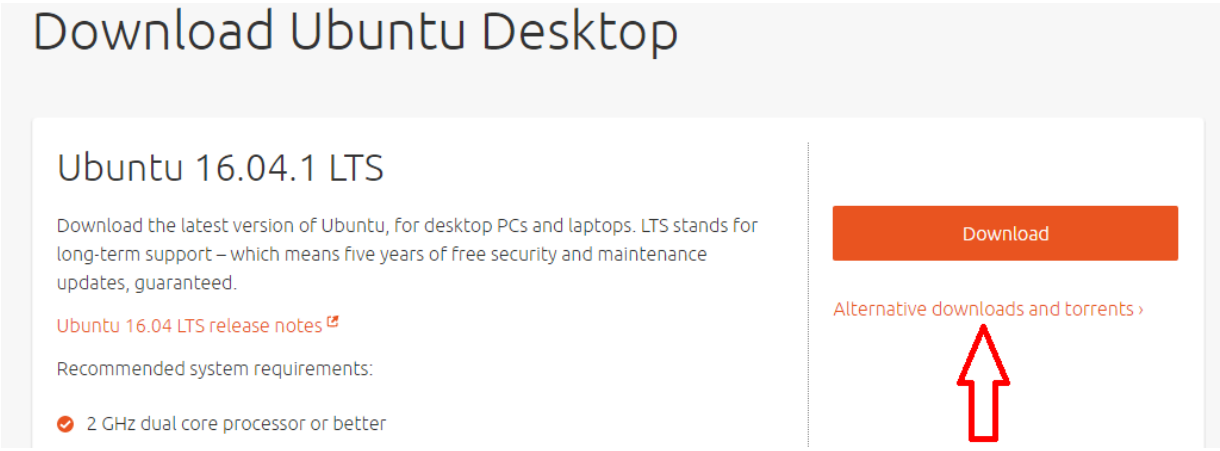


Рисунок 11 – Страница официального сайта для загрузки установщика Ubuntu

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 (см. рисунок 12).

Рисунок 12 – 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 (см. рисунок 13) и запустить виртуальную машину, активировав опцию «Запустить».

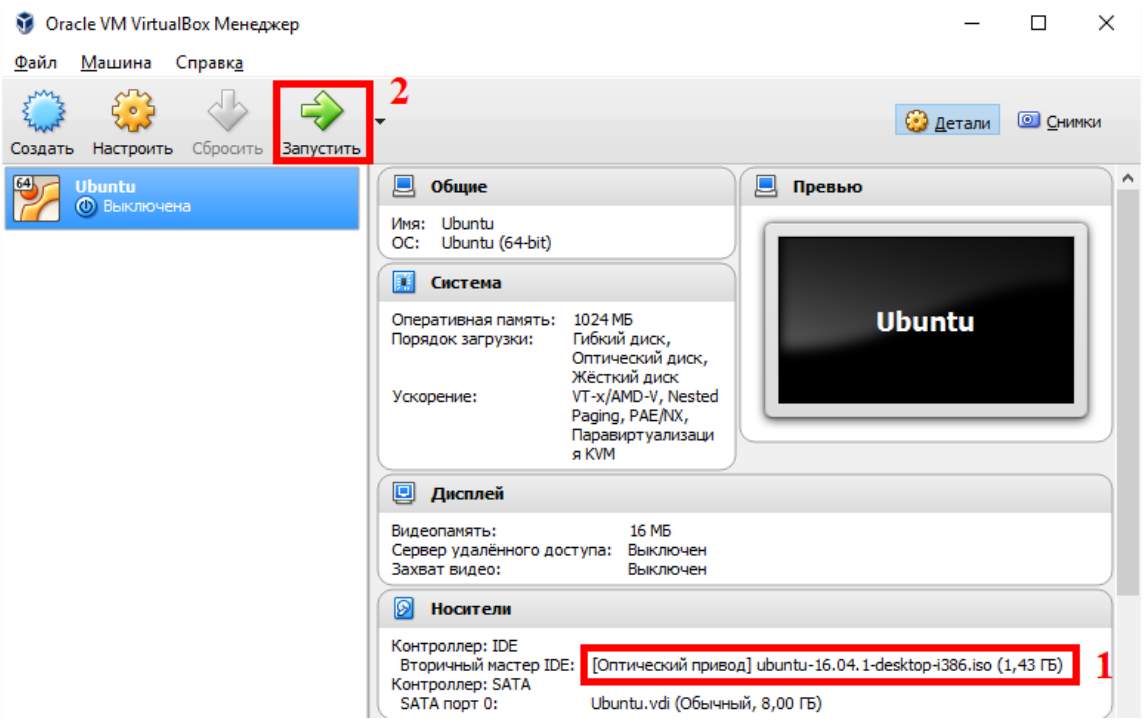
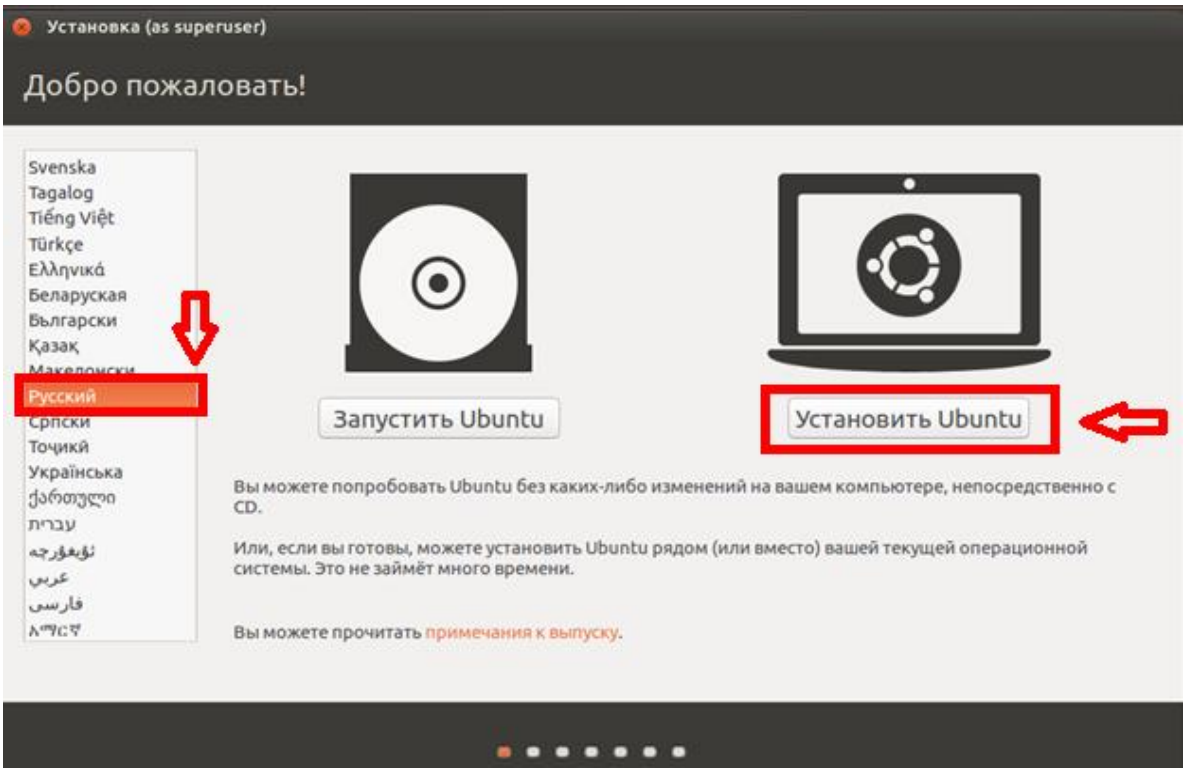


