

**Автономная некоммерческая организация профессионального образования  
«ПЕРМСКИЙ ГУМАНИТАРНО-ТЕХНОЛОГИЧЕСКИЙ КОЛЛЕДЖ»  
(АНО ПО «ПГТК»)**

**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ  
ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ  
МДК 01.02. УПРАВЛЕНИЕ И АВТОМАТИЗАЦИЯ БАЗ  
ДАННЫХ**

для специальности

**09.02.11 Разработка и управление программным обеспечением  
(код и наименование специальности)**

Квалификация выпускника  
**Программист**

Форма обучения  
**Очная**

Пермь 2026

Методические рекомендации по выполнению практических работ  
МДК 01.02. УПРАВЛЕНИЕ И АВТОМАТИЗАЦИЯ БАЗ ДАННЫХ составлен  
в соответствии с требованиями Федерального государственного  
образовательного стандарта среднего профессионального образования по  
специальности 09.02.11 Разработка и управление программным обеспечением

Данные методические рекомендации помогут организовать  
самостоятельную деятельность студентов на основе деятельного и  
компетентного подходов к обучению, что соответствует ФГОС СПО по  
специальности 09.02.11 Разработка и управление программным обеспечением.

Автор – составитель: Могильникова Н.С., старший преподаватель.

Методические рекомендации по выполнению практических работ предназначен для оценивания достижений запланированных результатов по дисциплине МДК 01.02. УПРАВЛЕНИЕ И АВТОМАТИЗАЦИЯ БАЗ ДАННЫХ. Методические рекомендации по выполнению практических работ представляет собой комплект материалов для проведения практических занятий (в форме практической подготовке) и осуществления контроля за выполнением работ.

Методические рекомендации по выполнению практических работ позволяет оценивать:

Код ОК, ПК	Уметь	Знать	Владеть навыками
ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам	распознавать задачу и/или проблему в профессиональном и/или социальном контексте, анализировать и выделять её составные части определять этапы решения задачи, составлять план действия, реализовывать составленный план, определять необходимые ресурсы выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы владеть актуальными методами работы в профессиональной и смежных сферах оценивать результат и последствия своих действий (самостоятельно или с помощью наставника)	актуальный профессиональный и социальный контекст, в котором приходится работать и жить структура плана для решения задач, алгоритмы выполнения работ в профессиональной и смежных областях основные источники информации и ресурсы для решения задач и/или проблем в профессиональном и/или социальном контексте методы работы в профессиональной и смежных сферах порядок оценки результатов решения задач профессиональной деятельности	
ОК 02 Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности	определять задачи для поиска информации, планировать процесс поиска, выбирать необходимые источники информации выделять наиболее значимое в перечне информации, структурировать получаемую информацию, оформлять результаты поиска оценивать практическую значимость результатов поиска применять средства информационных технологий для решения профессиональных задач использовать современное программное обеспечение в профессиональной деятельности	номенклатура информационных источников, применяемых в профессиональной деятельности приемы структурирования информации формат оформления результатов поиска информации современные средства и устройства информатизации, порядок их применения программное обеспечение в профессиональной деятельности, в том числе цифровые средства психологические основы деятельности коллектива	

	использовать различные цифровые средства для решения профессиональных задач		
ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста	грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке проявлять толерантность в рабочем коллективе	правила оформления документов правила построения устных сообщений особенности социального и культурного контекста	
ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках	понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы участвовать в диалогах на знакомые общие и профессиональные темы строить простые высказывания о себе и о своей профессиональной деятельности кратко обосновывать и объяснять свои действия (текущие и планируемые) писать простые связные сообщения на знакомые или интересующие профессиональные темы	правила построения простых и сложных предложений на профессиональные темы основные общеупотребительные глаголы (бытовая и профессиональная лексика) лексический минимум, относящийся к описанию предметов, средств и процессов профессиональной деятельности особенности произношения правила чтения текстов профессиональной направленности	
ПК.1.1 Проектировать базы данных	анализировать предметную область и выделять основные сущности; определять требования к базе данных; разрабатывать концептуальную, логическую и физическую модели баз данных; проектировать схему базы данных; работать с современными case-средствами проектирования баз данных; определять связи между таблицами;	основные положения теории баз данных, хранилищ данных, баз знаний; основные принципы структуризации и нормализации базы данных; основные принципы построения концептуальной, логической и физической модели данных; методы описания схем баз данных в современных системах управления базами данных;	разработки концептуальной модели базы данных; разработки инфологической модели базы данных; разработки физической модели базы данных; разработки требований к базе данных нормализация структуры базы данных документирования схемы базы данных, включая диаграммы ER и описания таблиц; документирования прав доступа и безопасности

	<p>определять типы данных для полей таблиц;</p> <p>оформление документации на спроектированную базу данных</p> <p>разработки схемы базы данных, используя NoSQL модели данных, такие как документо-ориентированные, ключ-значение, колоночные и др.;</p>	<p>структуры данных систем управления базами данных, основные понятия и принципы проектирования баз данных;</p> <p>структура реляционной базы данных;</p> <p>язык SQL и особенности его реализации в различных системах управления базами данных;</p> <p>оптимизация производительности баз данных</p> <p>принципы безопасности хранения данных</p>	<p>базы данных, включая учетные записи пользователей и их роли</p>
<p>ПК.1.2</p> <p>Разрабатывать объекты баз данных в соответствии с результатами анализа предметной области</p>	<p>разрабатывать объекты баз данных</p> <p>создавать таблицы, индексы, ограничения и другие объекты базы данных</p> <p>оптимизировать запросы к базе данных для повышения производительности</p> <p>разрабатывать хранимые процедуры и триггеры для баз данных;</p> <p>разрабатывать необходимые для различных групп пользователей представления</p>	<p>основы реляционной модели данных</p> <p>язык SQL и его основные команды</p> <p>принципы нормализации баз данных</p> <p>принципы работы с различными СУБД</p> <p>общий подход к организации представлений, таблиц, индексов и кластеров;</p> <p>методы организации целостности данных;</p> <p>способы контроля доступа к данным и управления привилегиями</p>	<p>работы с различными объектами базы данных</p>
<p>ПК.1.3</p> <p>Реализовывать базу данных в конкретной системе управления базами данных</p>	<p>разрабатывать объекты базы данных, такие как таблицы, индексы и связи между ними;</p> <p>программировать и создавать хранимые процедуры, функции и триггеры для обработки данных;</p> <p>управлять данными в базе данных, включая ввод, обновление и удаление данных;</p> <p>оптимизировать запросы и проводить мониторинг производительности базы данных;</p> <p>работать с NoSQL базами данных;</p>	<p>основные принципы создания объектов базы данных;</p> <p>синтаксис и основные приемы работы с SQL;</p> <p>методы оптимизации запросов и повышения производительности базы данных;</p> <p>основные принципы управления данными и обслуживания базы данных;</p> <p>основные принципы работы NoSQL баз данных и их моделей данных;</p> <p>преимущества и недостатки NoSQL технологий по сравнению</p>	<p>создания таблиц базы данных с определением структуры и типов данных для каждого атрибута;</p> <p>определения первичных и внешних ключей для установления связей между таблицами;</p> <p>создания индексов для оптимизации запросов и повышения производительности;</p> <p>разработки хранимых процедур, функций и триггеров для обработки данных и поддержки бизнес-логики;</p> <p>ввода, обновления и удаления данных в соответствии с</p>

	использовать запросы для работы с данными в NoSQL базах данных; оптимизировать производительность NoSQL баз данных.	с реляционными базами данных; методы оптимизации производительности NoSQL баз данных; основные принципы управления данными и обслуживания NoSQL баз данных.	требованиями бизнес-процессов; оптимизации запросов для повышения производительности системы; создания баз данных на основе NoSQL технологий создания запросов для работы с данными в NoSQL базах данных; оптимизации производительности NoSQL баз данных, используя индексы и другие техники;
ПК.1.4 Администрировать базы данных	устанавливать и настраивать СУБД; создавать и удалять базы данных; создавать пользователей и назначать права доступа; оптимизировать запросы к базе данных; обеспечивать безопасность баз данных; создавать и настраивать базы данных в соответствии с требованиями бизнеса; управлять транзакциями и контролировать целостность данных; обеспечивать безопасность и управлять доступом к данным; создавать и восстанавливать резервные копии данных работать с индексами и оптимизировать производительность запросов нормализовать базы данных и проектировать эффективные структуры данных мониторить и анализировать производительность баз данных работать с нереляционными базами данных и выбирать наиболее подходящий тип	архитектура СУБД основные принципы администрирования баз данных методы мониторинга и оптимизации работы баз данных принципы резервного копирования и восстановления баз данных методы защиты баз данных от внешних угроз особенности работы с различными СУБД Язык SQL (Structured Query Language) управление транзакциями и контроль целостности данных управление доступом и безопасностью баз данных резервное копирование и восстановление данных оптимизация производительности баз данных работа с индексами и оптимизация запросов мониторинг и анализ производительности принципы работы с реляционными базами данных принципы работы с нереляционными базами данных	установки и настройки СУБД; создания и удаления баз данных; восстановления баз данных; резервного копирования баз данных; создания пользователей и назначения прав доступа; оптимизации запросов к базе данных мониторинг и обслуживания NoSQL баз данных, включая резервное копирование и восстановление данных.

	базы данных для конкретной задачи		
ПК.1.5 Защищать информацию в базе данных с использованием технологии защиты информации	<p>разрабатывать и внедрять системы защиты баз данных от несанкционированного доступа</p> <p>разрабатывать и внедрять системы резервного копирования и восстановления баз данных</p> <p>проводить аудит безопасности баз данных</p> <p>устанавливать и настраивать механизмы аутентификации и авторизации пользователей</p> <p>создавать и управлять ролями и правами доступа к данным</p> <p>шифровать данные и обеспечивать их конфиденциальность</p> <p>контролировать целостность данных и обнаруживать изменения</p> <p>использовать механизмы аудита для отслеживания доступа к данным</p> <p>использовать механизмы мониторинга для обнаружения угроз безопасности</p> <p>создавать и управлять защищенными соединениями с базой данных</p> <p>использовать механизмы защиты от SQL-инъекций и других видов атак</p> <p>создавать и управлять бэкапами и резервными копиями данных</p> <p>обеспечивать безопасность базы данных при использовании облачных сервисов.</p>	<p>методы защиты баз данных от несанкционированного доступа</p> <p>методы создания и восстановления резервных копий баз данных</p> <p>особенности работы с различными типами СУБД</p> <p>методы проведения аудита безопасности баз данных</p> <p>принципы криптографии и методов шифрования данных</p> <p>стандарты и протоколы безопасности, таких как SSL/TLS, SSH, Kerberos и др.</p> <p>методы аутентификации и авторизации пользователей, включая использование паролей, сертификатов и биометрических данных</p> <p>методы контроля доступа, включая создание ролей и групп пользователей, управление правами доступа и аудит доступа к данным</p> <p>методы обнаружения и предотвращения атак, включая защиту от SQL-инъекций, DoS/DDoS-атак и других угроз безопасности</p> <p>методы мониторинга и анализа журналов событий для обнаружения угроз безопасности и анализа производительности базы данных</p> <p>методы создания и управления защищенными соединениями с базой данных, включая VPN-туннели и SSL-шифрование</p> <p>методы создания и управления бэкапами и</p>	<p>использования стандартных методов защиты объектов базы данных;</p> <p>разработки и внедрения систем защиты баз данных от несанкционированного доступа</p> <p>разработки и внедрения систем резервного копирования и восстановления баз данных</p> <p>аудита безопасности баз данных</p>

		резервными копиями данных, включая использование инкрементальных и дифференциальных бэкапов методы обеспечения безопасности базы данных при использовании облачных сервисов, включая защиту от утечки данных и управление доступом к облачным ресурсам законодательство и стандарты безопасности, такие как GDPR, HIPAA, PCI DSS и др.	
--	--	--	--

## Перечень практических занятий.

### Практическое занятие «Построение схемы базы данных»

Цель работы: Сформировать умения добавлять таблицы в базу данных с целью расширения ее функциональных возможностей.

Иногда в процессе разработки базы данных или в процессе опытной эксплуатации ее возникает необходимость добавления в нее новых таблиц. Очевидно, что спроектированная нами в предыдущей работе база данных Библиотека обладает очень ограниченными возможностями. Эта база данных, состоящая из трех таблиц: Издательства, Книги и Темы, не позволяет автоматизировать работу с читателями. В ней отсутствует информация о читателях.

В данной работе мы научимся добавлять таблицы в базу данных с целью расширения ее функциональных возможностей. Создание новых таблиц осуществляется точно так же, как это мы делали в предыдущей работе. Для добавления таблиц в ранее созданную схему данных и установления связи между таблицами используется кнопка Отобразить таблицу, размещенная на панели инструментов Связь.

#### Задание

1. Откройте базу данных Библиотека. Создайте в ней структуру таблицы Читатели, которая будет содержать следующие поля: Код читателя, Фамилию, Имя, Отчество, Домашний телефон, Домашний адрес. Типы данных для полей таблицы, их свойства определите самостоятельно по смыслу. В качестве ключа укажите поле Код читателя.
2. Аналогичным способом создайте структуру таблицы Выдача книг. В эту структуру включите три поля: Код читателя, Код книги, Дата заказа. В этой таблице ключевое поле не задавайте. Для поля Дата заказа укажите тип данных – Дата/время. Обратите внимание на то, что в последствии ключ Код читателя в таблице Читатели будет связываться с полем Код читателя в таблице Выдача книг. Поэтому эти поля должны иметь соответствующие типы данных и свойства.
3. Установите между добавленными таблицами: Читатели и Выдача книг, а также ранее созданными



таблицами: Издательства, Книги и Темы, связи так, как это показано в окне Схема данных на рис. 1.

Напомним, что для установления связи между таблицами надо открыть окно Схема данных. При его открытии появляется диалоговое окно Добавление таблицы, в котором надо выделить имена тех таблиц, между которыми будут устанавливаться связи. После этого нажимают кнопки Добавить и Заккрыть. Затем в окне Схема данных с помощью мыши перетаскивают ключевое поле одной таблицы на соответствующее поле в другой таблице. В появившемся окне Связи задают режим Обеспечение целостности данных и его подрежимы: каскадное обновление связанных полей и каскадное удаление связанных записей и нажимают кнопку Создать.

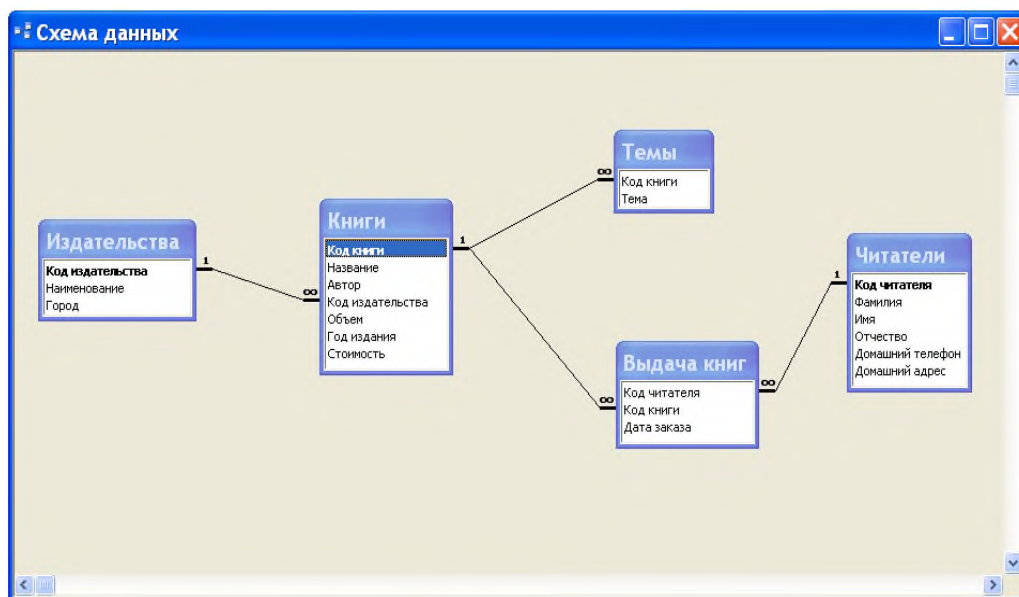


Рис. 1. Схема расширенной базы данных Библиотека.

4. Откройте таблицу Читатели и введите в нее данные, приведенные в таблице 1.

Таблица 1 Данные для ввода в таблицу Читатели

Код читателя	Фамилия	Имя	Отчество	Домашний телефон	Домашний адрес
1	Аксенов	Виктор	Сергеевич	252-88-13	ул. Есенина, 15-19
2	Голубева	Елена	Андреевна	220-99-29	ул. Чкалова, 7-38
3	Васильев	Игорь	Петрович	232-64-78	ул. Богдановича, 102-34
4	Кучеров	Валентин	Степанович	266-24-95	ул. Кнорина, 27-5
5	Мастяница	Вячеслав	Иванович	246-42-25	ул. Плеханова, 34-98
6	Победимская	Лариса	Анатольевна		ул. Чкалова, 9-10
7	Литвин	Борис	Николаевич	239-55-76	пр. Независимости, 46-54
8	Германович	Рита	Мироновна	278-31-51	ул. Казинца, 26-9
9	Бинцаровский	Теодор	Петрович		ул. Корженевская, 1-288

5. Введите в таблицу Выдача книг данные, приведенные в таблице 2.

Таблица 2 Данные для ввода в таблицу Выдача книг

Код читателя	Код книги	Дата заказа	Код читателя	Код книги	Дата заказа
1	1	1.09.07	4	3	7.01.08
1	3	5.07.08	4	4	25.10.07
1	4	21.10.07	5	2	23.04.08
2	1	4.11.07	6	1	18.06.08
3	2	3.08.08	7	3	20.01.08
8	7	25.12.07	9	6	2.02.08

Обратите внимание на то, что, если бы вы попробовали вначале ввести данные в таблицу Выдача книг, а затем в таблицу Читатели, то MS Access это не позволил бы сделать. Поэтому мы специально раньше установили связи между таблицами, а затем уже вводили данные в таблицы. В этом случае MS Access будет проверять целостность данных.

### Практическое занятие «Составление словаря данных»

Цель работы. Изучение средства создания словаря данных Access.

Теоретическая часть

Определив отдельные элементы данных, составляющие таблицы базы данных, и установив отношения между ними, необходимо подготовить описание базы данных, называемое словарем данных. Словари данных имеют огромное значение для баз данных. Недокументированной базой данных тяжело управлять и ее трудно обслуживать. Ошибки и промахи, допущенные на этапе ее разработки, во многих случаях можно обнаружить при подготовке предварительного словаря данных.

Завершив разработку структуры базы данных и убедившись в ее правильности, требуется довести до конца разработку подробного словаря данных. По мере добавления в приложение новых или модификации существующих форм и отчетов, необходимо соответствующим образом обновлять и словарь. Даже при создании базы данных для личного использования упрощенная версия словаря данных позволит получить большую пользу при малых затратах времени.

Стандартный словарь данных

Словарь данных содержит информацию о базе данных в целом, обо всех таблицах, полях, включенных в таблицы, о первичных и внешних ключах, а также включает толкование используемых кодов. В словаре также приводится назначение и описание каждого приложения, использующего базу.

Поскольку словари данных имеют иерархическую структуру, они могут храниться в традиционном формате, поддерживаемом текстовыми редакторами Windows. Ниже приводится структура стандартного словаря данных, в котором заданы традиционные заголовки:

1. БАЗА ДАННЫХ—Полное название базы и имя файла.

Описание назначения и общего содержания базы данных, а также лиц, которые могут ею пользоваться. Список приложений, работающих с базой, и информация о других базах данных, использующих данные из данной базы. Если имеются, то сюда же включаются диаграммы базы данных.

А. ОБЛАСТЬ ДАННЫХ — Название группы, к которой принадлежат таблицы.

Если таблицы классифицируются по группам, например, финансовая группа, то включается описание

каждой группы.

1. ТАБЛИЦА — Таблицы, входящие в область данных.
  - а) ДОСТУП — Права пользователей на доступ к таблице.
  - б) ЗАПИСЬ — Общее определение элементов данных.
    - (1) ПЕРВИЧНЫЙ КЛЮЧ - Поле (поля) первичного ключа.
      - (а) ИНДЕКС — Описание индекса первичного ключа.
    - (2) ВНЕШНИЕ КЛЮЧИ - Внешние ключевые поля.
      - (а) ИНДЕКС - Индексы . внешних ключей.
    - (3) ПОЛЯ — Неключевые поля.
      - (а) ПЕРЕЧИСЛИМЫЕ МНОЖЕСТВА — Допустимые коды для полей.

За каждым заголовком идет текст, описывающий назначение элемента, базы данных, к которому относится заголовок. Последующие разделы словаря включают описания объектов, которые используют таблицы базы данных, с подзаголовками для запросов, форм и отчетов. Изображения, снятые с экрана, и копии отчетов также добавляются в словарь данных. Распечатки кода программ обычно приводятся в приложениях словаря. Подробные словари данных необходимы для обслуживания базы данных. Кроме того, описание словаря можно представить в табличной форме.

### Интегрированный словарь данных

Надстройка Архивариус (Documentor), которая впервые появилась в Access 2.0., позволяет создать отчет, где подробно описываются объекты и значения их свойств для текущей базы данных.