Рисунок 13 – Выбор образа для установки операционной системы

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

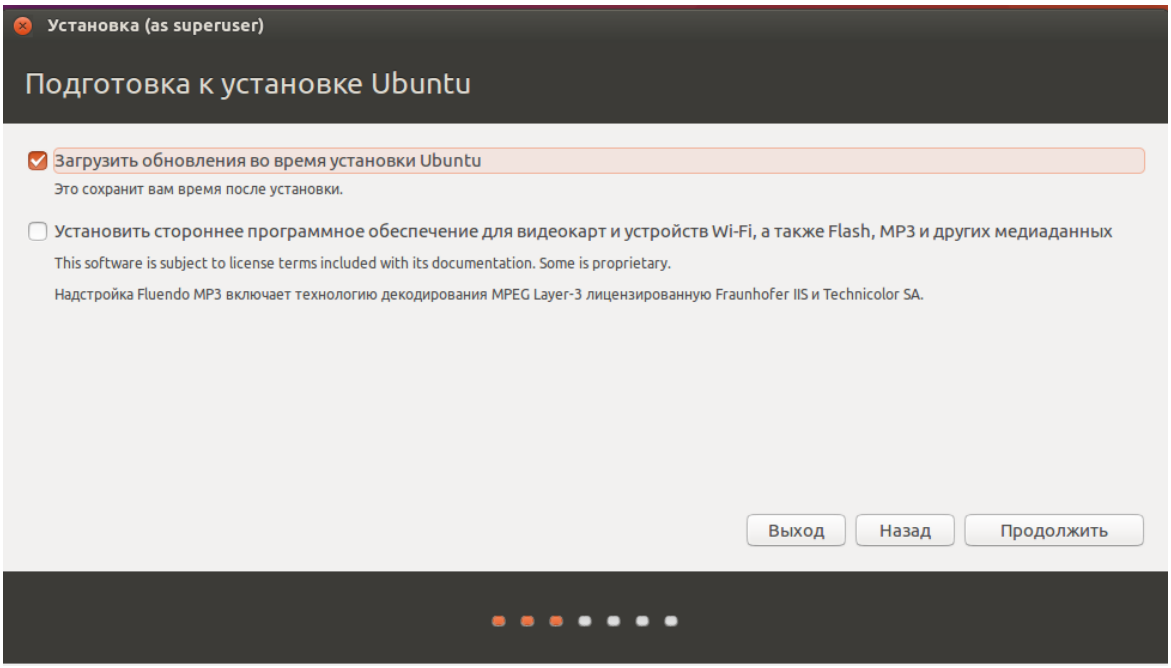
3.3.1 Шаг 1

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



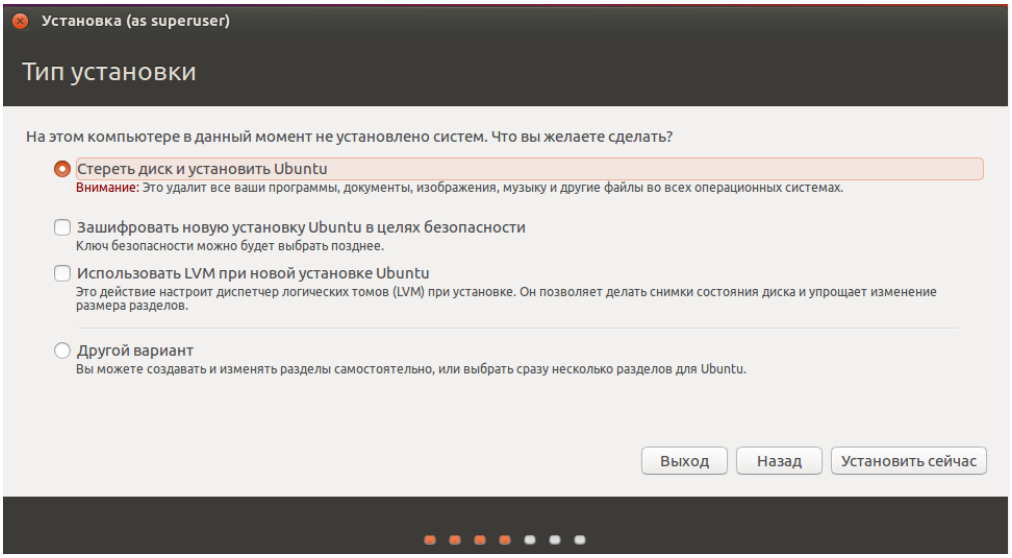
3.3.2 Шаг 2

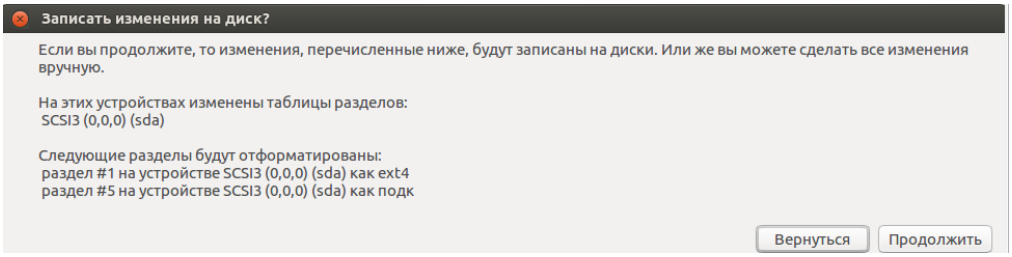
Необходимо активировать опцию «Загрузить обновления во время установки Ubuntu», но для того, чтобы обновления загрузились необходимо установить соединение с Интернет и активировать опцию «Продолжить».



3.3.3 Шаг 3

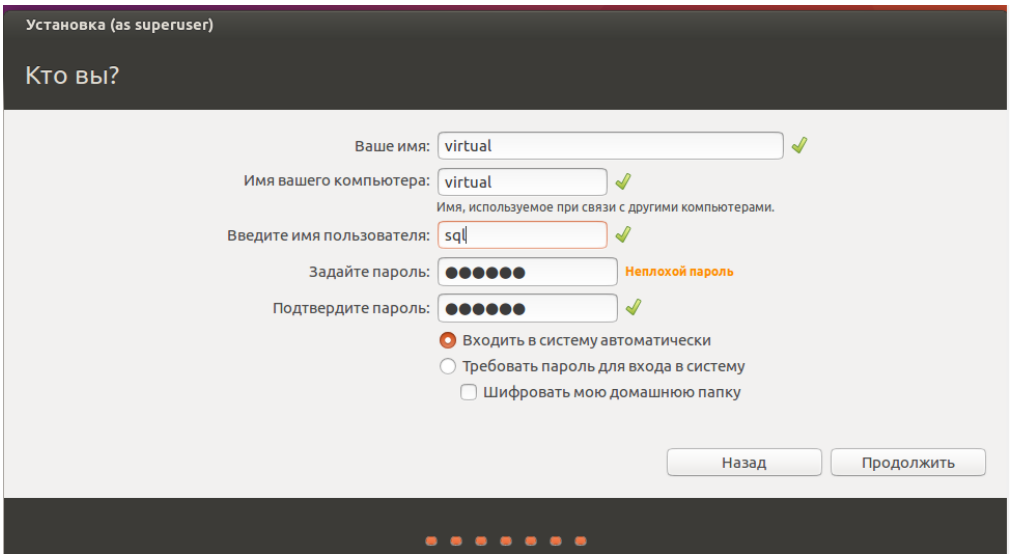
На следующем шаге установки Ubuntu необходимо выбрать пункт «Стереть диск и установить Ubuntu», активировать опцию «Установить сейчас» и принять появившееся предупреждение.



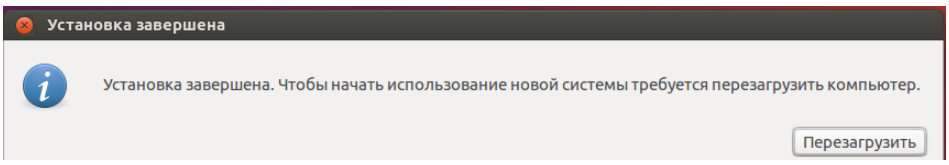


3.3.4 Шаг 4

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.



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.

3.4 Установка MySQL и MySQL Workbench на 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».