Средство Access 97 "Публикация в MS Word", запускаемое с помощью кнопки на панели инструментов, позволяет сохранить отчет в формате .RTF. Затем созданный файл можно импортировать в Microsoft Word или любой другой текстовый процессор, который обрабатывает файлы в формате RTF, например, WordPad. Кроме того, можно экспортировать отчет в формате BIFF, нажав на панели инструментов кнопку "Анализ в MS Excel".

Во многих случаях Архивариус (Documentor) сообщит вам больше, чем вы хотите знать о вашей базе данных; например, полный отчет обо всех объектах базы данных Борея (Northwind) составляет около 400 страниц. Однако чаще требуется поместить в словарь данных только информацию о таблицах и, возможно, запросах.

### Задание на выполнение работы

Для создания словаря данных с помощью Архивариуса (Documentor):

1. Откройте требуемую базу данных и выберите команду "Сервис, Анализ, Архивариус" (Tools, Analyze, Documentor).
2. Выберите вкладку с требуемым типом объектов, которые необходимо описать. Если выбрать вкладку "Все типы объектов" (All Object Types) и нажать кнопку "Выбрать все" (Select All), то можно создать описание для всех объектов базы данных.
3. Нажмите кнопку "Параметры" (Options) для вывода диалогового окна "Печать описания таблицы" (Print Table Definition). По умолчанию печатается наиболее подробное описание таблиц и индексов. Если база данных не имеет системы защиты, то можно сбросить флажок "Разрешения для пользователей и групп"

(Permissions By User and Group), чтобы не выводить в отчете данные о правах доступа. Нажмите кнопку ОК для возврата в первое диалоговое окно Архивариуса (Documentor).

4. Выберите требуемые таблицы, нажимая кнопку "Выделить" (Select). При этом будет установлен флажок напротив имени указанной таблицы. Кроме того, можно выделить объект, дважды щелкнув по его названию. Если необходимо документировать все таблицы базы данных, нажмите кнопку "Выбрать все" (Select All).

5. Выберите вкладку "Запросы" (Queries) для вывода запросов базы данных. Нажмите кнопку "Параметры" (Options) для вывода диалогового окна "Печать описания запроса" (Print Table Definition). Сбросьте флажок "Разрешения для пользователей и групп" (Permissions By Group and Group), чтобы не выводить в отчете данные о правах доступа, и задайте нужные параметры, чтобы не выводить данные, дублирующие информацию о полях таблиц. Нажмите кнопку ОК.

6. Выберите запрос "Сведения о заказах", дважды щелкнув по его имени, и нажмите кнопку ОК для создания отчета длиной 13 страниц.

7. Вскоре появляется окно предварительного просмотра "Описание объекта" (Object Definition). Обратите внимание на то, что отчет о трех объектах занимает одну страницу.

8. Нажмите на панели инструментов кнопку "Печать" (Print) для печати отчета либо щелкните по кнопке "Связи с Office" (Office Links) для создания файла в формате .RTF или .XLS

9. Нажмите на панели инструментов кнопку "Закрыть" (Close) либо дважды щелкните по кнопке системного меню для закрытия окна предварительного просмотра.

Документирование других объектов базы данных выполняется аналогичным образом. Можно вывести данные о самой базе данных, выбрав вкладку "Текущая база данных" (Current Database), или распечатать данные из выбранных форм, отчетов, макросов и модулей.

### **Практические занятия «Разработка технических требований к серверу баз данных», «Разработка требований к корпоративной сети».**

Цель: научиться разрабатывать технические требования к серверу баз данных, к корпоративным сетям.  
Задание.

Проектирование (модернизация) корпоративной сети является составной частью разработки (модернизации) распределенной информационной системы управления (РИСУ) предприятия. Проектирование РИСУ включает:

- проектирование объектов данных, которые будут реализованы в базе данных;
- проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;
- проектирование конкретной сетевой среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т.п.

Параметры (показатели) корпоративной сети должны соответствовать целям проекта РИСУ. В разделе 1 было показано, что основное требование к корпоративной сети – это обеспечение всем категориям пользователей доступа к разделяемым ресурсам сети с заданным качеством обслуживания (QoS – Quality of Service), причем качество обслуживания характеризуется следующими показателями:

- требуемая пропускная способность системы  $\nu$  [Мбит/с];
- требуемое время реакции системы на запрос  $T_{don}$  [с];
- безотказность работы системы;
- требуемый уровень информационной безопасности.

Одним из основных критериев качества обслуживания является производительность, причем в качестве показателей производительности используются время реакции, пропускная способность  $\nu$  (бит/с, пакетов/с) и задержка передачи.

Информационная безопасность – это защищенность сетевых ресурсов от несанкционированного доступа. В качестве показателей надежности используются: среднее время наработки на отказ  $T_{отк}$ , среднее время ремонта  $T_{рем}$  и коэффициент готовности  $K_g = T_{отк} / (T_{отк} + T_{рем})$ , определяющий вероятность работоспособного состояния сети в любой момент времени. Важным требованием к надежности вычислительных сетей является также отказоустойчивость, т.е. сохранение работоспособности при отказе отдельных элементов.

Ряд требований к компьютерным сетям связан с их эксплуатацией и развитием, а также с обеспечением удобства работы для пользователей.

Совместимость сетевого оборудования и программного обеспечения позволяет объединять разнообразные компоненты, приобретенные от разных производителей.

Расширяемость – это возможность расширения сети (добавления отдельных элементов, наращивания длины сегментов, замены оборудования на более мощное) без особых проблем. Масштабируемость – это возможность расширения сети в широких пределах без снижения производительности (с получением прогнозируемого эффекта).

Важным требованием, характеризующим удобство работы пользователей, является прозрачность доступа к сетевым ресурсам. Прозрачность означает, что при работе в сети пользователю не требуется знать детали устройства системы.

Требуемые пропускная способность и время реакции системы зависят от информационной нагрузки на сеть, которую создают пользователи. Для расчета требуемой пропускной способности  $\nu$  будем использовать аппарат теории систем массового обслуживания (СМО) и сетей массового обслуживания (СеМО) [13]. СеМО – это совокупность систем массового обслуживания, в которой заявки с выходов одних СМО могут поступать на входы других.

Например, в соответствии с теорией СеМО основным показателем информационной нагрузки на сервер сети является коэффициент загрузки сервера  $\rho = \lambda / \mu$ , где  $\lambda$  – интенсивность информационного потока, создаваемого пользователями, между рабочими станциями и сервером;  $\mu$  – интенсивность обслуживания сервера. В качестве единиц измерения  $\lambda$  и  $\mu$  используют число бит, пакетов или запросов в секунду [бит/с, пакетов/с, запросов/с].

При моделировании РИСУ в виде СеМО приходится исследовать системы, включающие в качестве подсистем многие системы массового обслуживания. Наиболее развита аналитическая теория экспоненциальных СеМО и точные методы расчета вероятностно-временных характеристик сложных систем. При моделировании РИСУ используют следующие разновидности моделей РИСУ: разомкнутая СеМО (РСеМО), замкнутая СеМО (ЗСеМО) и комбинированная СеМО. Для РСеМО характерно наличие одного или нескольких независимых внешних источников, которые генерируют заявки, поступающие в сеть, независимо от того, сколько заявок уже находится в сети, причем в любой момент времени в РСеМО может находиться произвольное число заявок (от 0 до  $\infty$ ). В ЗСеМО циркулирует фиксированное число заявок, а внешние независимые источники отсутствуют. В комбинированной СеМО постоянно циркулирует определенное число заявок, а также есть заявки, поступающие от внешних независимых источников.

Существует два основных режима работы сети: пакетный и диалоговый. Для диалогового режима важным параметром является среднее время обдумывания пользователя  $1/\eta$  [с] на один запрос к системе, где  $\eta$  – интенсивность обдумывания. Методика приближенного расчета пропускной способности магистрали сети приведена в разделе 1.

Для приложений с пакетным режимом работы величина  $1/\eta$  приблизительно равна среднему интервалу между моментами появления запросов к соответствующей сетевой службе. Интенсивность потока требований, генерируемых одним пользователем, можно вычислить по формуле  $\lambda_1 \approx 1/(1/\eta + T_{don})$ , где  $T_{don}$  – допустимое время реакции системы на запрос пользователя, например:  $1/\eta = 120$  с,  $T_{don} = 60$  с,  $\lambda_1 \approx 1/(120 + 60) = 0,0056$  запр/с.

## 8.2. Расчет параметров сети

### 8.2.1. Элементы теории СеМО для расчета параметров сети

Для оптимизации производительности сети ЭВМ используют методы и средства измерения, анализа и моделирования. Клиент-серверная архитектура и распределенная обработка данных в сети усложняют задачи моделирования.

Аналитическое моделирование основано на использовании моделей СМО и СеМО [13] и, как правило, связано со значительными упрощениями. Тем не менее результаты аналитического исследования могут быть очень ценными, даже если они не учитывают всех деталей реальной сети. Такие модели позволяют достаточно быстро получить приближенную инженерную оценку влияния характеристик оборудования и программного обеспечения на показатели производительности сети.

Модель сети строится из отдельных блоков, каждый из которых представляет один узел или канал передачи сети. Блок состоит из буферного накопителя заявок и обслуживающего элемента (рис.8.1). При анализе сетей ЭВМ в качестве заявок могут фигурировать запросы пользователей или отдельные пакеты, на которые разбиваются эти запросы. На вход блока поступает поток заявок, характеризуемый функцией распределения интервалов времени между моментами поступления заявок  $A(t)$ .

Интенсивность  $\lambda$  входного потока заявок – это среднее число заявок, поступающих на вход блока в единицу времени. Обратная величина  $1/\lambda$  – это среднее значение интервала между моментами поступления заявок, которое определяется интегралом  $\int_0^\infty t dA(t)$ .

Интенсивность обслуживания блока – это  $\mu$  среднее число обрабатываемых заявок в единицу времени. Обратная величина  $1/\mu$  – это среднее значение длительности обслуживания заявки, которое определяется интегралом  $\int_0^\infty t dB(t)$ , где  $B(t)$  – функция распределения длительности обслуживания. Отношение  $\rho = \lambda/\mu$  называется коэффициентом загрузки блока. Реальный блок имеет буфер ограниченной емкости  $r$  (см. рис.8.1,б). Идеализированный модуль может иметь неограниченный по емкости буфер (см. рис.8.1,а).



Рис. 8.1.

Блок М/М/1. Рассмотрим самую простую модель типа М/М/1 (один обслуживающий элемент, неограниченная емкость буфера, экспоненциальные законы распределения интервалов времени между моментами поступления заявок и времени обслуживания, дисциплина обслуживания FIFO) для блока, изображенного на рис.8.1, а. В этом случае  $A(t) = 1 - e^{-\lambda t}$ ,  $B(t) = 1 - e^{-\mu t}$ , среднее время задержки заявки в блоке

$$T = 1 / (\mu - \lambda), \quad (8.1)$$

а среднее число заявок в блоке (в очереди и в процессе передачи)

$$L = \lambda / (\mu - \lambda). \quad (8.2)$$

Среднее время ожидания в очереди  $W = T - (1/\mu) = \rho / (\mu - \lambda)$ , а среднее число заявок в очереди  $L_w = L - \rho = \rho\lambda / (\mu - \lambda)$ .

Блок М/Г/1. Эта модель отличается от модели типа М/М/1 только тем, что распределение времени

обслуживания  $B(t)$  может быть произвольным. Рассмотрим случай, когда распределение  $B(t)$  задается для блока двумя параметрами: интенсивностью обслуживания  $\mu$  и дисперсией времени обслуживания

$$D = \int_0^{\infty} t^2 dB(T) - 1/\mu^2.$$

Тогда среднее время нахождения заявки в очереди  $W = (1 + v^2) W^{\Pi}$ , где  $W^{\Pi} = (\rho/2\mu) \times (1-\rho)^{-1}$  – время нахождения заявки в очереди при постоянной длительности обслуживания;  $v^2 = \mu^2 D$  – квадрат коэффициента вариации времени обслуживания. Для постоянного времени обслуживания  $v=0$ , а для экспоненциального распределения времени обслуживания  $v=1$ . Для модели  $M/G/1$  оценка времени пребывания заявки в блоке  $T = W + (1/\mu)$ , длины очереди в буфере  $L_w = \lambda W$  и общего числа заявок в блоке  $L = L_w + \rho$ .

Блоки  $M/M/1/r$  и  $M/G/1/r$ . Модель типа  $M/G/1/r$  для блока, изображенного на рис.8.1,б, отличается от модели  $M/G/1$  тем, что емкость буфера ограничена величиной  $r$  (предполагается, что обрабатываемая заявка находится также в буфере). Эта модель характеризуется вероятностью потери заявки (отказа в обслуживании) [11]

$$P_{отк} \approx (1-\rho) \rho^{\Psi(r,v)} / (1-\rho^{\Psi(r,v)+1}), \quad (8.3)$$

где  $\Psi(r,v) = 2r/(1+v^2)$ , причем  $v$  – коэффициент вариации. Абсолютная пропускная способность блока  $M/G/1/r$

$$\lambda_{ABC} = \lambda (1 - P_{отк}).$$

При  $v=1$  формула дает точное значение  $P_{отк}$  для экспоненциального распределения  $B(t)$ , т.е. для блоков  $M/M/1/r$ .

Сеть блоков  $M/M/1$ . Модель сети можно представить в виде СеМО, т.е. сети блоков [3]), содержащих буферы. Простые аналитические формулы можно получить для открытой сети блоков  $M/M/1$ , пример которой представлен на рис.8.2.

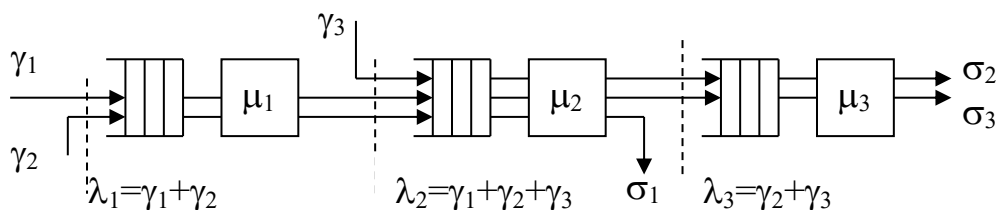


Рис. 8.2.

В этой сети, состоящей из трех блоков, три входных потока пакетов, имеющих интенсивности  $\gamma_1$ ,  $\gamma_2$  и  $\gamma_3$  соответственно. Требуется оценить среднюю задержку пакетов для каждого потока. Потоки, выходящие из блоков, не ветвятся, причем выходные потоки  $\sigma_1 = \gamma_1$ ,  $\sigma_2 = \gamma_2$  и  $\sigma_3 = \gamma_3$ , т.е. соблюдается баланс

$$\gamma_1 + \gamma_2 + \gamma_3 = \sigma_1 + \sigma_2 + \sigma_3.$$

Очереди в этой сети можно рассматривать по отдельности [3], причем число пакетов в блоке  $j=1 \dots 3$  оценивается по формуле (8.2), а именно



$$L_j = \lambda_j / (\mu_j - \lambda_j).$$

Интенсивность  $\lambda_j$  потока на входе каждого блока равна сумме интенсивностей элементарных потоков, поступающих на блок в соответствии с рис.3:  $\lambda_1 = \gamma_1 + \gamma_2$ ,  $\lambda_2 = \gamma_1 + \gamma_2 + \gamma_3$ ,  $\lambda_3 = \gamma_2 + \gamma_3$ .

Можно показать [3], что средняя задержка пакета в сети

$$T = \frac{1}{\gamma} \sum_{j=1}^{j=n} \frac{\lambda_j}{\mu_j - \lambda_j}, \quad (8.4)$$

где  $n$  – число блоков в системе;  $\gamma$  – сумма интенсивностей всех потоков, входящих в систему.

Формула (8.4) верна при следующих предположениях.

- Закон распределения интервалов времени между моментами поступления пакетов  $A(t)$  для отдельных потоков экспоненциальный, причем потоки являются независимыми процессами. Это предположение может быть выполнено на практике.

- Закон распределения времени обслуживания  $B(t)$  также экспоненциальный, причем процессы обслуживания в каждой очереди независимы. Это предположение не может быть выполнено, поскольку время обслуживания пакета пропорционально его длине, и, следовательно, нельзя говорить о независимости времен обслуживания в очередях.

Однако моделирование показывает [3], что применение формулы (8.4) дает приемлемую оценку средней задержки пакета в сети.

Имитационное моделирование позволяет имитировать поведение реальной сети ЭВМ. Имеется много программных средств для имитационного моделирования компьютерных сетей (GPSS, COMNET III фирмы Caci Products Co., BONEs Designer фирмы Cadence Inc., OPNET фирмы Modeler Mil3 Inc., ns2 и др.).

### 8.2.2. Расчет параметров сети в пакетном режиме

Для расчета параметров сети для пакетного режима обслуживания запросов используется теория РСеМО [13]. Предполагаем, что в результате анализа бизнес-процессов предприятия выявлен состав пользователей сети и состав приложений (сервисов). Для того чтобы обеспечить требуемое качество обслуживания (QoS) для различных категорий пользователей, распределим множество  $H$  всех пользователей по типам. Будем считать, что пользователи типа  $t \in T$ , где  $T$  – множество типов, характеризуются следующими параметрами:

$Q_{ts}$  [Кбайт] – средний объем информации, который необходимо передать по сети для работы пользователя типа  $t$  с сервисом  $s \in S$ , где  $S$  – множество сервисов, на интервале  $\Delta t$  [час];

$L_{ts}$  [Кбайт] – средний объем транзакции пользователя типа  $t$  по сервису  $s$ ;

$\tau_{ts.don}$  [с] – допустимое время реакции сети для пользователя типа  $t$  по сервису  $s \in S$ .

Предполагаем, что на одном сервере (компьютере) может быть реализован один или несколько сервисов. Все пользователи распределены по рабочим группам, причем пользователи группы размещаются достаточно компактно: в одном или смежных помещениях размещается множество пользователей  $H_g = \bigcup_{t \in T} H_{tg}$  группы  $g \in G$ , где  $G$  – множество рабочих групп,  $H_{tg}$  – множество пользователей типа  $t$  в группе  $g$ .

Для того чтобы обеспечить требуемое качество обслуживания пользователей, необходимо подобрать параметры оборудования сети таким образом, чтобы они соответствовали информационной нагрузке всех типов пользователей всех рабочих групп.

Пусть  $E$  – множество элементов сетевого оборудования и  $E_{gs} \subset E$  – подмножество элементов, участвующих в передаче данных между сервисом  $s \in S$  и пользователями из группы  $g \in G$ .

Тогда, интенсивность трафика пользователя типа  $t$  по сервису  $s$  на интервале  $\Delta t$

$$\lambda_{ts} = \frac{8 * 10^3 * Q_{ts}}{3600 * \Delta t} [\text{бум} / \text{с}]. \quad (8.5)$$

Интенсивность трафика через элемент  $e$ :

$$\lambda_e = \sum_{g \in G} \sum_{t \in T} \sum_{h \in H_{tg}} \sum_{s \in S} x_{egs} \lambda_{ts}, \quad (8.6)$$

где

$$x_{egs} = \begin{cases} 1, & \text{если } e \in E_{gs}; \\ 0 & \text{в противном случае} \end{cases}$$

Интенсивность трафика сервиса  $s \in S$ :

$$\lambda_s = \sum_{g \in G} \sum_{t \in T} \sum_{h \in H_{tg}} \lambda_{ts}, \quad (8.7)$$

причем интенсивность обслуживания сервера  $\mu_s = \rho_s \lambda_s$ , где  $\rho_s$  – коэффициент загрузки сервера.

### 8.2.3. Расчет производительности серверов

В предыдущем разделе найдена интенсивность обслуживания сервера  $\mu_s$  [бит/с]. Требуется связать значение  $\mu_s$  с такими параметрами сервера, как  $\nu$  – производительность процессора сервера [оп/с],  $V_D$  – скорость передачи на НМД сервера [Кбайт/с] и  $t_D$  – время доступа к НМД сервера [с].

Среднее время обработки пакета процессором сервера можно определить по следующей формуле:

$$1/\mu_{\text{пак}} = \frac{q \cdot K_p}{\nu}, \quad (8.8)$$

где  $q$  – длина пакета в сети (байт);  $K_p$  – коэффициент расширения (число операций, приходящихся на 1 байт информации, берется по данным таблицы 8.1).

Таблица 8.1. Число операций на 1 байт информации (коэффициент пересчета)

Тип задач	Коэффициент пересчета
Учетные	200 ÷ 500
Планирования	1000 ÷ 5000
Прогнозирования	10000 ÷ 20000

Время обработки пакета на магнитном диске сервера складывается из двух составляющих: времени доступа и времени передачи, т.е. может быть определено из выражения:

$$1/\mu_D = \frac{q}{V_D} + t_D \quad (8.9)$$

где  $V_D$  – скорость передачи НМД,  $t_D$  – время доступа к НМД.

На рис. 8.4 изображена эквивалентная схема сервера, рассматриваемого как СеМО. На этом рисунке  $p$  – вероятность обращения к НМД при обработке входящих пакетов,  $\mu_{\text{пак}}$  – интенсивность обработки пакетов процессором сервера,  $\mu_D$  – интенсивность обработки пакета на магнитном диске.

Время произвольного доступа (random access time) – от 3 до 15 мс. Скорость передачи данных для внутренней зоны диска от 44,2 до 74,5 Мбайт/с и от 60,0 до 111,4 Мбайт/с для внешней зоны.

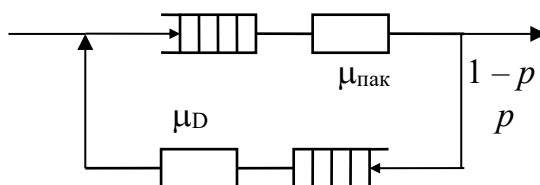


Рис. 8.4. Эквивалентная схема сервера

Схему на рис. 8.4 можно заменить одной СМО с интенсивностью обслуживания  $\mu_p = \mu_s / q$  [пак/с], определяемой формулой:

$$\frac{1}{\mu_p} = \frac{\frac{1}{\mu_{нак}} + p \frac{1}{\mu_D}}{1 - p} \quad (8.10)$$

Уравнение (8.10) получено на основе уравнений баланса с учетом простых свойств слияния и разветвления потоков [10]: при слиянии  $n$  потоков заявок с интенсивностями  $\lambda_1, \dots, \lambda_n$  образуется поток, имеющий интенсивность  $\lambda = \lambda_1 + \dots + \lambda_n$ . При ветвлении потока с интенсивностью  $\lambda$  на  $n$  направлений, вероятности перехода заявки в которые равны  $p_1, \dots, p_n$ , образуется  $n$  потоков с интенсивностями  $\lambda p_1, \dots, \lambda p_n$  соответственно.

Подставим  $1/\mu_{нак}$  из формулы (8.8) в (8.10) и выразим требуемую производительность процессора сервера  $V$  через остальные параметры:

$$V = \frac{K_p q}{(1 - p) / \mu_p - p(q / V_D + t_D)} \quad (8.11)$$

### 8.3. Обеспечение информационной безопасности

Безопасная система – это система, которая, во-первых, надежно хранит информацию и всегда готова предоставить ее своим пользователям, а во-вторых, система, которая защищает эти данные от несанкционированного доступа (<http://www.citfotum.ru/nets/spsmp>).

При проектировании корпоративной сети следует рассмотреть расширенный набор технических средств обеспечения информационной безопасности, например,:

1. Системы контроля доступа, включающие средства аутентификации и авторизации пользователей.
2. Средства аудита.
3. Системы шифрования информации.
4. Системы цифровой подписи, используемые для аутентификации документов.
5. Средства доказательства целостности документов (использующие, например, дайджест-функции).
6. Системы антивирусной защиты
7. Пакетная фильтрация в маршрутизаторах.
8. Пакетная фильтрация в межсетевых экранах.
9. Сервисы-посредники (proxyservices).

Из расширенного набора следует отобрать оптимальный состав средств обеспечения информационной безопасности для конкретной проектируемой системы.

### 8.4. Применение стандартов при проектировании корпоративных сетей

Стандартизация играет ключевую роль в создании и внедрении инфо-телекоммуникационных систем различных уровней и назначений. Основы стандартизации в области вычислительных сетей и средств телекоммуникаций были рассмотрены в первом разделе. Современные информационные системы (ИС) основаны на интеграции информационных, вычислительных и телекоммуникационных ресурсов. Эффективная интеграция современных информационных систем (ИС) – это сложная, комплексная межатраслевая проблема, которая решается на основе методов функциональной стандартизации и применении технологии открытых ИС. На таких ИС базируется информатизация всех сфер современного общества России: органов государственного управления, финансово-кредитной сферы, информационного обслуживания предпринимательской деятельности, производства сельского хозяйства, науки, медицины, образования и т. д.

Развитие и распространение технологии открытых ИС неразрывно связаны с применением функциональных стандартов. Методы функциональной стандартизации позволяют идентифицировать профили, т.е. группы базовых стандартов для выполнения функций, реализуемых конкретными ИС в разных предметных областях деятельности.

Профиль – это совокупность нескольких базовых стандартов (или подмножество одного базового стандарта) с четко определенными и гармонизированными (согласованными) подмножествами обязательных и факультативных возможностей, предназначенная для реализации заданной функции или группы функций ИС.

Технология открытых систем является основой инфраструктуры всех уровней – от уровня предприятия до уровня отрасли и национальной информационной инфраструктуры. Эта технология обеспечивает интеграцию с мировым информационным пространством и с мировой экономикой. Суть технологии открытых систем состоит в использовании стандартных интерфейсов между разнородными аппаратными и программными компонентами систем, в стандартизации и сертификации ИТ.

Существуют следующие виды стандартов в области ИТ: международные стандарты; национальные стандарты; стандарты специальных объединений и комитетов и стандарты отдельных фирм. В Российской Федерации действует около 500 стандартов области ИТ, которые разделяются на межгосударственные (ГОСТ) и государственные стандарты (ГОСТ Р).

Стандарты на кабельные системы зданий. При разработке локальных (ЛВС) и корпоративных вычислительных сетей (КВС) необходимо ориентироваться, прежде всего, на международные стандарты на проектирование кабельных систем поскольку в настоящее время не существует российских стандартов на структурированные кабельные системы зданий. Существует три группы международных стандартов на кабельные системы зданий: американские, международные и европейские. Поскольку за основу были взяты американские требования, эти стандарты различия, в основном, в терминологии.

Отметим четыре основных стандарта: TIA/EIA 568A – стандарт телекоммуникационных кабельных систем коммерческих зданий; TIA/EIA 569 – стандарты прокладки телекоммуникационных каналов коммерческих зданий; TIA/EIA 606 – стандарт администрирования телекоммуникационной инфраструктуры коммерческих зданий; TIA/EIA 607 – требования по заземлению и электрическим соединениям телекоммуникационных систем коммерческих зданий.

### **Практическое занятие «Конфигурирование сети»**

Цель занятия: изучение инструментов конфигурирования сети в UNIX, включающих настройку параметров TCP/IP-сети.

Начальные условия: Командная строка суперпользователя после входа в систему.

Получить сведения обо всех настроенных сетевых интерфейсах с помощью команды `ifconfig -a`:

```
desktop ~ # ifconfig -a
```

```
eth0  Link encap:Ethernet HWaddr 00:0D:60:8D:42:AA
      inet addr:192.168.1.5 Bcast:192.168.1.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:204779 errors:0 dropped:0 overruns:0 frame:0
      TX packets:107606 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:302429520 (288.4 Mb) TX bytes:9177476 (8.7 Mb)
      Base address:0x8000 Memory:c0220000-c0240000
```

```
lo    Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      UP LOOPBACK RUNNING MTU:16436 Metric:1
      RX packets:228 errors:0 dropped:0 overruns:0 frame:0
      TX packets:228 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:17724 (17.3 Kb) TX bytes:17724 (17.3 Kb)
```

Проверить возможность соединения с локальной машиной с помощью команды `ping 127.0.0.1`.

```
desktop ~ # ping 127.0.0.1
```

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
```

```
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.052 ms
```

64 bytes from 127.0.0.1: icmp\_seq=2 ttl=64 time=0.051 ms

64 bytes from 127.0.0.1: icmp\_seq=3 ttl=64 time=0.055 ms

Перед конфигурированием интерфейса eth0 необходимо убедиться, что он отключен. Отключение сетевого интерфейса eth0 производится командой `ifconfig eth0 down`.

```
desktop ~ # ifconfig eth0 down
```

```
desktop ~ # ifconfig -a
```

```
lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      UP LOOPBACK RUNNING  MTU:16436  Metric:1
      RX packets:228 errors:0 dropped:0 overruns:0 frame:0
      TX packets:228 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:17724 (17.3 Kb)  TX bytes:17724 (17.3 Kb)
```

Для связи сетевого интерфейса eth0 с IP-адресом 192.168.1.1 выполним команду `ifconfig eth0 192.168.1.1 up`.

```
desktop ~ # ifconfig eth0 192.168.1.1 up
```

```
desktop ~ # ifconfig -a
```

```
eth0   Link encap:Ethernet  HWaddr 00:0C:F1:2E:0E:F9
      inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
      UP BROADCAST MULTICAST  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
      Interrupt:11 Base address:0x2000 Memory:c0210000-c0210fff
```

```
lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      UP LOOPBACK RUNNING  MTU:16436  Metric:1
      RX packets:228 errors:0 dropped:0 overruns:0 frame:0
      TX packets:228 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
```

RX bytes:17724 (17.3 Kb) TX bytes:17724 (17.3 Kb)

При этом по умолчанию используется сеть класса C, т.е. маска сети «255.255.255.0».

Для задания специфической маски подсети используется параметр `netmask`. Например, данная команда задаёт параметры сети класса A: `ifconfig eth0 10.10.1.1 netmask 255.0.0.0 up`.

```
desktop ~ # ifconfig eth0 10.10.1.1 netmask 255.0.0.0 up
```

```
desktop ~ # ifconfig -a
```

```
eth0    Link encap:Ethernet HWaddr 00:0C:F1:2E:0E:F9
        inet addr:10.10.1.1 Bcast:10.255.255.255 Mask:255.0.0.0
        UP BROADCAST MULTICAST MTU:1500 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
        Interrupt:11 Base address:0x2000 Memory:c0210000-c0210fff
```

```
lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:16436 Metric:1
        RX packets:228 errors:0 dropped:0 overruns:0 frame:0
        TX packets:228 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:17724 (17.3 Kb) TX bytes:17724 (17.3 Kb)
```

С помощью команды `arp` можно узнать текущую ARP-таблицу операционной системы (соответствие MAC-адресов канального уровня IP-адресам). Таблица автоматически поддерживается операционной системой в процессе сетевого обмена.

```
desktop ~ # arp
```

Address	HWtype	HWaddress	Flags	Mask	Iface
gate.localnet	ether	00:02:44:8F:16:B7	C		eth0



## Сценарий: Настройка таблицы маршрутизации

В сценарии производится изучение и настройка таблицы маршрутизации IP. С помощью специальной программы производится изучение маршрута следования пакетов.

Начальные условия: Командная строка суперпользователя, сетевой интерфейс настроен на статический IP-адрес.

Для просмотра таблицы маршрутизации воспользуемся командой `route -n`:

```
desktop ~ # route -n
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	10.10.1.254	0.0.0.0	UG	0	0	0	eth0

Без использования ключа `-n` для всех имён будут использоваться символные значения: `route`

```
desktop ~ # route
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
localnet	*	255.0.0.0	U	0	0	0	eth0
loopback	*	255.0.0.0	U	0	0	0	lo
default	gate.localnet	0.0.0.0	UG	0	0	0	eth0

Для добавления новой строки в таблицу нужно воспользоваться параметром `add`: `route add -host 10.10.2.1 dev eth0`.

```
desktop ~ # route add -host 10.10.2.1 dev eth0
```

```
desktop ~ # route -n
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.10.2.1	0.0.0.0	255.255.255.255	UH	0	0	0	eth0
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	10.10.1.254	0.0.0.0	UG	0	0	0	eth0

Эта команда добавляет явный маршрут до отдельного хоста с указанным IP-адресом через интерфейс eth0.

Аналогичным образом маршрут удаляется, используется параметр `del`: `route del -host 10.10.2.1`.

```
desktop ~ # route add -host 10.10.2.1 dev eth0
```

```
desktop ~ # route -n
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	10.10.1.254	0.0.0.0	UG	0	0	0	eth0

В качестве назначения маршрута можно указывать также целую сеть (параметр `-net`). Рассмотрим команду, которая задаёт маршрут в сеть «192.168.1.0» через шлюз «10.10.1.253»: `route add -net 192.168.1.0 gw 10.10.1.253`.

```
desktop ~ # route add -net 192.168.1.0 gw 10.10.1.253
```

```
desktop ~ # route -n
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.1.0	10.10.1.253	255.255.255.255	UG	0	0	0	eth0
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	10.10.1.254	0.0.0.0	UG	0	0	0	eth0

Рассмотрим простой маршрут движения пакетов до хоста в Internet с помощью программы `tracert` [ya.ru](http://ya.ru):

```
desktop ~ # tracert ya.ru
```

tracert to ya.ru (213.180.204.8), 64 hops max, 40 byte packets

```
1 10.10.1.254 (10.10.1.254) 3.418 ms 2.67 ms 0.719 ms
2 cs7206.rinet.ru (195.54.192.28) 1.34 ms 1.378 ms 0.647 ms
3 ix2-m9.yandex.net (193.232.244.93) 1.554 ms 1.457 ms 1.420 ms
4 c3-vlan4.yandex.net (213.180.210.146) 2.137 ms 2.154 ms 1.842 ms
5 ya.ru (213.180.204.8) 2.646 ms 2.183 ms 2.220 ms
```

## Сценарий: Изучение службы доменных имён

Сценарий посвящен изучению службы доменных имён — её использованию и конфигурированию.

Начальные условия: Командная строка суперпользователя, сетевой интерфейс настроен на статический IP-адрес.

Посмотрим содержимое файла `/etc/hosts`, содержащего имена локальных хостов: `cat /etc/hosts`

```
desktop ~ # cat /etc/hosts
#
# hosts      This file describes a number of hostname-to-address
#            mappings for the TCP/IP subsystem.  It is mostly
#            used at boot time, when no name servers are running.
#            On small systems, this file can be used instead of a
#            "named" name server.  Just add the names, addresses
#            and any aliases to this file...
#
# Localhost
127.0.0.1    localhost

# Home LAN
10.10.1.254  gate.localnet gate
10.10.1.20   boss.localnet boss
```

Проверим работоспособность DNS с помощью команды обращения к хосту в Internet по имени `ping ya.ru`:

```
desktop ~ # ping ya.ru
PING ya.ru (213.180.204.8) 56(84) bytes of data.
64 bytes from ya.ru (213.180.204.8): icmp_seq=1 ttl=54 time=3.56 ms
64 bytes from ya.ru (213.180.204.8): icmp_seq=2 ttl=54 time=2.22 ms
```

Для корректной работы службы доменных имен необходимо прописать используемые серверы DNS в файле `/etc/resolv.conf`. Посмотрим его содержимое `cat /etc/resolv.conf`:

```
desktop ~ # cat /etc/resolv.conf
domain localnet
nameserver 10.10.1.17
```

С помощью команды `host ya.ru` узнаем информацию DNS о хосте в Internet:

```
desktop ~ # host ya.ru
ya.ru has address 213.180.204.8
ya.ru mail is handled by 10 cmail.yandex.ru.
```

Вторым аргументом команды `host ya.ru ns1.yandex.ru` имя DNS-сервера, с которого необходимо получить информацию:

```
desktop ~ # host ya.ru ns1.yandex.ru
ya.ru has address 213.180.204.8
Using domain server:
Name: ns1.yandex.ru
Address: 213.180.193.1#53
Aliases:
```

```
Using domain server:
Name: ns1.yandex.ru
Address: 213.180.193.1#53
Aliases:
```

```
ya.ru mail is handled by 10 cmail.yandex.ru.
```

### **Сценарий: Простая диагностика работы сети**

Сценарий рассматривает самые простые способы диагностики работы сети.

Начальные условия: Командная строка суперпользователя, сетевой интерфейс настроен на статический IP-адрес.

Для проверки работоспособности сетевых служб воспользуемся командой удалённого терминала: `telnet ya.ru 80`. В данном случае будет установлено соединение с хостом в Internet по порту 80 (HTTP):

```
desktop ~ # telnet ya.ru 80
```

```
Trying 213.180.204.8...
```

```
Connected to ya.ru.
```

```
Escape character is '^['.
```

```
GET / HTTP/1.0
```

```
HTTP/1.0 200 OK
```

```
Server: thttpd/2.25b 29dec2003
```

```
Content-Type: text/html; charset=windows-1251
```

```
Date: Wed, 23 Nov 2005 05:40:33 GMT
```

```
Last-Modified: Mon, 07 Nov 2005 15:13:14 GMT
```

```
Accept-Ranges: bytes
```

```
Connection: close
```

```
Content-Length: 2005
```

```
<html>
```

```
<head>
```

```
...
```

Если во время соединения с удалённым узлом ввести команду `netstat -t`, то можно увидеть, что состояние этого соединения – «ESTABLISHED»:

```
desktop ~ # netstat -t
```

```
Active Internet connections (servers and established)
```

```
tcp    0    0 desktop:42639      ya.ru:http        ESTABLISHED
```

Информацию обо всех соединениях в системе можно получить с помощью команды `netstat -a`. В этом случае будет выводиться информация обо всех TCP-, UDP- и локальных сокетах:

```
desktop ~ # netstat -a
```

```
Active Internet connections (servers and established)
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	*:32769	*.*	LISTEN
tcp	0	0	*:32770	*.*	LISTEN
tcp	0	0	*:sunrpc	*.*	LISTEN
tcp	0	0	*:ssh	*.*	LISTEN

```

tcp    0    0 desktop:42639      ya.ru:http         ESTABLISHED
udp    0    0 *:32768            *:.*
udp    0    0 *:32769            *:.*
udp    0    0 *:sunrpc            *:.*

```

Active UNIX domain sockets (servers and established)

```

Proto RefCnt Flags   Type       State      I-Node Path
unix  2    [ ACC ]  STREAM    LISTENING  8344  /var/run/acpid.socket
unix  2    [ ACC ]  STREAM    LISTENING  8866  /var/run/sdp

```

...

## Сценарий: Работа по удалённому терминалу

Сценарий рассматривает работу по удалённому сетевому терминалу с использованием программы ssh.

Начальные условия: Командная строка суперпользователя, сетевой интерфейс настроен на статический IP-адрес.

С помощью команды ssh user@10.10.1.222

```
desktop ~ # ssh user@10.10.1.222
```

Password:

```
Last login: Sat Nov 21 15:56:20 2005 from 10.10.1.5
```

```
user@remote ~ $
```

```
user@remote ~ $ exit
```

Выполним команду who, чтобы убедиться, что находимся на удалённой машине. Для всех пользователей, работающий удалённо, указывается IP-адрес.

```
user@remote ~ $ who
```

```
user    vc/1      Nov 14 14:04
```

```
user    pts/0    Nov 22 10:55 (10.10.1.5)
```

Для завершения сеанса удалённого терминала нужно выйти из командной оболочки с помощью команды exit.