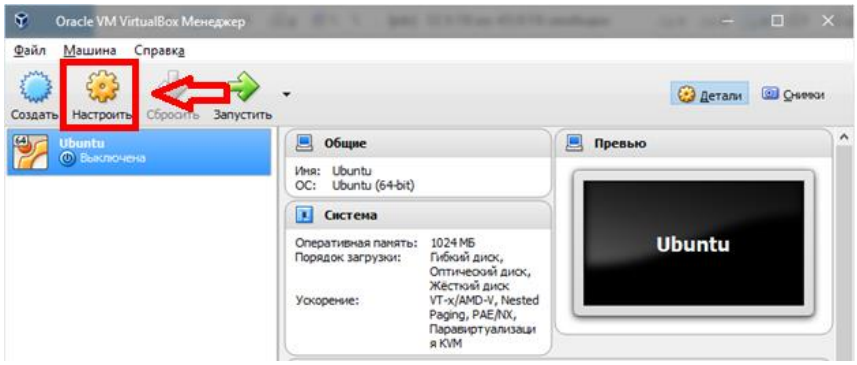
4 СОЗДАНИЕ И НАСТРОЙКА СЕТИ МЕЖДУ РЕАЛЬНОЙ И ВИРТУАЛЬНОЙ МАШИНАМИ

4.1 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».

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.

4.1.1 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» (см. рисунок 14).

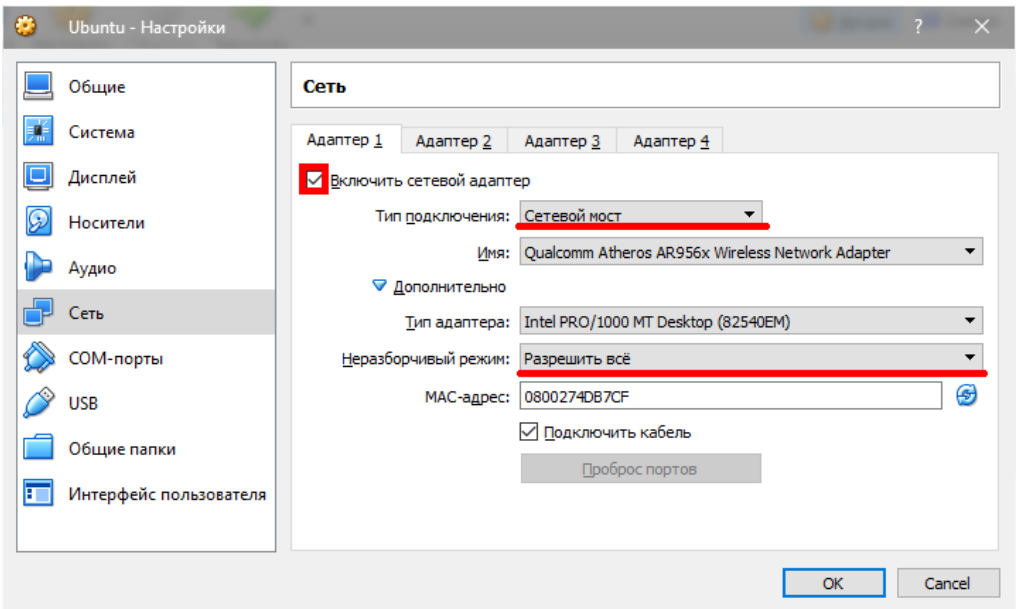


Рисунок 14 – Настройка сетевого подключения между между реальной и виртуальной машинами

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 физическим адаптером минуя хост-машину.

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

4.1.2 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» (см. рисунок 15).
В этом случае используется специальное устройство – VirtualBox Host-Only Ethernet Adapter, которое создает подсети и назначает IP-адреса гостевым операционным системам.

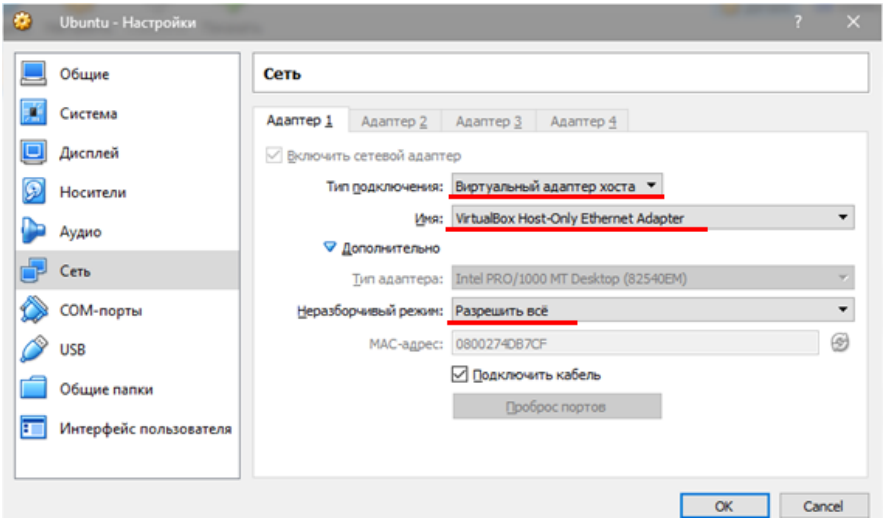


Рисунок 15 – Настройка сетевого подключения между между реальной и виртуальной машинами

4.2 Установка статического IP-адреса

По умолчанию виртуальной машине автоматически присваивается динамический IP-адрес, который при каждом новом запуске может быть разным. Поэтому, для того, чтобы обеспечить постоянное стабильное подключение 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» (см. рисунок 16).