```
user@remote ~ $ exit
```

```
logout
```

Connection to 10.10.1.5 closed.

desktop ~ #

### Практическое занятие «Сравнение технических характеристик серверов»

Цель работы: научиться сравнивать и подбирать сервера.

Задание.

Сравнить сервера HP, IBM, DELL аналогично примеру. Сделать вывод.

BM System X3550 M3 — это уверенный «середнячок» бренда IBM, с улучшенной версией процессора Intel Xeon, который обладает следующими параметрами:

- максимальное возможное встроенное количество процессоров — 2 шт.;
- количество ядер — до 12;
- кэш — 12 МБ;
- наличие слотов для памяти — 18;
- предельно возможный объем оперативной памяти — 288 ГБ;
- применяемый модуль памяти — DDR3 RDIMM (возможен DDR3 UDIMM);
- количество портов — 1;
- имеющиеся слоты расширений — 2 (тип слота PCI-E 2.0.);
- собственный объем памяти — 7,2 Тб;
- количество используемых дисков с типоразмером 2,5D — до 6 шт.;
- количество используемых блоков питания — до 2 шт.;
- наличие вентиляторов с возможностью оперативной замены в горячем режиме — до 6 шт. (+1 резервный);
- предоставляемый срок гарантии — 3 года.

HP Proliant DL360 G7 — яркий представитель ультратонких HP Blade серверов, который имеет следующие характеристики:

- максимальное возможное встроенное количество процессоров — 2 шт.;
- количество ядер — до 12;
- кэш — 12 МБ;
- наличие слотов для памяти — 18;
- предельно возможный объем оперативной памяти — 384 ГБ;
- применяемый модуль памяти — DDR3 RDIMM (возможен DDR3 UDIMM);
- количество портов — 2;
- имеющиеся слоты расширений — 2 (тип слота PCI-E 2.0.);
- собственный объем памяти — 9,6 Тб;
- количество используемых дисков с типоразмером 2,5D — до 8 шт.;

- количество используемых блоков питания — до 2 шт.;
- наличие вентиляторов с возможностью оперативной замены в горячем режиме — до 6 шт.;
- предоставляемый срок гарантии — 3 года.

Dell PowerEdge R610 — наиболее популярная модель серверов от торговой марки Dell, которая поставляет серверные комплектующие и ИТ-технологии. Сервер Dell PowerEdge R610 имеет такие параметры:

- максимальное возможное встроенное количество процессоров — 2 шт.;
- количество ядер — до 12;
- кэш — 12 МБ;
- наличие слотов для памяти — 18;
- предельно возможный объем оперативной памяти — 192 ГБ;
- применяемый модуль памяти — DDR3 RDIMM (возможен DDR3 UDIMM);
- количество портов — 2;
- имеющиеся слоты расширений — 2 (тип слота PCI-E 2.0.);
- собственный объем памяти — 12 Тб;
- количество используемых дисков с типоразмером 2,5D — до 8 шт.;
- количество используемых блоков питания — до 2 шт.;
- наличие вентиляторов с возможностью оперативной замены в горячем режиме — до 6 шт.;
- предоставляемый срок гарантии — 3 года.

Все перечисленные модели используют схожие программы для удаленного доступа, поэтому управление между ними ничем не отличается.

Из представленных параметров трех серверов с аналогичными характеристиками наиболее привлекательно выглядят серверные расширения от Hewlett Packard, и это только при поверхностном анализе. Если углубиться в расшифровку возможностей, то отставание IBM и Dell еще больше увеличится. Изделия от IBM обладают средними характеристиками, а системы хранения данных от Dell отстают по основным показателям среди представленных моделей.

## **Практическое занятие «Формирование аппаратных требований и схемы банка данных»**

### **Вопросы для обсуждения**

1. Что такое банк данных?
2. Какова структура банка данных?
3. Сформулируйте понятие «база данных».
4. Что представляет собой внемашинаая информационная база проектов?



5. Приведите обобщающие характеристики документов внемашинной информационной базы.
6. Приведите понятие реквизита документа внемашинной информационной базы.
7. Что такое предметная область исследования?
8. Каким образом по характеру возникновения подразделяются документы внемашинной информационной базы?
9. Каким образом по технологии обработки подразделяются внемашинной информационной базы?
10. Каким образом обеспечивается связь документов внемашинной информационной базы?
11. Приведите понятие «реквизит» документа внемашинной информационной базы?
12. Что представляет собой внутримашинная информационная база?

## Практическое занятие «Установка и настройка сервера MySQL»

ЦЕЛЬ: ознакомиться установкой, запуском MySQL Server 5.0, научиться подключаться к серверу баз данных MySQL в локальной сети, управлять учетными записями пользователей, ознакомиться с возможностями программ-утилит MySQL Administrator, Navicat for MySQL и phpMyAdmin по администрированию баз данных. ¶Теоретические сведения

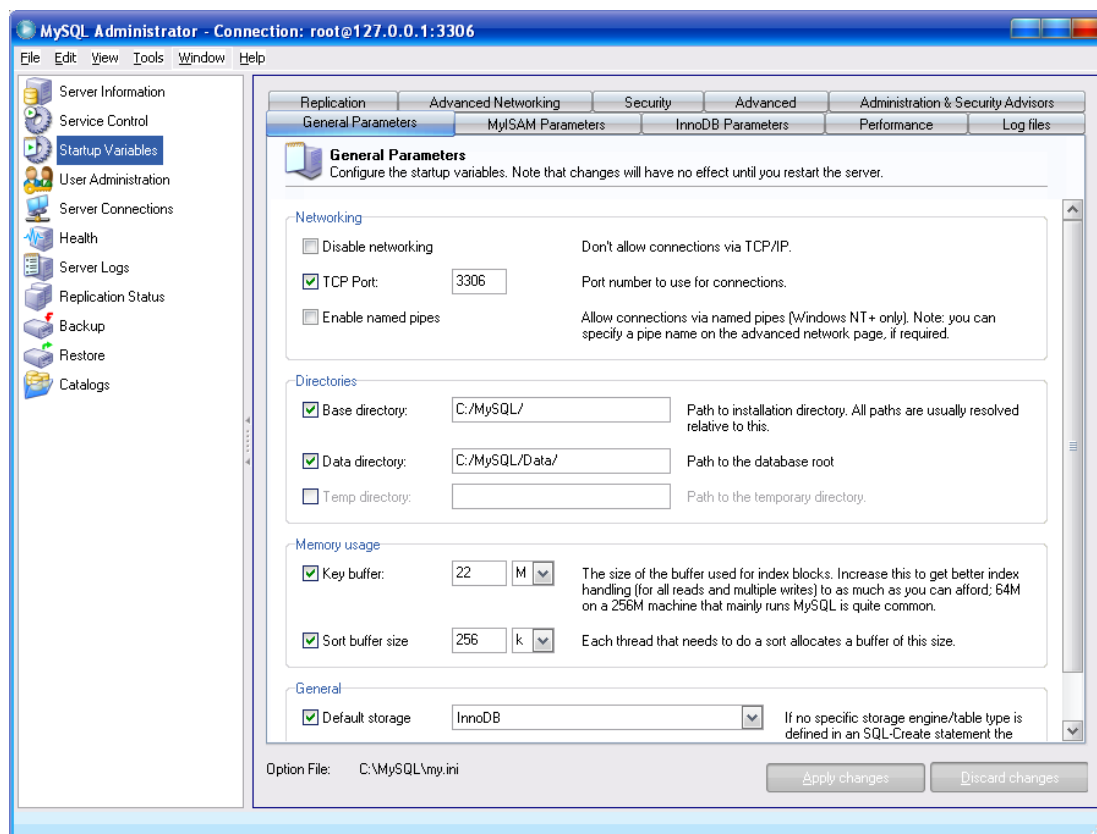
### 1. Вспомогательные утилиты MySQL

Различные вспомогательные утилиты MySQL располагаются в подкаталоге bin корневого каталога MySQL. Далее приведен список наиболее часто используемых утилит:

- `myisampack`— сжимает таблицы типа MyISAM, уменьшая их в размере и делая доступными только для чтения;
- `mysql` — консольный клиент для доступа к MySQL-серверу, позволяет выполнять SQL-запросы и осуществлять администрирование сервера;
- `mysqladmin`— утилита для выполнения административных функций, таких как создание или удаление баз данных, получения информации с сервера о номере версии, процессах, состоянии сервера и т. п.;
- `mysqlbinlog`— данная утилита используется для чтения содержимого журнала двоичной регистрации при восстановлении данных в нештатных ситуациях;
- `mysqlcheck` — утилита, используемая для описания, проверки, оптимизации и восстановления таблиц;
- `mysqldump`— выводит содержимое базы данных MySQL в виде файла с SQL-операторами или в виде текстовых файлов с символом табуляции в качестве разделителя. Такие файлы часто называют дампом (dump) базы данных;
- `mysqlhotcopy` — утилита для создания резервной копии таблиц без остановки сервера MySQL, т. е. создания "горячей" копии базы данных;
- `mysqlimport` — выполняет перенос информации из текстового файла в таблицы базы данных;
- `mysqlshow` — отображает информацию о существующих базах данных, таблицах, полях и индексах.

### 2. Графическая утилита MySQL Administrator

1. Назначение утилиты MySQL Administrator – собственная графическая утилита MySQL полноценного администрирования СУБД.¶

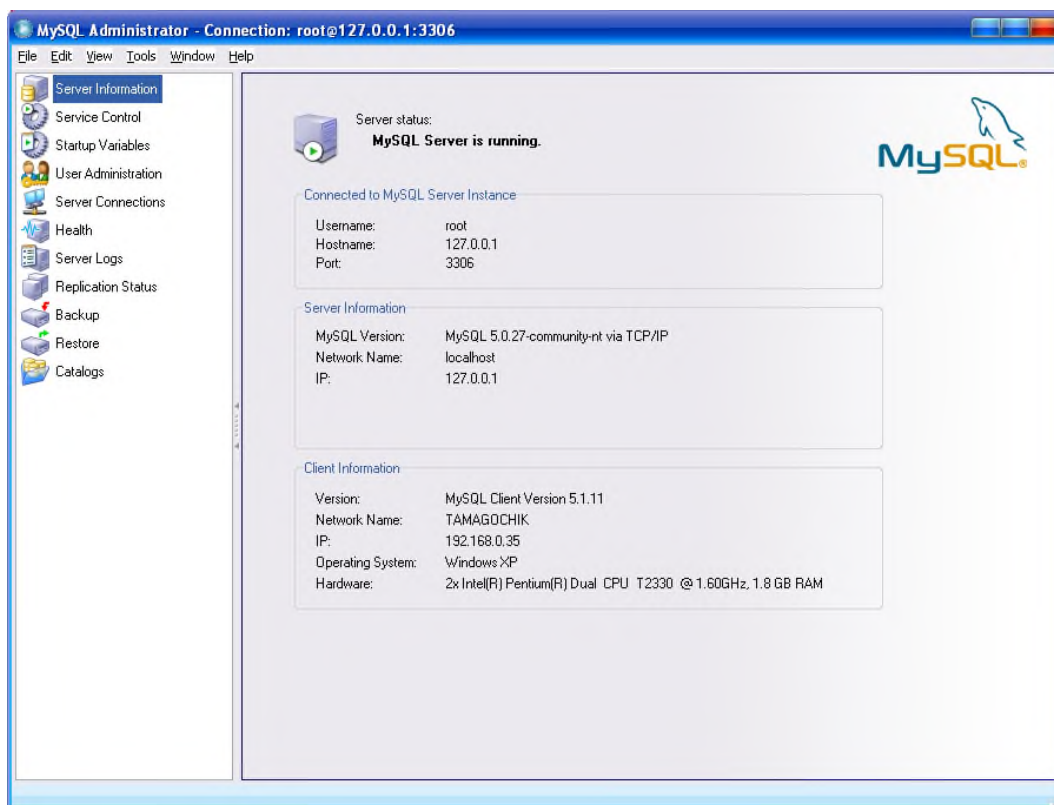


2.

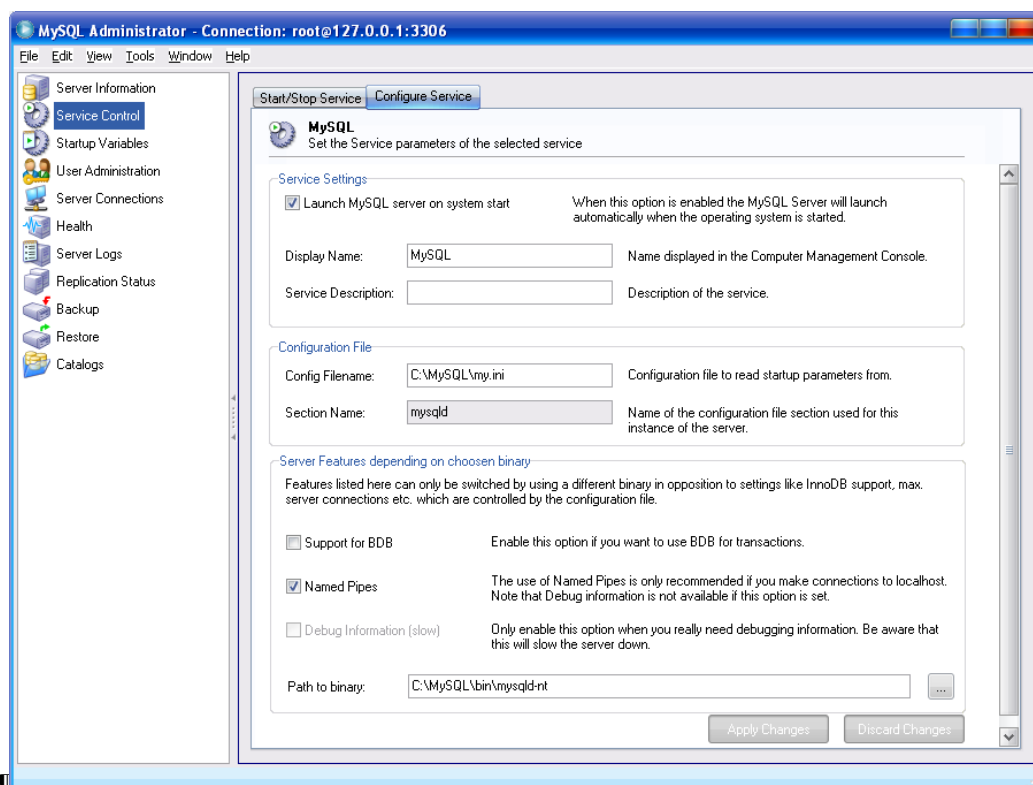
Пакет ^ MySQL Administrator предоставляет продвинутые возможности настройки всех параметров сервера. ¶ Утилита предназначена для администрирования сервера, а также может выполнять операции резервного копирования и восстановления информации (как над отдельными, так и над всеми базами, причем есть встроенный планировщик заданий). Кроме СУБД MySQL поддерживается Oracle или другие БД через ODBC (не проверялось, но понятно, что в таком случае будут отключены все возможности, специфические именно для MySQL, да и программа разрабатывалась главным образом для MySQL, а все остальное – как бесплатный довесок). ¶ Основные функции утилиты сгруппированы в 11 пунктов графического меню в стиле MS Windows 2003. Утилита распространяется на правах OpenSorce GPL и доступна для платформы Win32 и Linux как в бинарном виде, так и в исходных кодах. Текущая версия 1.0.1 alpha работает довольно стабильно, хотя есть некоторые недочеты в интерфейсной части и часть функций не реализована (только интерфейсная часть).

## 2. Меню и главные функции программы

Server Information. Первый пункт стандартен – Server Information. Здесь можно посмотреть на какой платформе запущен сервер, хост/порт, текущий аккаунт пользователя, IP, операционную систему, процессор и размер доступной памяти. Тут же показывается текущий статус сервера – работает или остановлен.¶

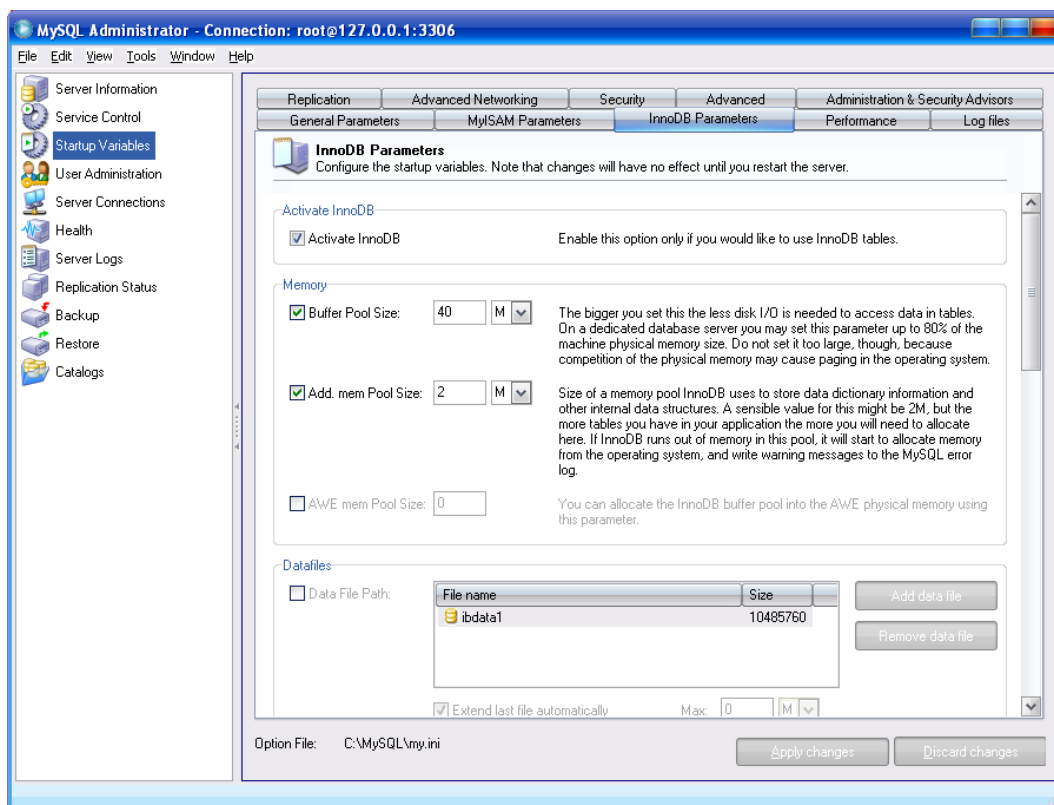


Service Control. Позволяет запустить или перезапустить сервер, просмотреть лог загрузки. Вторая вкладка позволяет настроить основные параметры – пути к файлу конфигурации, директории с бинарными файлами и настроить сервер для поддержки специфических расширений (например, таблиц формата InnoDB с поддержкой транзакций или использования для работы именованных каналов). В разных версиях сервера (mysqld-opt, mysqld-nt, mysqld-max, mysqld-max-nt), кроме различий в производительности, еще и по-разному реализована поддержка расширений (таблицы InnoDB поддерживаются в mysqld-opt, но не поддерживаются

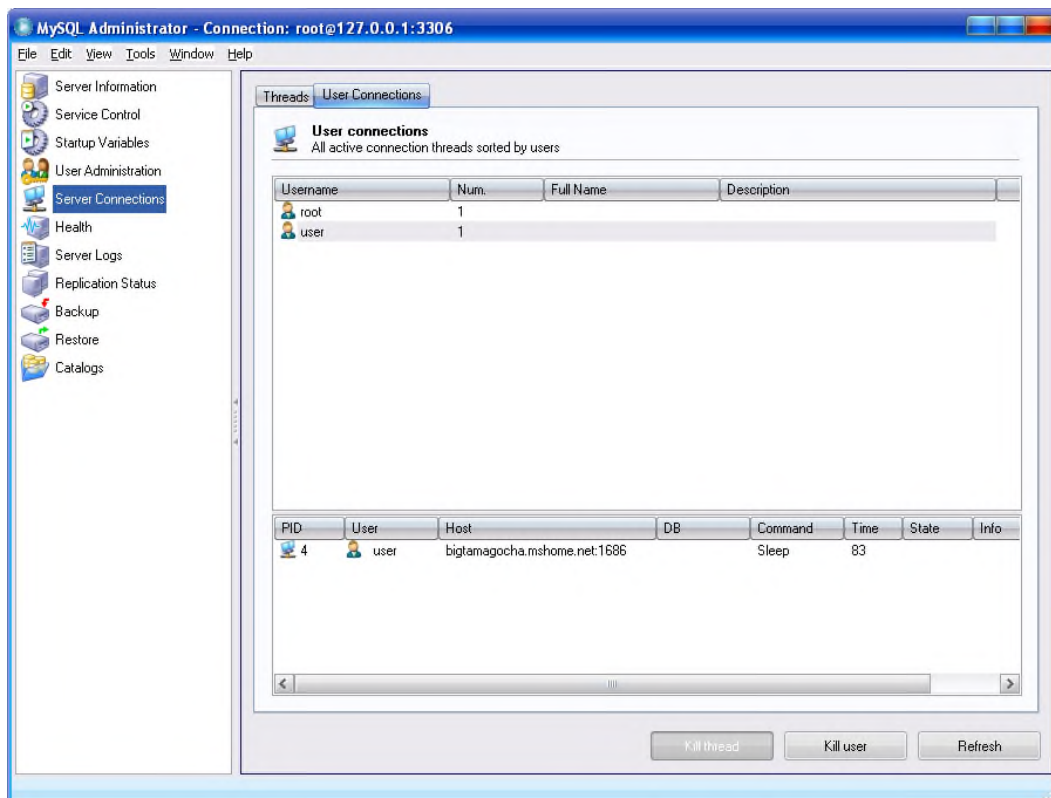


в стандартном mysqld).

Startup Variables. Этот раздел позволяет настроить множество тонких опций, влияющих на производительность сервера. Кроме базовых настроек (поддержка сетевых соединений, настройка портов, пути к служебным каталогам, настройка работы с памятью), отдельно можно настроить опции для различных типов таблиц (MyISAM, InnoDB). Для увеличения производительности есть настройки кеширования, для работы с несколькими серверами есть настройки репликации. Вкладка Advanced позволяет очень тонко настроить работу сервера, а некоторые опции нельзя настроить традиционным путем при помощи командной строки. На отдельных вкладках сгруппированы настройки безопасности и сетевые возможности. Как некоторый недочет - очень небольшое количество настроек безопасности, а ведь при серьезной работе в коммерческих приложениях безопасности уделяется повышенное внимание.¶

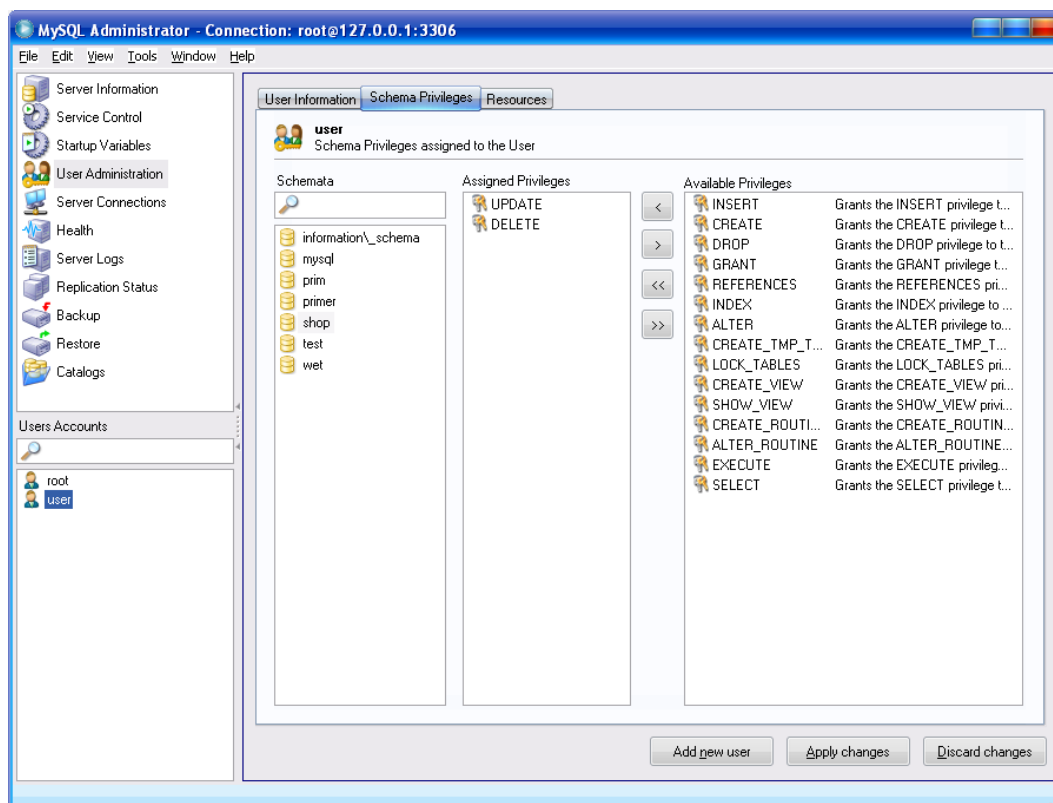


Server Connections¶ В этом меню можно просмотреть все текущие подключения клиентов к серверу и при необходимости завершить любое из них.¶



User

Administration¶Позволяет настроить привилегии для каждого пользователя и оперативно управлять ими (раньше для этого надо было вручную править служебные таблицы MySQL с помощью SQL запросов).



При использовании MySQL Administrator все операции над пользователями проходят в графическом режиме и в любое время за считанные секунды можно настроить привилегии для любого пользователя или базы данных. Эта функция одна из самых полезных и придется по вкусу администраторам и разработчикам.

¶Health¶Меню Health чисто информационное и позволяет отслеживать производительность сервера в

реальном времени, расход памяти и сетевой трафик, а также показывает статистику по типам запросов и эффективность задействования ключей. Эти возможности можно использовать, чтобы наблюдать изменения в работе сервера при изменении настроек, тестировать их влияние на производительность и таким образом постепенно настроить сервер на максимум для конкретного круга задач.

^ Server Logs¶Показывает стандартные логи сервера – общий лог и записи об ошибках.

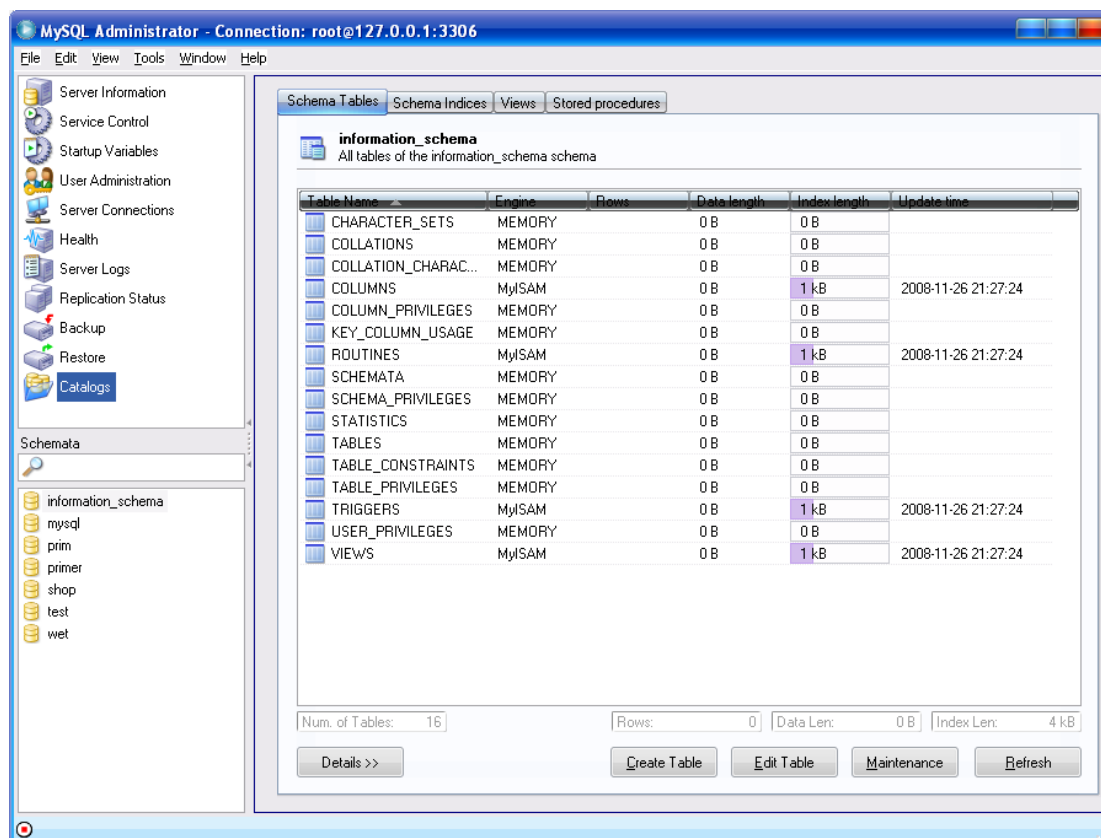
Вот самое интересное и полезное - это реализованные функции резервирования и восстановления БД. Реализован удобный планировщик, и теперь можно делать раздельное резервирование отдельных БД и таблиц – одни БД один раз в сутки, другие в конце недели и т.д. Опции резервирования также гибко настраиваются – для этого есть вкладка Advanced Options. Пока для резервирования доступен лишь один формат - простой текстовый файл с SQL запросами, без сжатия и т.д. Но будем надеяться, что в следующих версиях будет хоть какой-то алгоритм сжатия.

К сожалению, вся вкладка Schedule пока неактивна – видимо, разработчики еще не до конца протестировали эту возможность. Все-таки, этот релиз обозначен как альфа. В любом случае, путь, по которому пошли разработчики не может не радовать – наконец СУБД MySQL обзаводится мощными средствами конфигурирования и настройки и теперь ее можно применять и в корпоративных приложениях, а не только в Web-проектах. ¶

При восстановлении из резервной копии также можно выбрать отдельные таблицы и базы данных, которые подлежат восстановлению.

Replication Status. Функция Replication Status скорее всего еще до конца не реализована, присутствует только вкладка Server Information. Для реализации и настройки репликации надо перейти на вкладку Replication раздела Startup variables, где можно детально настроить параметры репликации. ¶

Catalogs. Последний раздел дает доступ к отдельным базам и таблицам, позволяет осуществлять операции оптимизации, проверки и восстановления таблиц, а также показывает сведения об индексах и их параметрах, а также пользователей, которые имеют доступ к конкретной базе или таблице. ¶



В целом, MySQL Administrator способна обеспечить все потребности администратора или разработчика, как по управлению пользователями, так и по тонкой настройке производительности, анализе работы сервера, и может служить инструментом для резервирования и репликации баз данных. Для полноценного использования, думаю, еще надо подождать следующего релиза, где будут исправлены все недочеты и увеличена функциональность (в основном это касается функций резервирования). ¶ Распространение такого мощного средства на условиях OpenSource будет способствовать продвижению СУБД и теперь можно ожидать, что появятся вполне серьезные разработки корпоративного масштаба, основанные на MySQL. Версия сервера – 5.0 выполняет поддержку хранимых процедур. ¶^

## ***ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ РАБОТЫ:***

### ***Часть 1 – Установка, конфигурирование, запуск MySQL Server 5.0 и XAMPP***

1. ¶ Установить MySQL Server 5.0 (или XAMPP, в составе которого также есть сервер MySQL) на компьютер пользователя (если он установлен).
2. ¶ Проверить состояние (статус) сервера на данный момент, задать настройки запуска сервера MySQL на компьютере пользователя – открыть консоль управления сервисами, выполнив команду Пуск | Настройка | Панель управления | Администрирование | Службы или вызвать окно Управление компьютером. В результате этого откроется окно, представленное на рис. В консоли управления сервисами необходимо найти сервис MySQL. ¶



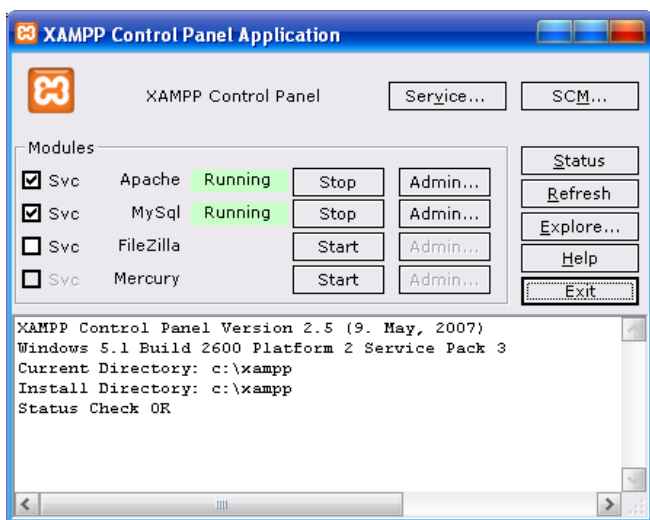
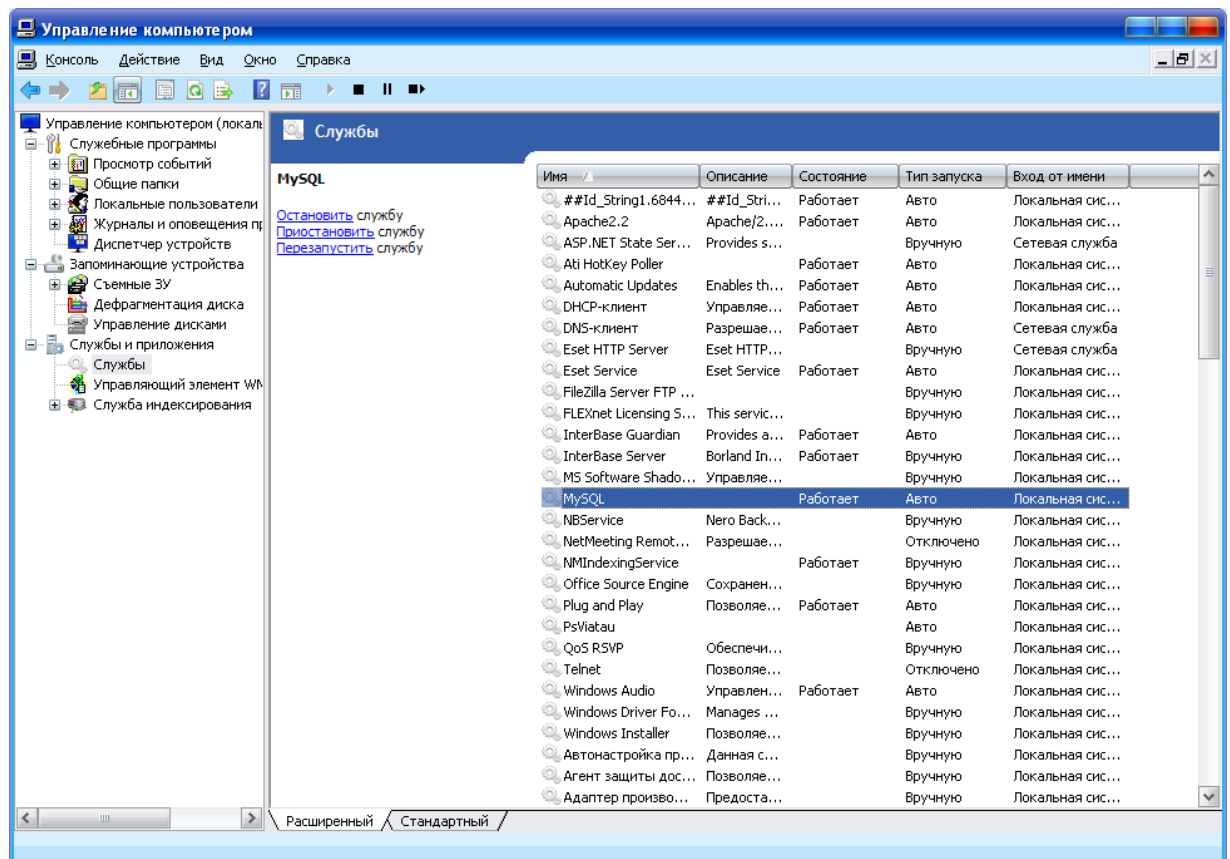
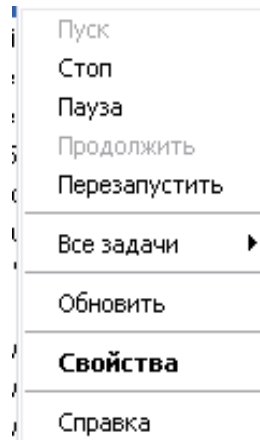


Рис. Консоль управления сервисами



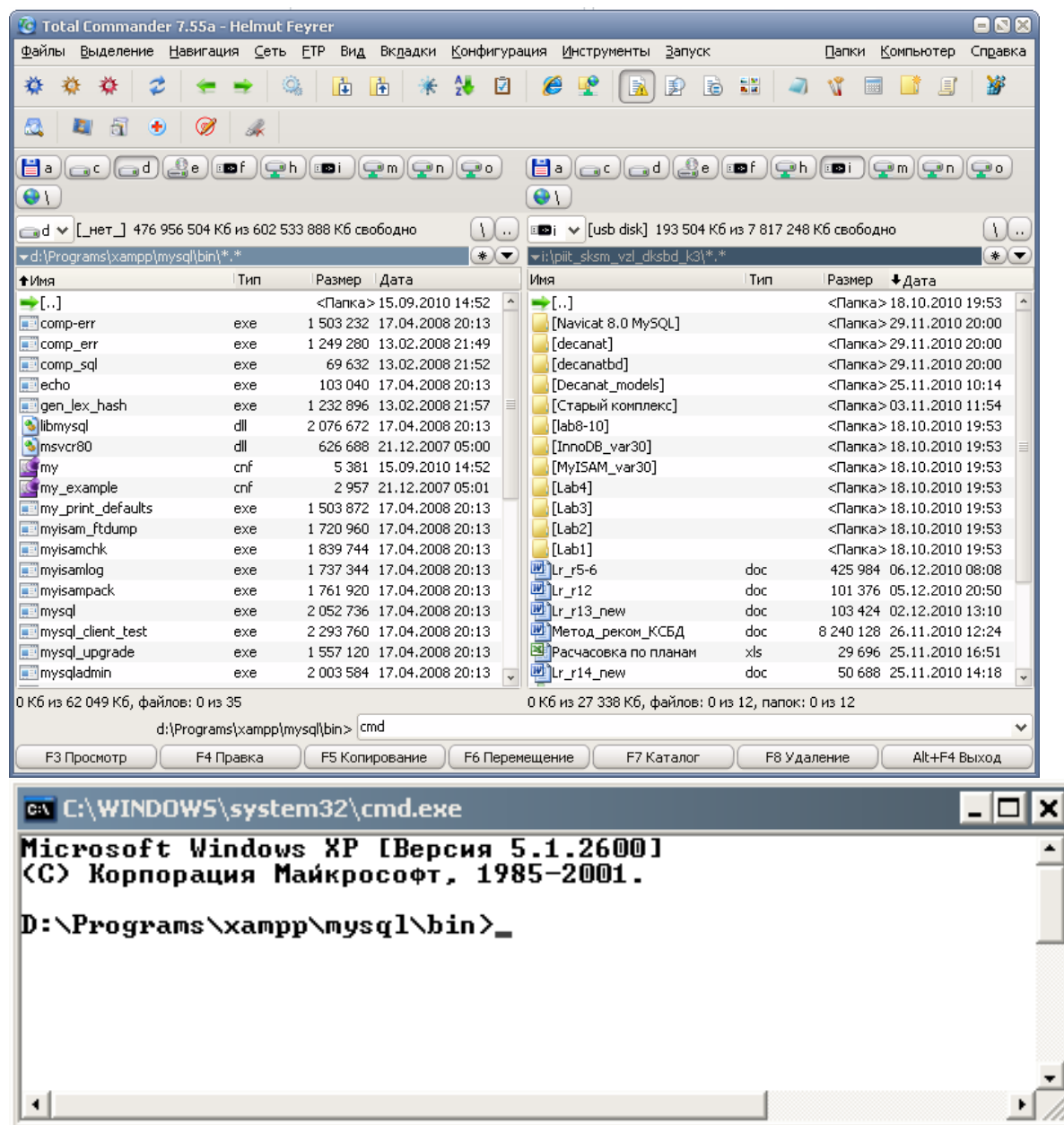
ЗАМЕЧАНИЕ: Если поле

Состояние данного сервиса пусто, то он не запущен.

Для его запуска следует выбрать в Контекстном меню пункт Пуск. Для остановки сервиса необходимо выбрать пункт Стоп. В столбце Тип запуска: значение Авто сообщает Windows о необходимости запускать сервис при старте операционной системы, значение Вручную означает необходимость запуска сервиса пользователем через консоль управления сервисами. Можно изменить режим запуска сервиса, выбрав в контекстном меню сервиса пункт Свойства. Также можно воспользоваться диалоговым окном XAMPP Control Panel для просмотра сведений о состоянии сервера. Если сервер не запущен, то выполнить перезапуск XAMPP (файл xampp\_restart.exe), и в случае успешного перезапуска, проверить состояние сервера, например, в диалоговом окне управляющей панели.

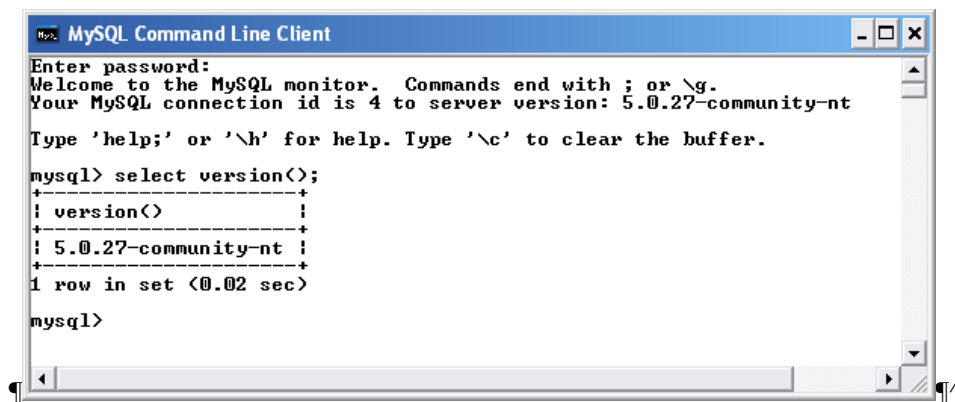
3. ¶Запустить командную строку.

**ЗАМЕЧАНИЕ:** для удобства последующего запуска сервера MySQL рекомендуется для запуска командной строки выполнить следующие действия: запустить программу-надстройку TotalCommander, в одной из панелей войти в папку ...MYSQL\BIN, затем в командной строке внизу экрана программы прописать команду `cmd` и нажать ENTER. При этом каталог BIN в командной строке в приглашении MS-DOS будет сразу установлен рабочим.



4. ¶Получить сведения об IP-адресе компьютера, выполнив команду `ipconfig`. Полученный адрес выписать в тетрадь. Получить сведения об IP-адресах еще 4-х компьютеров в локальной сети и записать.
5. ¶Сделать рабочим каталог ...MYSQL\BIN, если он не является рабочим.
6. ¶Подключиться к локальному серверу MySQL, как пользователь `root`

7. ¶Получить сведения о версии сервера – после того, как появилось приглашение `mysql>`, ввести команду `SELECT VERSION()`;