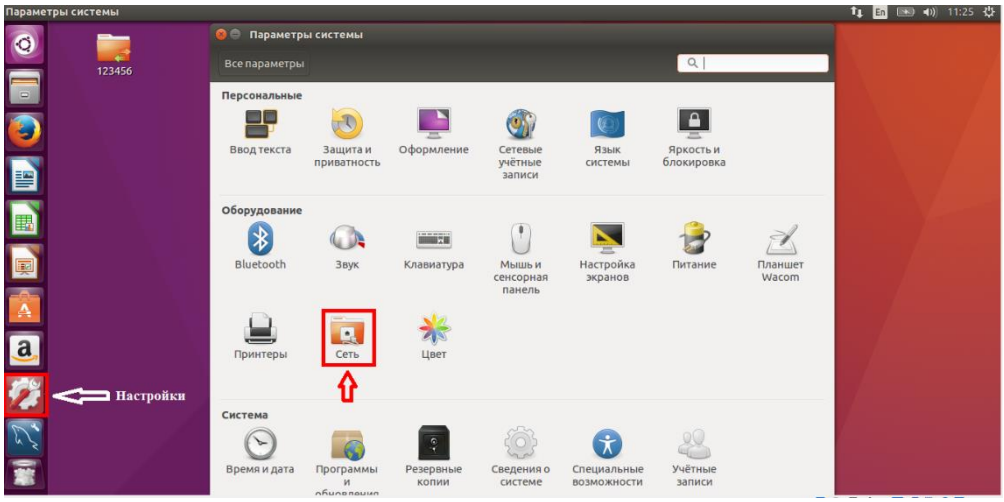


Рисунок 16 – Вход в меню сетевых настроек виртуальной машины

В появившемся окне необходим запомнить текущий IP-адреса виртуальной машины (на рисунке 17 это адрес IPv4 – 192.168.0.105) и шлюз по умолчанию, если указан (на рисунке 17 это маршрут по умолчанию – 192.168.0.1). Затем необходимо выбрать тип соединения – «Проводное» и активировать опцию «Параметры…» (см. рисунок 17).

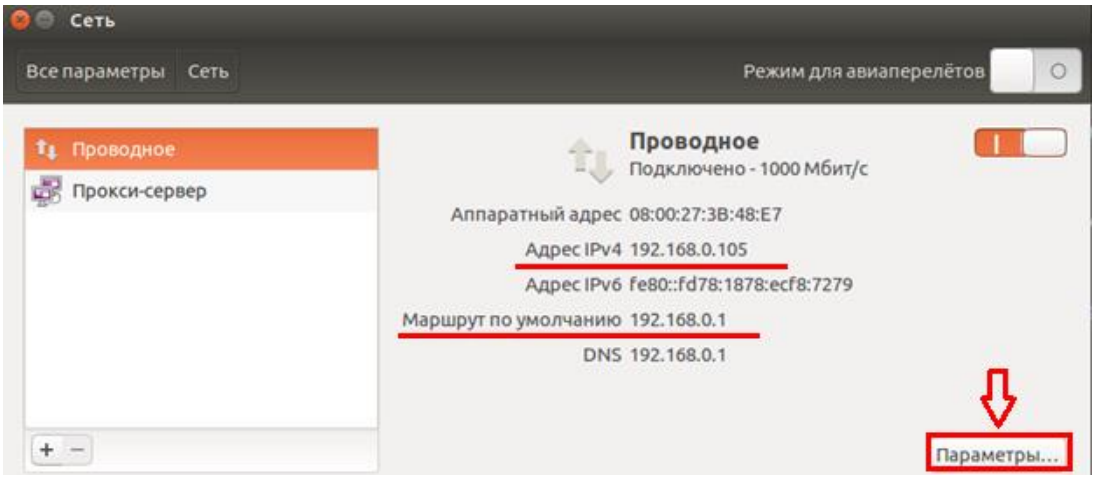


Рисунок 17 – Меню сетевых настроек виртуальной машины

Затем, в окне изменения настроек проводного соединения необходимо выбрать раздел «Параметры IPv4» и в меню «Способ настройки» активировать опцию «Вручную» вместо предложенной «Автоматически (DHCP)» (см. рисунок 18).

Затем необходимо сделать динамически выделенный виртуальной машине 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. рисунок 17, поле «Адрес IPv4»), в столбце «Маска подсети» – указать стандартное значение маски – 255.255.255.0, а в столбце «Шлюз» – значение маршрута по умолчанию (см. рисунок 17), и затем активировать опцию «Сохранить» (см. рисунок 18).

Настройка статического IP-адреса для виртуальной машины завершена.

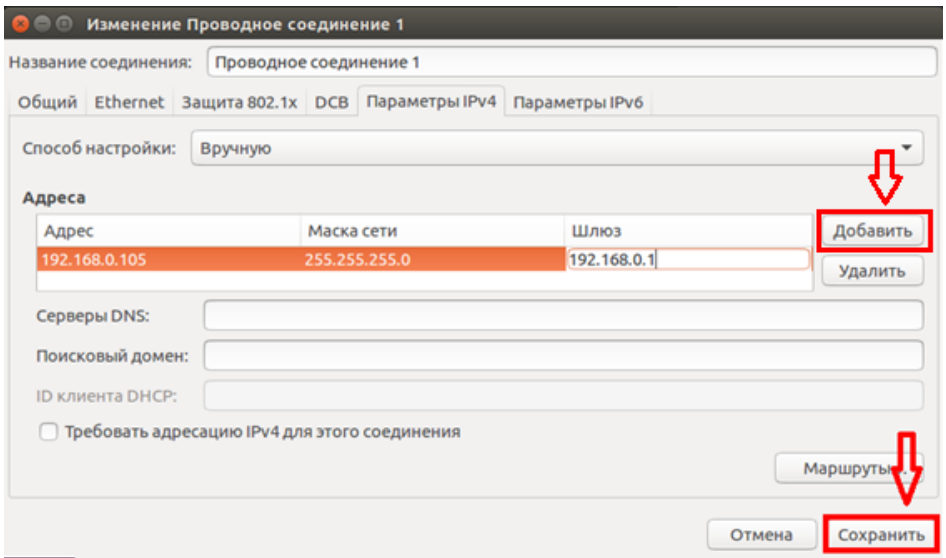


Рисунок 18 – Изменение настроек проводного соединения

4.3 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 машин, 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 выполнить команду для обновления таблиц MySQL, в которых хранится информация о пользователях:

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

5 СИНХРОНИЗАЦИЯ СТРУКТУРЫ ДАННЫХ

5.1 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 (см. рисунок 19).

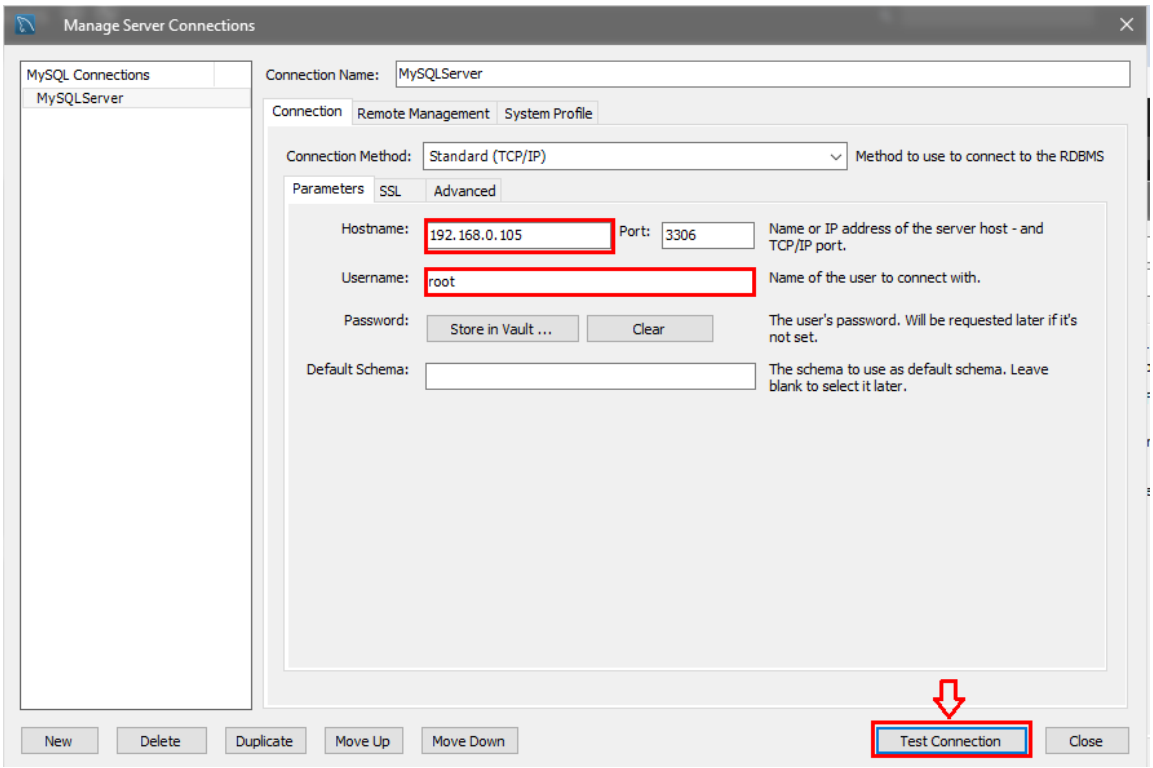


Рисунок 19 – Настройка подключения к удаленному серверу MySQL

Активировав опцию «Test connection» необходимо проверить статус подключения. Если все сделано правильно, появится сообщение об успешном установлении соединения с сервером MySQL (см. рисунок 20).

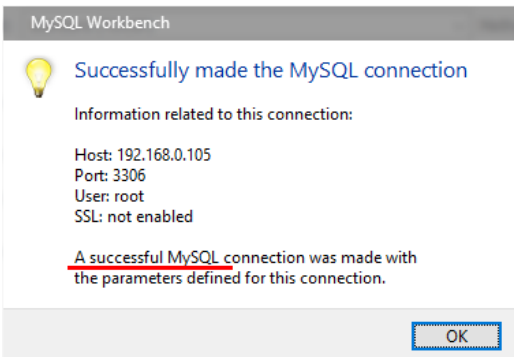


Рисунок 20 – Проверка подключения к удаленному серверу MySQL

5.2 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 рисунке 21 представлено окно настроек экспорта.