```
MySQL Command Line Client
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 5.0.27-community-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> select version();
+-----+
| version() |
+-----+
| 5.0.27-community-nt |
+-----+
1 row in set (0.02 sec)

mysql>
```

## *Часть 2 – Работа с утилитой MySQL. Управление учетными записями пользователей: создание, удаление, переименование*

8. ¶Получить сведения о существующих учетных записях на локальном сервере.
9. ¶Создать по указанию преподавателя следующие учетные записи пользователей:
  1. ¶Имя пользователя – фамилия студента, возможность обращения только с локального сервера, задать пароль 123456;
  2. ¶Имя пользователя – user, возможность обращения с любого хоста только в локальной сети той аудитории, где проводится занятие, например, 153;
  3. ¶3 учетные записи, в которых имя пользователя – фамилии студентов группы с номерами мониторов, например, `retrov04` (поочередно), возможность обращения только с хостов данных студентов группы, пароли можно не задавать.

¶ЗАМЕЧАНИЕ: выписать на доску имена пользователей, созданных учетных записей для одnogруппников.

10. ¶Получить сведения о существующих учетных записях на локальном сервере и их привилегиях (грантах).
11. ¶Выполнив выход из-под пользователя `root`, подключиться локально на свой сервер под своей фамилией (авторизировать пользователя).

¶ЗАМЕЧАНИЕ: для захода под паролем, следует ввести команду следующего шаблона¶`... \MYSQL\BIN> mysql -u <имя пользователя> -p¶Имя реального пользователя в угловые скобки не брать!`

12. ¶Выполнить выход.
13. ¶Подключиться, как `user` к любому хосту в локальной сети 153.

¶ЗАМЕЧАНИЕ: для захода под паролем, следует ввести команду следующего

шаблона¶...\\MYSQL\\BIN> mysql -u <имя пользователя> --host= ¶IP-адрес ввести в виде, полученном по команде ipconfig¶Дописать в конце команды -p (если есть пароль на вход).

14. ¶Выполнить выход.

15. ¶Подключиться к серверу на хосте студента группы, который зарегистрировал Вас.

Выполнить выход.

16. ¶Авторизировать пользователя root.

17. ¶Переименовать учетную запись user в superuser.

¶ЗАМЕЧАНИЕ: задать в команде старое имя пользователя с указанием хоста.

18. ¶Удалить учетную запись с фамилией студента на локальном сервере.

¶ЗАМЕЧАНИЕ: задать в команде имя пользователя с указанием хоста.

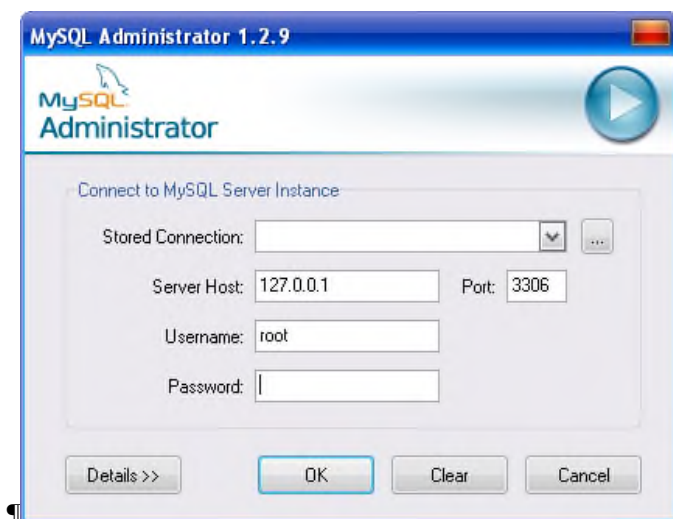
19. ¶Завершить работу с окном клиента.

¶^ Часть 3 – Работа с графическими программами-утилитами MySQL Administrator, Navicat 8 for MySQL и phpMyAdmin

20. ¶Установить утилиту MySQL Administrator на компьютер пользователя в папку, с установленным сервером MySQL, если она не установлена.

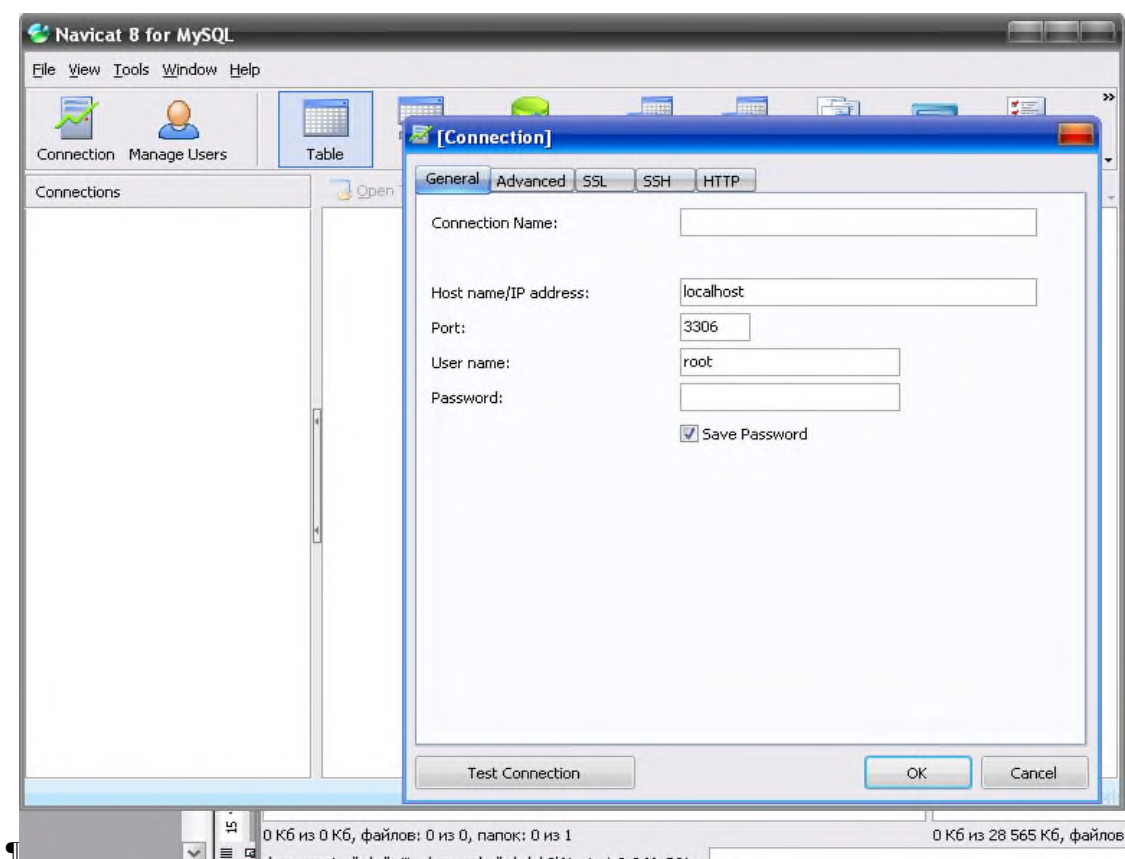
21. ¶Скопировать в папку Data учебную базу с заданным именем по указанию преподавателя.

22. ¶Подключиться к серверу локально, как пользователь root

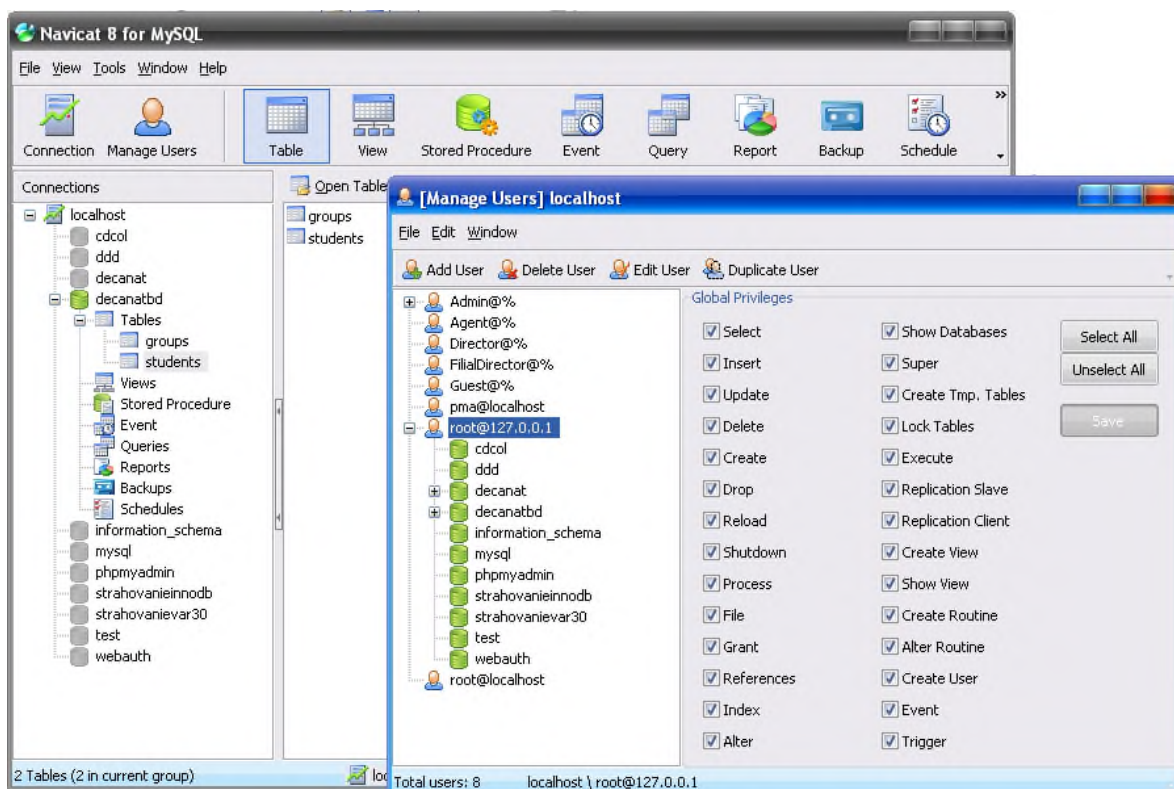


23. ¶Ознакомиться с экраном программы, пунктами графического меню. Просмотреть учетные записи пользователей, привилегии, а также содержимое таблиц, выложенных на сервере.

24. ¶Выполнить выход File \ Reconnect и подключиться к серверу на хосте студента группы, который зарегистрировал Вас. Ознакомиться с его учетными записями. Выполнить выход.
25. ¶Запустить программу-утилиту Navicat for MySQL (если она не установлена, то можно запустить, не устанавливая, локально из папки по указанию преподавателя).
26. ¶Подключиться к серверу локально, как пользователь root, нажав на кнопку Connection и заполнив поля на вкладке General. Можно также выполнить тест подключения (кнопка Test Connection).



27. ¶Ознакомиться с экраном программы, пунктами графического меню. Просмотреть учетные записи пользователей (кнопка Manage Users), привилегии, а также содержимое таблиц, выложенных на сервере.



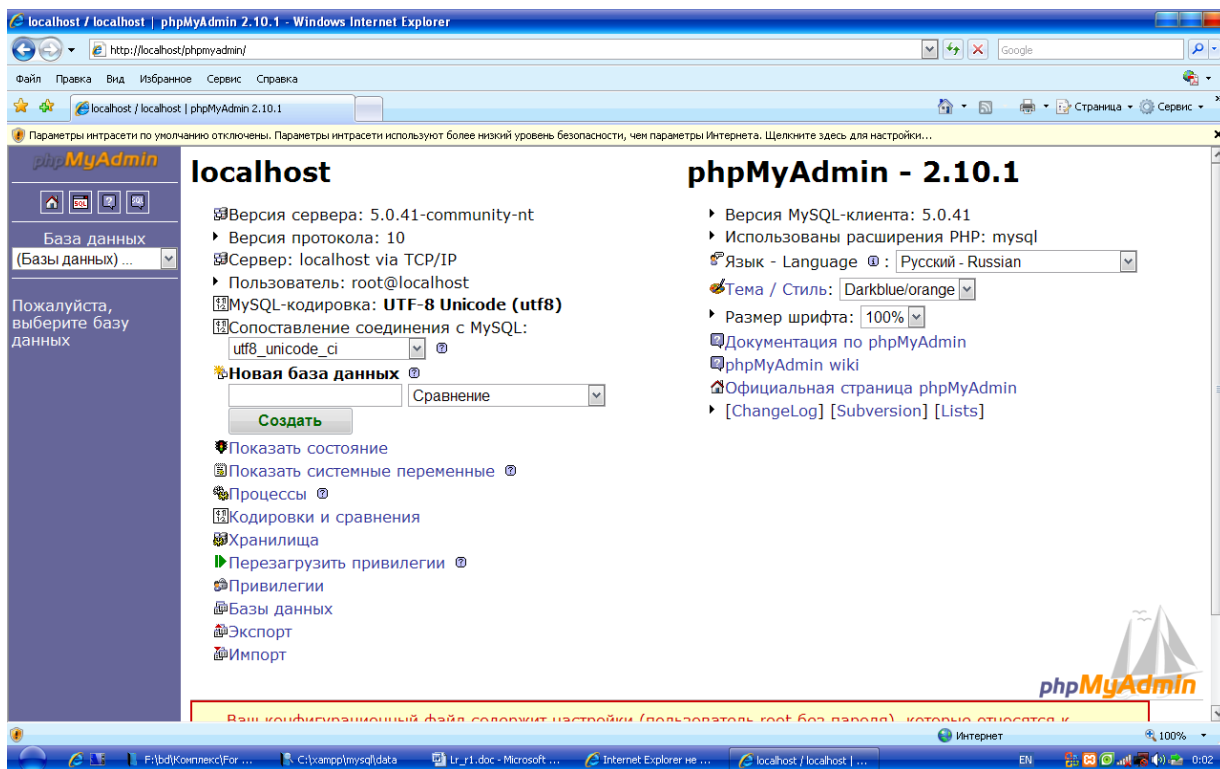
28. ¶Выполнить выход File \ Close Connection и подключиться к серверу на хосте студента группы, который зарегистрировал Вас (кнопка Connection). Ознакомиться с его учетными записями. Выполнить выход (File \ Close).

29. ¶Запустить браузер и в строке адреса ввести localhost

¶ЗАМЕЧАНИЕ: предварительно убедиться, что сервер Apache (в диалоговом окне XAMPP) запущен.

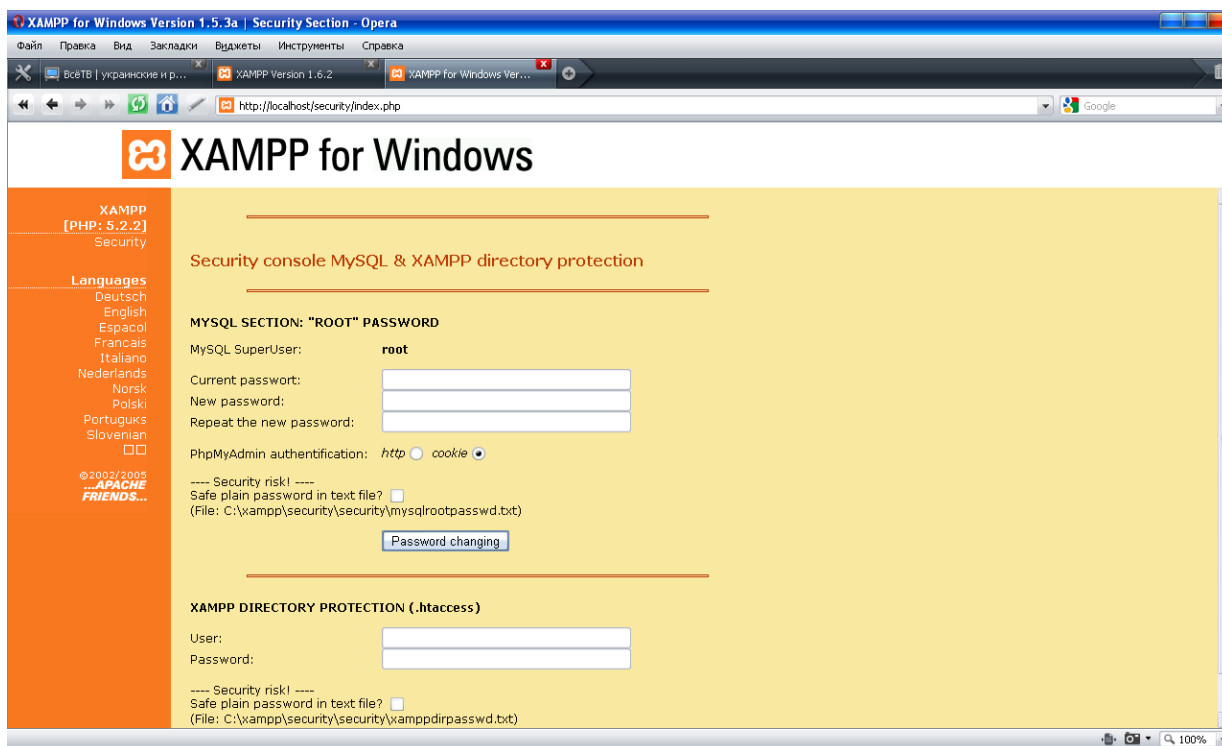
30. ¶Выбрать при первом запуске английский язык, перейти по ссылке [phpMyAdmin](#)

31. ¶Аналогично ознакомиться с окном, пунктами графического меню и т.д.¶



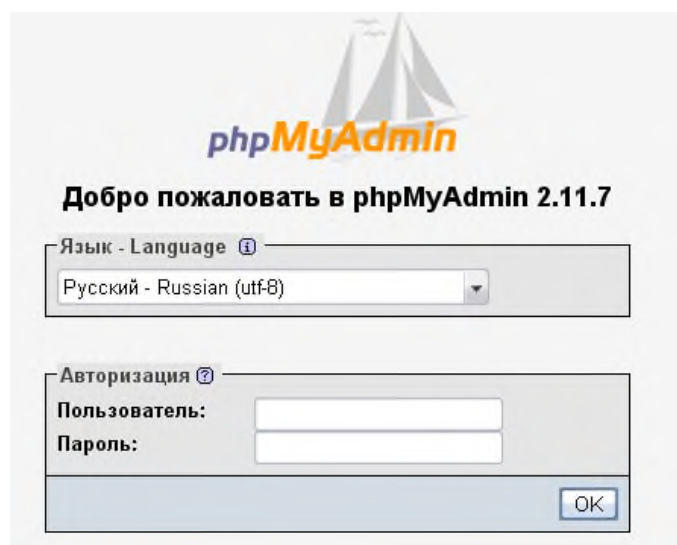
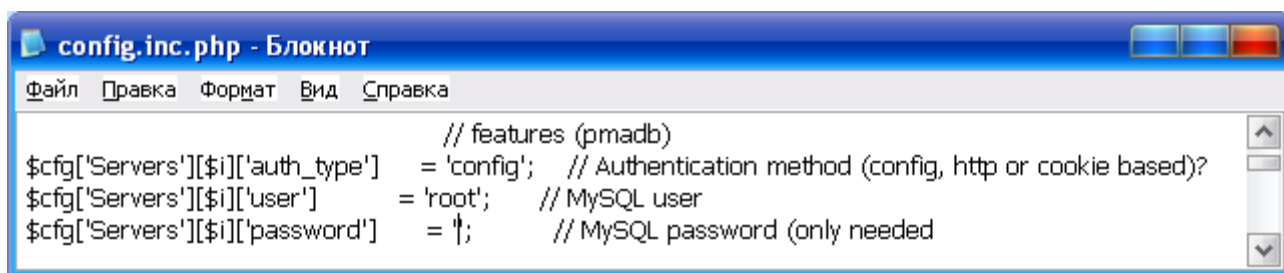
¶**ЗАМЕЧАНИЕ:** обратите внимание, что при подключении к любому хосту в локальной сети (например, если прописать в строке адреса в браузере 192.168.153.102), происходит подключение к серверу на заданном хосте под именем root, что автоматически дает полный доступ с Вашего ПК к базам данных сервера (как администратора). ¶На странице загрузки будет выведено соответствующее сообщение и рекомендация о желательной смене пароля пользователя root с заданного по умолчанию на другой. В реальных рабочих условиях это необходимо, но в учебных целях пароль (пустая строка) следует оставить без изменения. ¶Для корректной замены пароля следует зайти в главном окне XAMPP в левой панели по ссылке Security и в появившемся окне выбрать ссылку <http://localhost/security/xamppsecurity.php>, затем поменять пароль. ¶