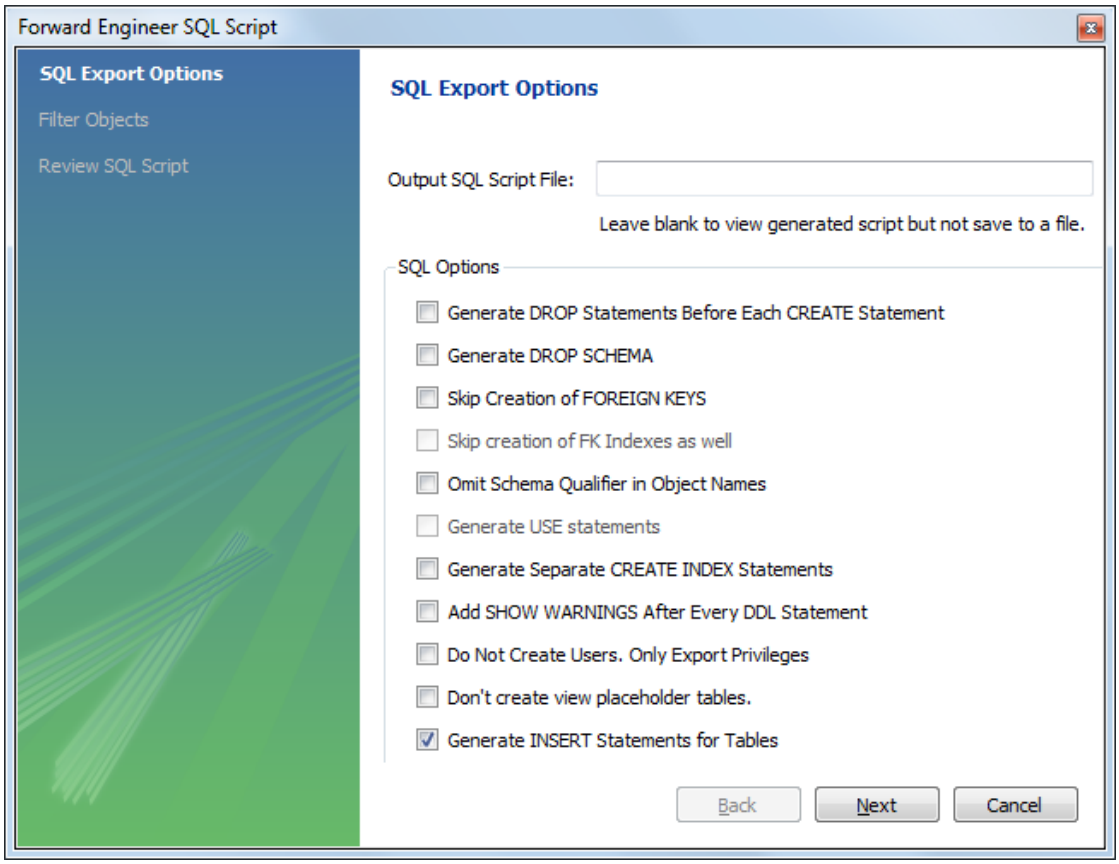


Рисунок 21 – Окно настроек экспорта модели MySQL Workbench на сервер

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 (см. рисунок 22).

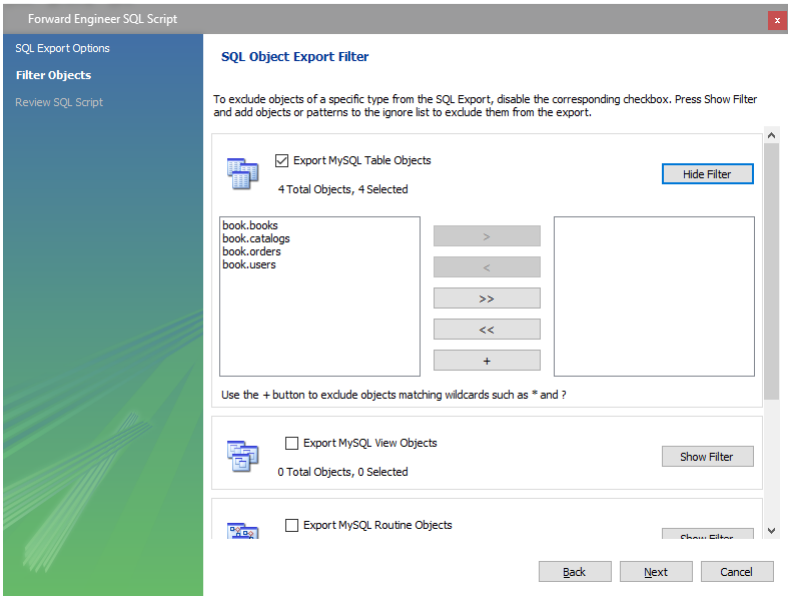


Рисунок 22 – Выбор объектов для экспорта

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

5.3 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 (см. рисунок 23).

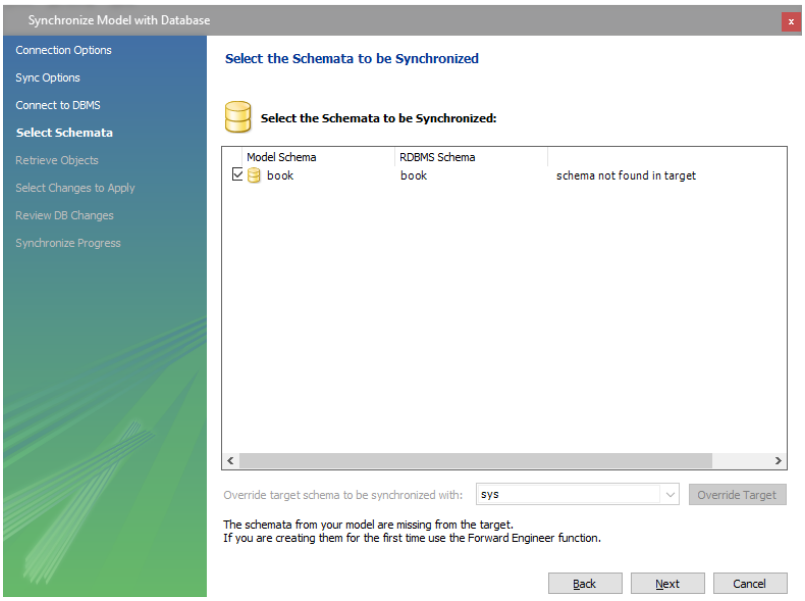


Рисунок 23 – Доступные модели и базы данных для синхронизации

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 (см. рисунок 24).

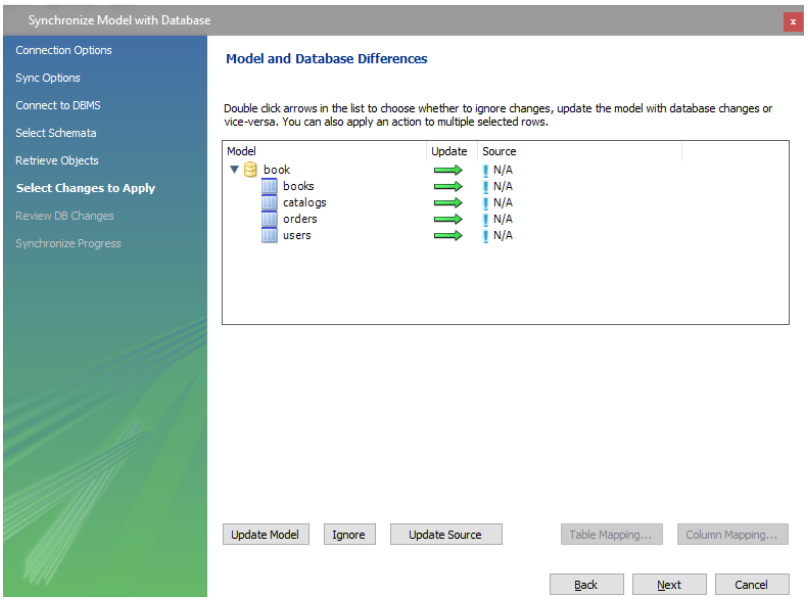


Рисунок 24 – Список различий между локальной и удаленной схемами данных

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 рисунке 25.

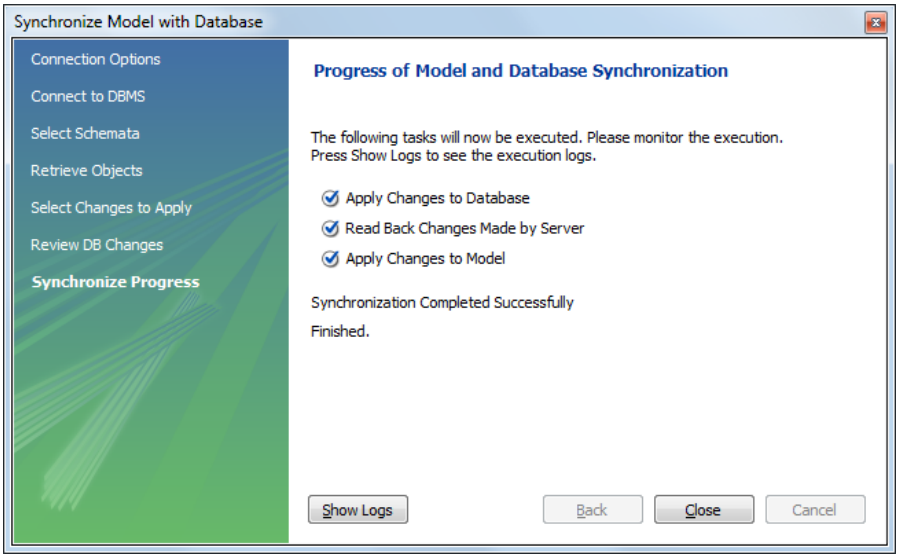


Рисунок 25 – Отчет об успешно выполненной синхронизации

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

5.4 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, описанному в п. 4.2. Его можно так же использовать, если требуется простая выгрузка схемы данных на сервер без синхронизации.

6 ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1. Нормализовать отношения в базе данных, построенной в лабораторной работе №1, до третьей и четвертой нормальной формы.
2. Построить полную атрибутивную модель базы данных в нотации IDEF1X.
3. Создать виртуальную машину, установить на ней операционную систему Ubuntu и настроить на ней SQL-сервер.
4. Настроить удаленное подключение к виртуальной (серверной) машине с реальной (клиентской) и выгрузить на сервер разработанную в MySQL Workbench схему.
5. Разработать приложение для организации доступа к данным, хранящихся в БД, которая была разработана в лабораторных работах №№1, 2 и 3. 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, реализованном в лабораторной работе №1).
6. 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.

7. 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.

7 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.