Если

выполнить смену пароля иначе, например, исправить учетную запись на странице Привилегии, то в дальнейшем следует в файле конфигурации <путь к каталогу>\xampp\phpMyAdmin\config.inc.php пароль изменить вручную в соответствующей строке (на рисунке – это строка, где установлен текстовый курсор ).



Если пароль на root был изменен, то нужно помнить,

что новый пароль будет действителен во всех утилитах при обращении к серверу.

32. Для создания диалогового окна входа нужно в файле config.inc.php дописать следующие строки, если их нет: `$cfg['Servers'][$i]['auth_type'] = 'cookie';` `$cfg['Servers'][$i]['host'] = 'localhost';` Строки из предыдущего



диалогового окна закомментировать. Вместо localhost можно указать IP-адрес нужного сервера. Предварительно создать учетные записи.

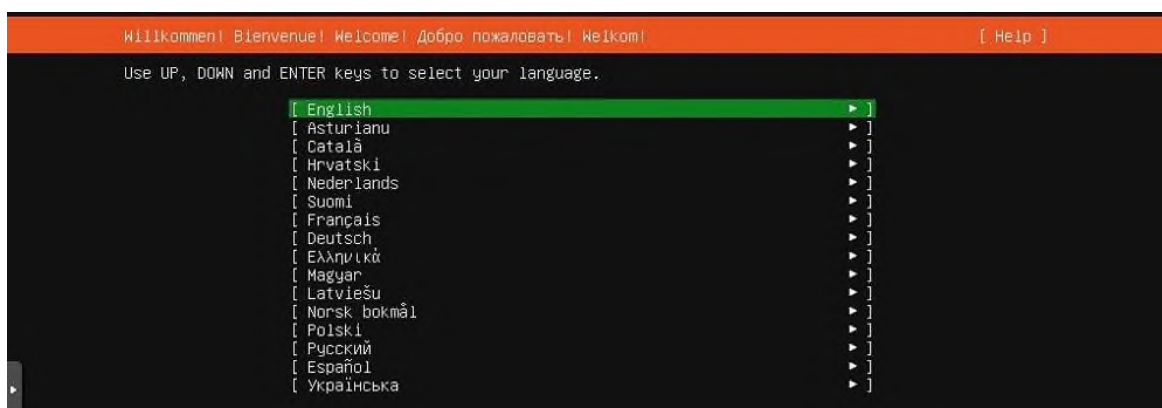
32. ¶Оформить отчет.

### Практическое занятие «Установка и настройка сервера под UNIX»

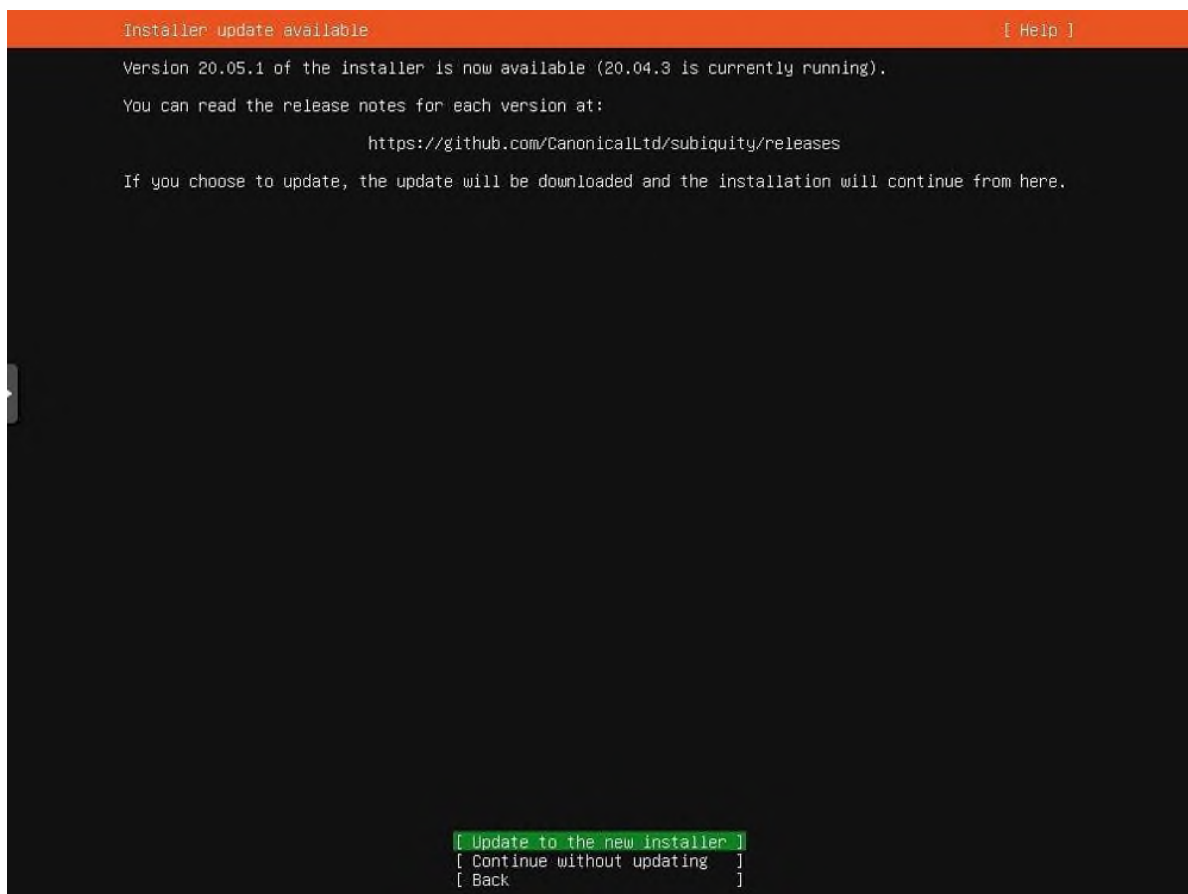
Цель: ознакомиться установкой, запуском Ubuntu 20.04 Server.

#### *Установка Ubuntu 20.04 Server*

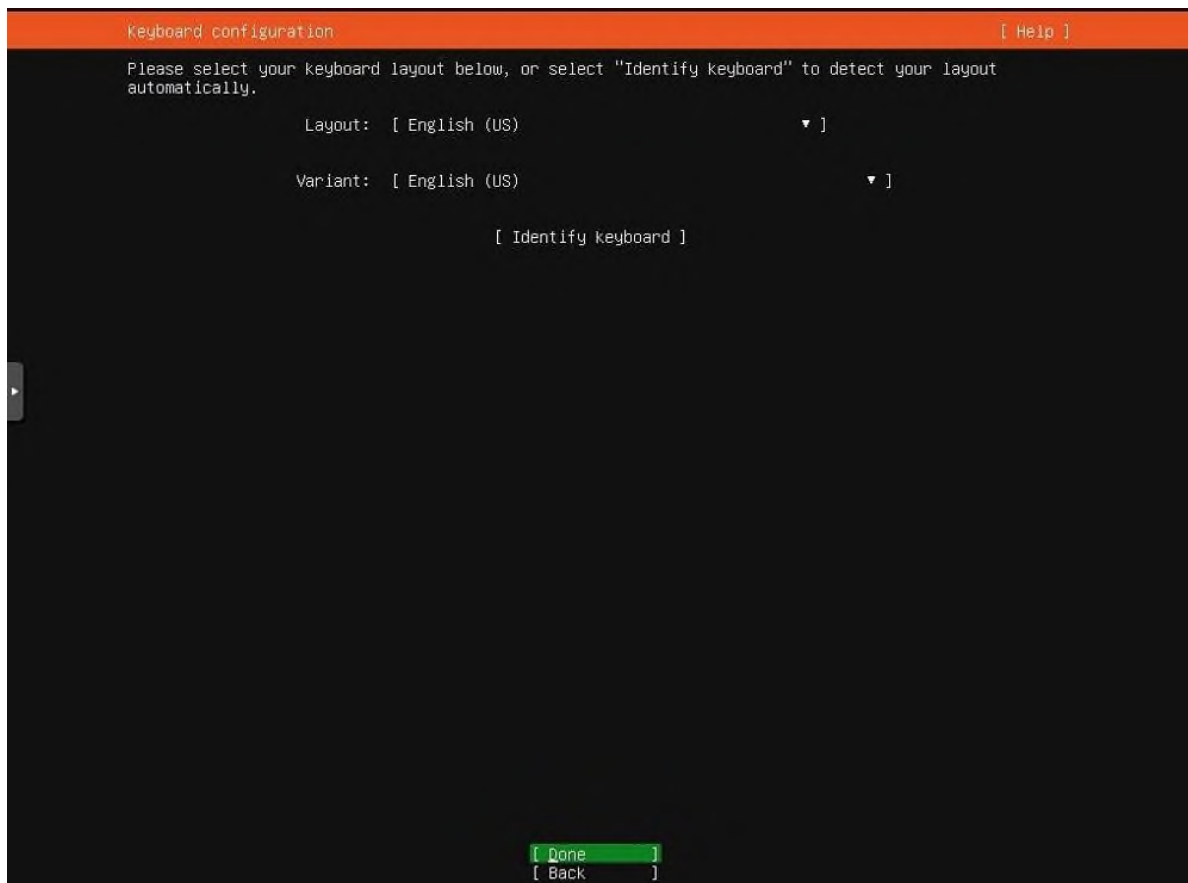
Создать загрузочный флеш накопитель приступить к установке. Выберите язык:



На следующем этапе будет предложено использовать действительный инсталлятор, или обновить его. Обновление происходит в фоновом режиме и занимает не более 10 секунд, обновите его:



Выберите раскладку клавиатуры. Указывайте везде English:



Настройте сеть. По умолчанию, получение IP адреса настроено по DHCP. В нашем примере мы будем использовать статический IP адрес. Используя интуитивно понятную навигацию, заполните необходимые поля

Network connections [ Help ]

Configure at least one interface this server can use to talk to other machines, and which preferably provides sufficient access for updates.

NAME	TYPE	NOTES
[ ens18 ]	eth	-

DHCPv4: 192.168.111.120/24  
2a:fa:e2:d8:fc:7d / Red Hat, Inc. / Virtio network device

[ Create bond ▶ ]

Edit ens18 IPv4 configuration

IPv4 Method: [ Manual ▼ ]

Subnet: 178.20.152.0/25

Address: 178.20.152.79

Gateway: 178.20.152.125

Name servers: 194.0.200.115, 8.8.4.4  
IP addresses, comma separated

Search domains:   
Domains, comma separated

[ Save ]  
[ Cancel ]

[ Done ]  
[ Back ]

Если доступ в интернет у вас осуществляется через прокси-сервер, укажите его:

Configure proxy [ Help ]

If this system requires a proxy to connect to the internet, enter its details here.

Proxy address:

If you need to use a HTTP proxy to access the outside world, enter the proxy information here. Otherwise, leave this blank.

The proxy information should be given in the standard form of "http://[[user][:pass]@]host[:port]/".

[ Save ]  
[ Cancel ]

Далее, установщик предложит вам ближайшее зеркало (Mirror), исходя из вашего регионального расположения. Оставьте предложенное по умолчанию, или укажите свой:

Configure Ubuntu archive mirror [ Help ]

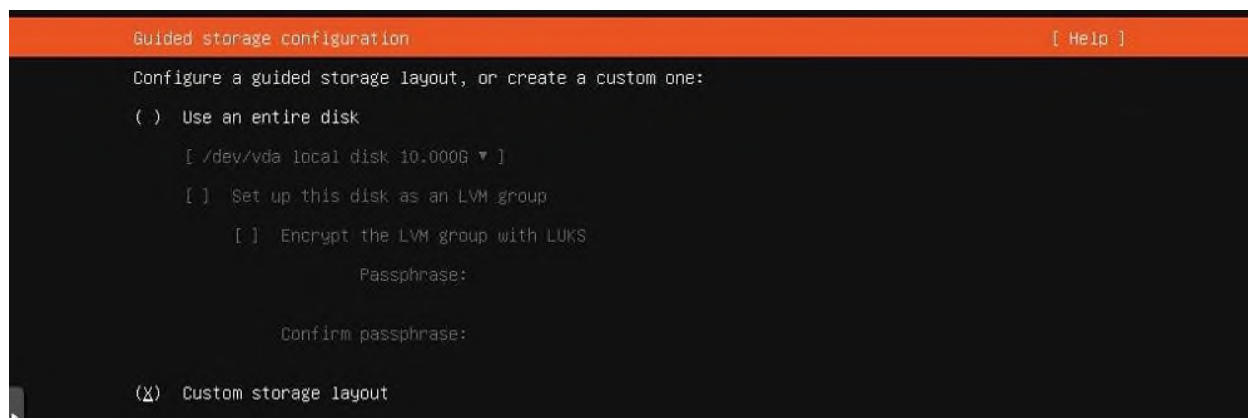
If you use an alternative mirror for Ubuntu, enter its details here.

Mirror address: http://ua.archive.ubuntu.com/ubuntu

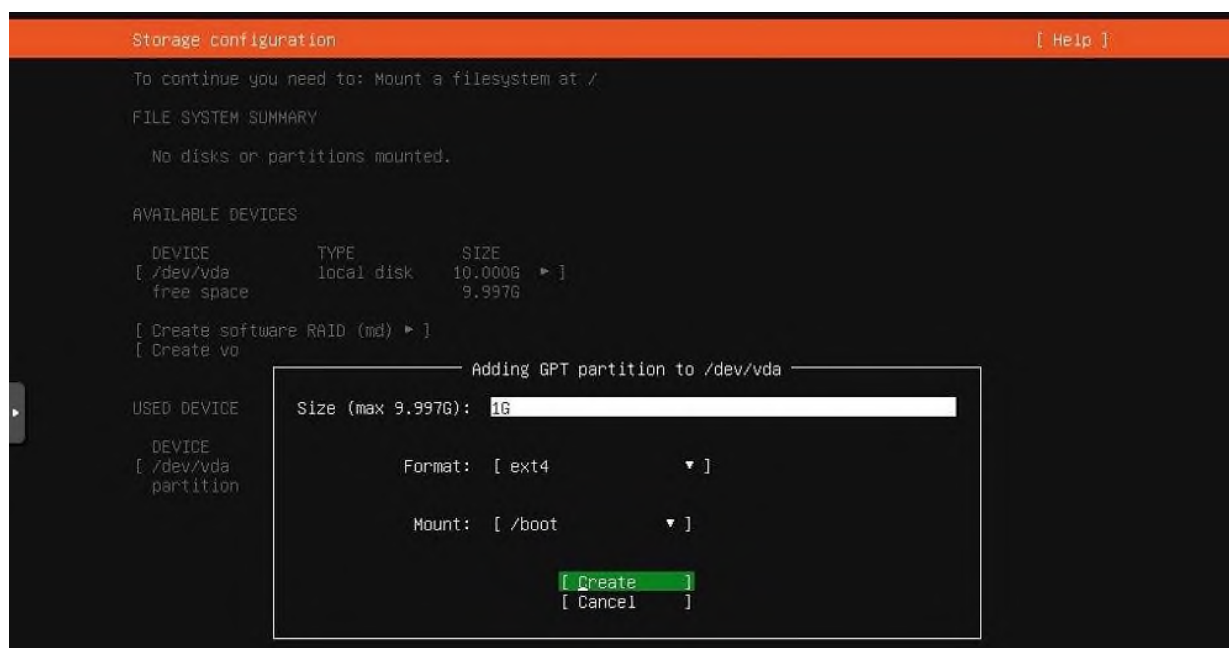
You may provide an archive mirror that will be used instead of the default.

[ Save ]  
[ Cancel ]

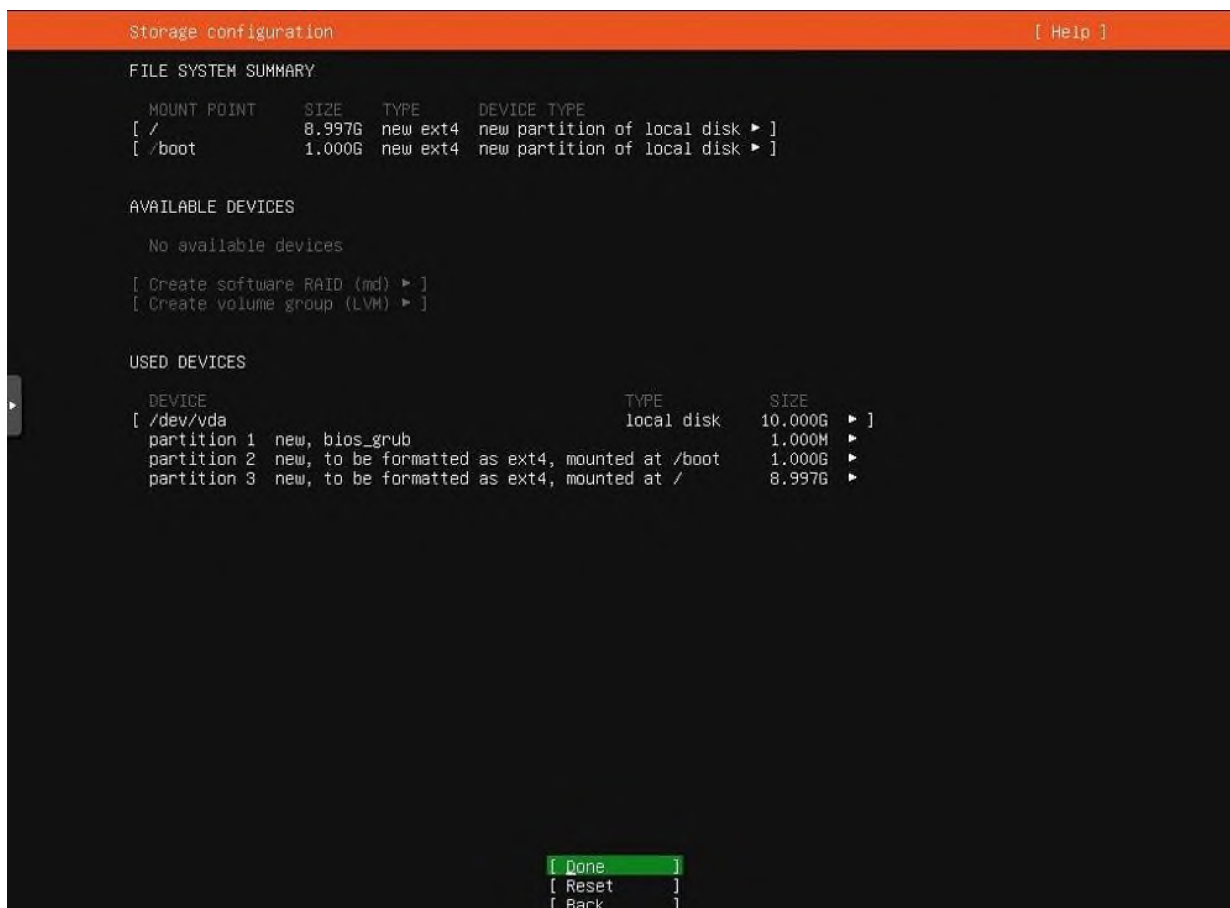
На этом этапе будет предложено разметить дисковое пространство. Выбрав «Use an entire disk» установщик сам разметит диски в автоматическом режиме. В зависимости от задач, вы можете выполнить разбивку разделов на собственное усмотрение, выбрав «Custom storage layout»:



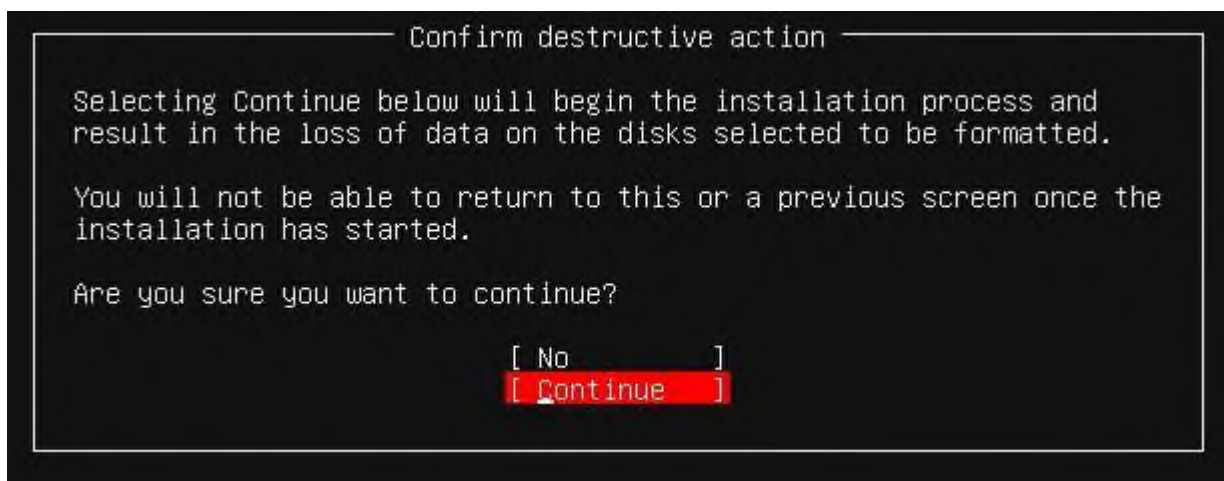
Следующий пример для наглядности, выносить раздел boot отдельно нет необходимости, вы можете все отдать под корень «/», что будет равнозначно «Use an entire disk». Вы можете сделать отдельным разделом boot или swap, выбрать формат файловой системы. Так же есть возможность создания LVM разделов. Меню интуитивно понятное:



Проверьте внимательно, все ли настроено как нужно и нажмите Done



Выберите «Continue» для подтверждения настроек:

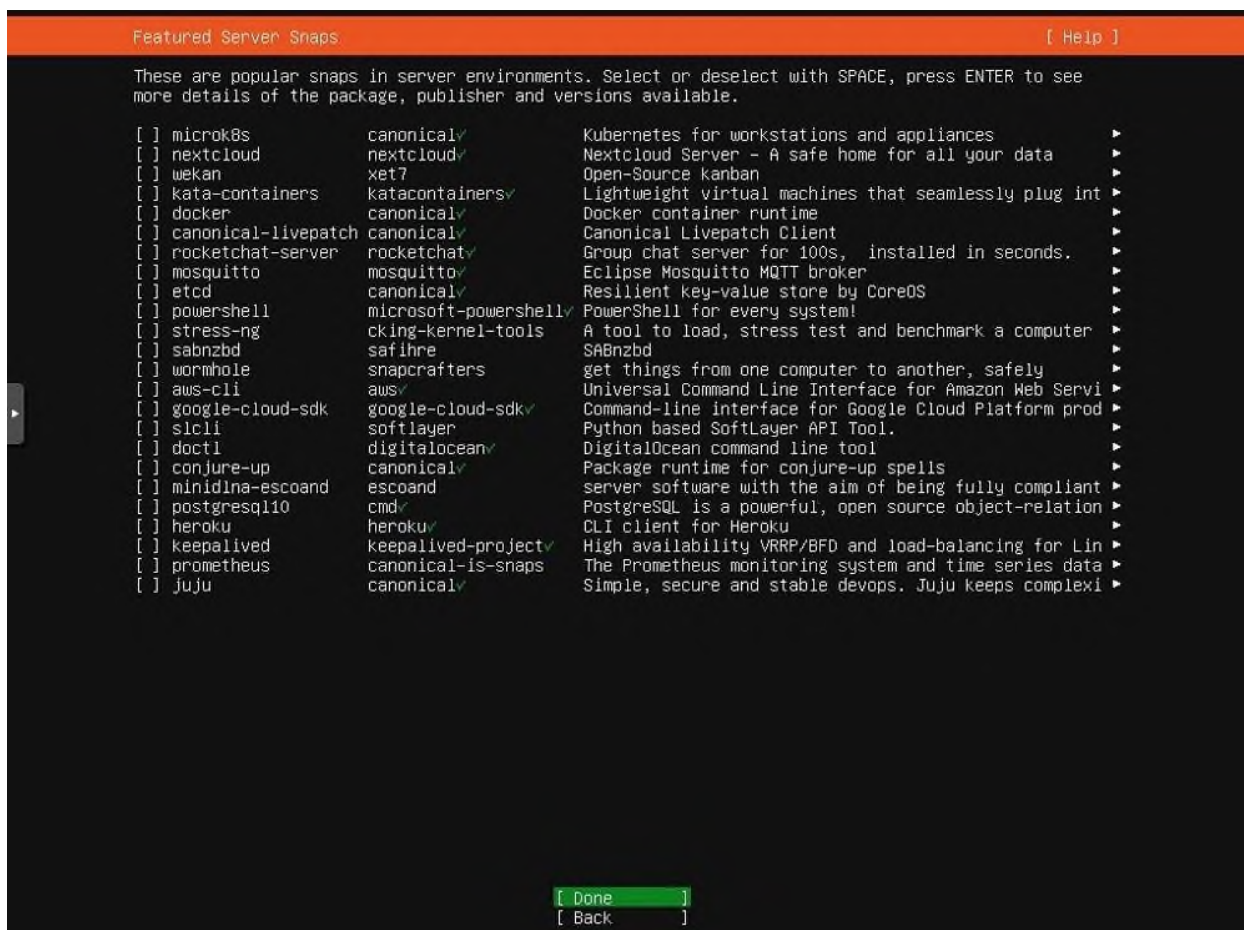


Укажите имя сервера и пользовательские данные для доступа к серверу. Вы можете выбрать любого пользователя кроме «root» и «admin», которые зарезервированы системой. Так же придумайте сложный пароль, он должен быть не менее 10 символов, включать в себя строчные и заглавные символы латинского алфавита, а так же цифры и спецсимволы:

Отметьте установку OpenSSH Server, для возможности удаленного подключения к нему, и нажмите «Done»:

Перед вами появится длинный список того что можно установить «из коробки». Очень интересный этап, здесь доступна и интеграция с Kubernetes 1.18, etcd, интеграция с Google Cloud, Postgresql10, Prometheus и много чего другого. Выбирать компоненты для установки нужно осознанно, при условии, что вам это действительно необходимо для задач.





Подтвердите «Done» и перед вами появится консоль установки в интерактивном режиме:

```
Install complete! [ Help ]

removing previous storage devices
configuring disk: disk-vda
configuring partition: partition-1
configuring partition: partition-3
configuring format: format-1
configuring partition: partition-4
configuring format: format-2
configuring mount: mount-2
configuring mount: mount-1
writing install sources to disk
running 'curtin extract'
curtin command extract
acquiring and extracting image from cp:///media/filesystem
configuring installed system
running '/snap/bin/subiquity.subiquity-configure-run'
running '/snap/bin/subiquity.subiquity-configure-apt /snap/subiquity/1866/usr/bin/python3
true'
curtin command apt-config
curtin command in-target
running 'curtin curthooks'
curtin command curthooks
configuring apt configuring apt
installing missing packages
configuring iscsi service
configuring raid (mdadm) service
installing kernel
setting up swap
apply networking config
writing etc/fstab
configuring multipath
updating packages on target system
configuring pollinate user-agent on target
updating initramfs configuration
finalizing installation
running 'curtin hook'
curtin command hook
executing late commands
final system configuration
configuring cloud-init
installing openssh-server -

[ View full log ]
```

По окончании установки, внизу интерактивной консоли появится «Reboot». Извлеките носитель с установщиком и выполните перезагрузку сервера.

```
[ View full log ]
[ Reboot ]
```

Первоначальная перезагрузка займет более продолжительное время, чем это будет в дальнейшем. Будут инициализированы службы `snapper`, выполнится инициализация устройств `cloud-init` и прочее. Все зависит от того, каким образом устанавливается Ubuntu, на гипервизоре, или `baremetall`. По окончании вы увидите окно аутентификации в систему. Так же, сервер будет доступен по `ssh`.

```
Ubuntu 20.04 LTS dedicated tty1
dedicated login:
```



## *Настройка сервера*

При подключении, установите минимальный набор утилит, которые понадобятся для работы:

```
root@dedicated:~# apt-get update
root@dedicated:~# apt-get install atop wget ntp vim net-tools -y
```

### **Подключение к серверу по ssh**

Первое, на что стоит обратить внимание, это то, что в Ubuntu пользователь root отключен от аутентификации. Это выполнено с точки зрения безопасности. Подключение к серверу происходит по имени пользователя, который указан при установке:

```
admt@freehost:~$ ssh user@178.20.152.78
user@178.20.152.78's password:
```

Чтобы попасть в режим суперпользователя, введите следующую команду, которая запросит пароль пользователя user и переключит в режим суперпользователя:

```
user@dedicated:~# sudo su
[sudo] password for user: *****
root@dedicated:/home/user#
```

Если по каким-то причинам вы хотите разрешить доступ пользователя root при подключении по ssh, то смените для него пароль:

```
root@dedicated:~# passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@dedicated:~#
```

Затем в конфигурационном файле /etc/ssh/sshd\_config раскомментируйте строку PermitRootLogin указав значение yes:

```
PermitRootLogin yes
```

Однако, в целях безопасности сервера, доступ по логину root лучше не предоставлять.

## Настройка сетевых интерфейсов

Начиная с Ubuntu 18.04 настройка сетевых интерфейсов происходит не в привычном каталоге `/etc/network/interfaces` как это реализовано в Debian-подобных дистрибутивах, а в каталоге `/etc/netplan/`. После установки ОС, в нем будет сгенерирован конфигурационный файл `00-installer-config.yaml`

Если конфигурационный файл по какой-то причине пуст, сгенерируйте его:

```
root@dedicated:~# netplan generate
```

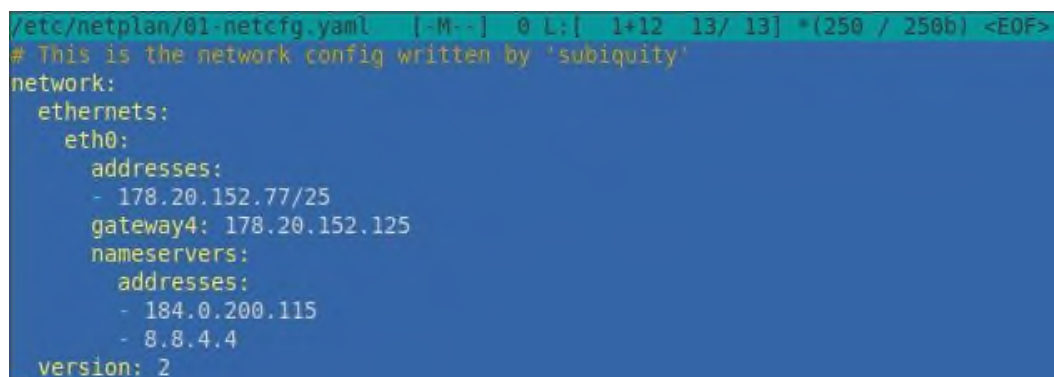
Для примера, выполним с ним некоторые операции. Переименуем его в более удобный вид:

```
root@dedicated:~# mv /etc/netplan/00-installer-config.yaml /etc/netplan/01-netcfg.yaml
```

Применить настройки:

```
root@dedicated:~# netplan apply
```

Конфигурационный файл имеет следующий вид:



```
/etc/netplan/01-netcfg.yaml [-M-] 0 L:[ 1+12 13/ 13] *(250 / 250b) <EOF>
# This is the network config written by 'subiquity'
network:
  ethernets:
    eth0:
      addresses:
      - 178.20.152.77/25
      gateway4: 178.20.152.125
      nameservers:
        addresses:
        - 184.0.200.115
        - 8.8.4.4
  version: 2
```

Он имеет формат `yaml` и чувствителен регистру. При конфигурации файлов данного формата табуляция недопустима. Только четная последовательность пробелов и структура наследования директив, иначе, вы получите синтаксическую ошибку.

В нашем примере, сетевой интерфейс имеет название `eth0`, но при инсталляции Ubuntu он имел название `enp0s8`. Переименовать в конфиге сетевой интерфейс недостаточно. Так же следует править загрузчик `grub`:

```
root@dedicated:~# mcedit /etc/default/grub
```

найдите и отредактируйте значение:

```
GRUB_CMDLINE_LINUX="net.ifnames=0 biosdevname=0"
```

```
root@dedicated:~# grub-mkconfig -o /boot/grub/grub.cfg
```

```
root@dedicated:~# reboot
```

## Удаление ненужных компонентов

В зависимости от того на чем вы устанавливали сервер, Ubuntu может без вашего ведома инициализировать некоторые компоненты, которые с одной стороны полезны, с другой стороны нет, если использование их не планируется. В нашем примере мы устанавливали виртуальный сервер на гипервизор Proxmox, Ubuntu без нашего ведома инициализировала службу cloud-init. Cloud-init представляет собой сценарий активации и инициализации виртуальных машин, который широко применяется для OpenStack и других облачных технологий с помощью скриптов, и манифестов. Можно провести параллели с Terraform. Это гибкая и полезная вещь. Однако, если специфика вашей работы не предусматривает автоматизированный деплой виртуальных машин в гипервизор или облако, удалить его можно следующим образом.

```
root@dedicated:~# echo 'datasource_list: [ None ]' | sudo -s tee /etc/cloud/cloud.cfg.d/90_dpkg.cfg
```

```
root@dedicated:~# apt-get purge cloud-init -y
```

```
root@dedicated:~# rm -rf /etc/cloud/
```

```
root@dedicated:~# rm -rf /var/lib/cloud/
```

## Пакетный менеджер snap

Давайте так же, командой lsblk отобразим наши дисковые устройства:

```
root@dedicated:~# lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
loop0  7:0    0  55M  1 loop /snap/core18/1705
loop1  7:1    0  69M  1 loop /snap/lxd/14804
loop2  7:2    0 27.1M  1 loop /snap/snapd/7264
sr0    11:0    1 1024M  0 rom
vda    252:0    0  10G  0 disk
├─vda1 252:1    0   1M  0 part
├─vda2 252:2    0   1G  0 part /boot
└─vda3 252:3    0   9G  0 part /
```

Начиная с версии 16.04, Ubuntu все настойчивей пытается продвинуть менеджер управления пакетами snap. Приведем краткое объяснение с официального сайта разработчиков:

«snap пакеты Ubuntu содержат саму программу, а также все ее зависимости. Это имеет несколько преимуществ перед обычными deb или rpm пакетами, обрабатывающими зависимости. А главное, из них - разработчик может быть уверен что нет никаких регрессий из-за изменений версий библиотек в системе»

Snap это удобная вещь, и мы рекомендуем с ним ознакомиться, документация Ubuntu довольно информативна как на русскоязычных, так и на англоязычных ресурсах.

Отобразим список установленных пакетов, которые установились при инсталляции:

```
root@dedicated:~# snap list
```

Если нам что-то не нужно, удалим:

```
root@dedicated:~# snap remove lxd
```

```
root@dedicated:~# snap remove core18
```

Добавить пакет можно следующим образом:

```
root@dedicated:~# snap install docker
```

Обновить все пакеты:

```
root@dedicated:~# snap refresh
```

Обновить определенный пакет:

```
root@dedicated:~# snap refresh docker
```

Посмотреть список доступных пакетов:

```
root@dedicated:~# snap changes
```

Однако, стоит заметить, сейчас не так много snap пакетов. С полным перечнем вы можете ознакомиться на сайте разработчиков <https://snapcraft.io/> Вместо snap можете использовать пакетный менеджер apt. Если у вас snapd не установлен, установить его можно следующим образом:

```
root@dedicated:~# apt install snapd
```

## Подключение swap раздела и настройка кеширования

Для начала убедитесь, что он не подключен:

```
root@dedicated:~# swapon --show
```

Если список пуст, создайте файл подкачки, например 2Gb. Измените права, и активируйте его. Порядок команд следующий:

```
root@dedicated:~# fallocate -l 2G /swapfile
root@dedicated:~# sudo chmod 600 /swapfile
root@dedicated:~# mkswap /swapfile
root@dedicated:~# swapon /swapfile
```

Проверка результата:

```
root@dedicated:~# swapon --show
root@dedicated:~# free -h
```

Что бы раздел swap после автозагрузки примонтировался автоматически, добавьте о нем информацию в /etc/fstab

```
root@dedicated:~# echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

Примечание: перед правкой fstab рекомендуется сделать его резервную копию. Если вы допустите какую либо ошибку, сервер может «уйти в себя» при перезагрузке.

Настроим swappiness. Параметр vm.swappiness по умолчанию имеет значение 60, и контролирует процент свободной памяти. Он контролирует значение, после которого начнется активный сброс данных в swap. Значение «60» означает следующее:  $100-60=40\%$ . Другими словами, при загрузке ОЗУ более чем 40%, данные начнут кешироваться в swap. Не существует оптимального значения, он обусловлен только конфигурацией вашего сервера. Если ОЗУ достаточно, это значение можно уменьшить следующими командами:

```
root@dedicated:~# sysctl vm.swappiness=10
```

В конце файла /etc/sysctl.conf добавьте следующую строку:

```
vm.swappiness=10
```

Настройте кеширование. Измените параметр `vfs_cache_pressure`, отвечающий за скорость удаления индексов из кеша. По умолчанию это значение равно 100. Так как обращение к индексам довольно частое, это значение вы можете уменьшить:

```
root@dedicated:~# sysctl vm.vfs_cache_pressure=50
```

В конце файла `/etc/sysctl.conf` добавьте следующую строку:

```
vm.vfs_cache_pressure=50
```

Опционально, не будет лишним удалить поддержку IPv6. Даже если вы этот протокол не используете, некоторое программное обеспечение все равно его прослушивает и иногда случаются сбои в его работе:

```
root@dedicated:~# sh -c 'echo 1 > /proc/sys/net/ipv6/conf/all/disable_ipv6'
```

В конце файла `/etc/sysctl.conf` добавьте следующую строку, в зависимости от интерфейсов:

```
# Отключение на всех интерфейсах
```

```
net.ipv6.conf.all.disable_ipv6 = 1
```

```
# Отключение на определенном интерфейсе
```

```
net.ipv6.conf.eth0.disable_ipv6 = 1
```

### Практическое занятие «Выполнение запросов к базе данных»

Цель работы: научиться создавать запросы простые и сложные к готовой базе данных.

Задание 1. Создание базы данных

1. Любым, изученным, способом создайте в новой базе данных 2 таблицы Студент и Работник и заполните их данными по образцу.

Студент

Код Студент	Фамилия	Имя	Отчество	Адрес	Номер телефона	Специализация
1	Иванов	Иван	Иванович	г. Тула	457896	администратор

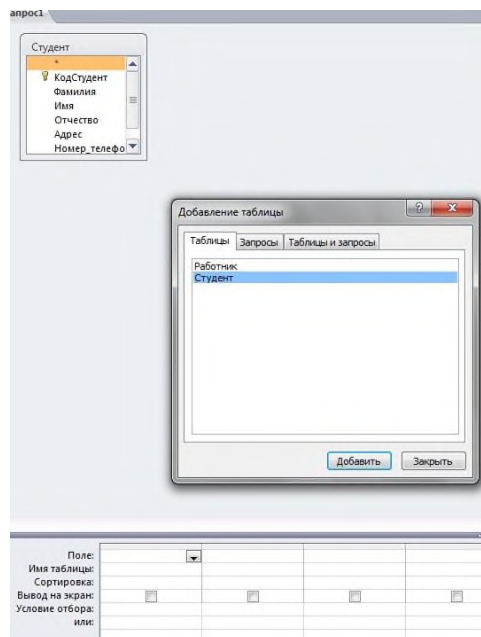
2	Петров	Сергей	Петрович	г. Москва	7458962	технолог
3	Голубева	Ольга	Ивановна	г. Белев	3698521	бухгалтер
4	Соколова	Инна	Олеговна	г. Тула	852967	бухгалтер
5	Мухина	Олеся	Петровна	г. Москва	8625471	технолог
6	Андреева	Анна	Романовна	г. Люберцы	748596	повар
7	Галкина	Дина	Евгеньевна	г. Люберцы	919597	технолог
8	Сорина	Ольга	Сергеевна	г. Москва	9191954	бухгалтер
9	Сидоров	Илья	Владимирович	г. Волгоград	126578	слесарь

#### Работник

Код работни ка	Фамилия	Имя	Отчество	Адрес	Организац ия	должность	телефон
1	Бубнов	Олег	Петрович	Г.Москва	РотФронт	директор	785456
2	Фролова	Ольга	Ивановна	Г. Люберцы	Гранд	бухгалтер	213568
3	Сорян	Виктория	Викторовна	Г.Тула	Рубин	бухгалтер	526598
4	Фельдман	Генрих	Вениаминов ич	Г.Москва	Растр	инженер	569726
5	Ильичева	Татьяна	Федоровна	Г. Воронеж	Симка	секретарь	569875
6	Тимофеева	Инна	Вячеславов на	Г.Елец	Магнит	продавец	687459

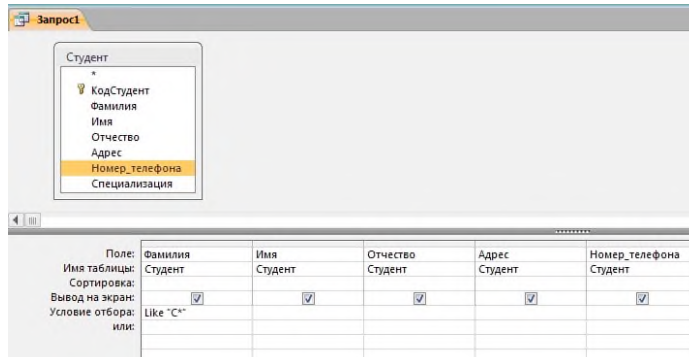
Задание 2. Создание запроса на выборку.

1. Выполните команду Создание – Конструктор запросов.
2. В появившемся диалоговом окне Добавление таблицы выберите из списка имя таблицы Студент, щелкните по кнопке Добавить



3. Закончите выбор, щелкнув по кнопке **Заккрыть**. Появится возможность выбора полей из таблицы “Студент”. Для этого достаточно дважды щелкнуть по именам полей или перетащить мышью названия полей в клетку запроса.

4. Создайте телефонную книгу для всех студентов, фамилии которых начинаются на букву С. Для этого в поле **Условие отбора** напишите условие Like “С\*”



5. Сохраните запрос, щелкнув по кнопке **Сохранить**. Введите имя запроса **Телефонная книга** и щелкните по кнопке **ОК**.

6. Щелкните по кнопке **Выполнить** для представления запроса. Закройте запрос.

7. Убедитесь в правильности полученного запроса, щелкнув по имени запроса **Телефонная книга** слева в окне **Все объекты Access**. Закройте таблицу.

8. Создайте запрос на выборку тех студентов, которые приехали из Москвы или Люберцы.

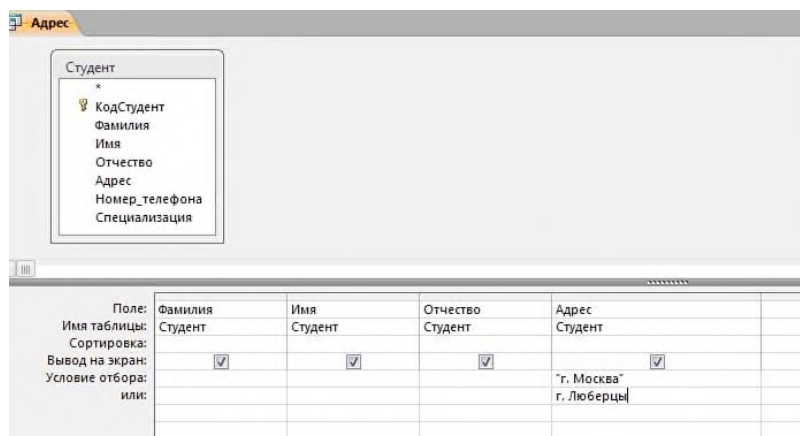
9. Для этого выполните команду **Создание – Конструктор запросов**.

10. В появившемся диалоговом окне **Добавление таблицы** выберите из списка имя таблицы **Студент**, щелкните по кнопке **Добавить**



11. Закончите выбор, щелкнув по кнопке Заккрыть. Появится возможность выбора полей из таблицы “Студент”. Для этого достаточно дважды щелкнуть по именам полей или перетащить мышью названия полей в клетку запроса.

12. В поле Условие отбора напишите условия для поля Адрес так, как показано на рисунке



13. Сохраните запрос, щелкнув по кнопке Сохранить. Введите имя запроса Адрес и щелкните по кнопке ОК.

14. Щелкните по кнопке Выполнить для представления запроса. Закройте запрос.

Самостоятельное задание.

1. Составьте запрос на выборку тех студенток, имя которых – Ольга.
2. Составьте запрос на выборку работников организаций, названия которых начинаются на букву Р, используя таблицу Работник.
3. Составьте запрос на выборку всех студентов, которые обучаются по специальности технолога.
4. Составьте запрос на выборку работников организаций, которые работают по должности инженер или бухгалтер.
5. Результаты предъявите учителю.

Задание 3. Завершение работы с программой Access.

1. Выполните команду Файл – Выход.
2. Если вы производили редактирование в базе данных, появится вопрос о сохранении изменений. Ответьте утвердительно.

**Практическое занятие «Выполнение изменений в базе данных, создание триггеров»**

Цель: изучение принципов применения, создания и отладки триггеров для баз данных в MS SQL Server

### Создание триггера

Для создания триггера необходимо быть владельцем таблицы, для которой триггер создается, либо входить в роль БД db\_owner или db\_ddladmin, либо же являться администратором SQL Server-a, т.е. входить в фиксированную роль сервера sysadmins. При добавлении триггера к таблице изменяется тип доступа, отношение к ней других объектов и т.д. Создание триггера похоже на объявление хранимой процедуры и имеет следующий синтаксис:

```
CREATE TRIGGER имя_триггера
ON таблица
[WITH ENCRYPTION]
{
  {FOR | AFTER | INSTEAD OF } {[DELETE] [,] [INSERT] [,] [UPDATE] }
  [WITH APPEND]
  [NOT FOR REPLICATION]
  AS
  { IF UPDATE (столбец_i)
    [ {AND | OR} UPDATE (столбец_j)]
    [ ... n]
    | IF (COLUMNS_UPDATED() {побитовый_оператор} битовая_маска )
    {оператор_сравнения} битовая_маска_столбца [ ... n ]
  }
  инструкции_SQL [ ... n ]
}
```

Имя\_триггера - должно соответствовать стандартным соглашениям об именах объектов SQL Server и быть уникальным в базе данных.

Таблица – название таблицы, для которой создается триггер.

WITH ENCRYPTION – эта опция дает разработчикам возможность запретить пользователям читать текст триггера после его загрузки на сервер. Опять же отметим, что для того, чтобы сделать текст триггера действительно невозстановимым, следует после шифрования удалить соответствующие ему строки из таблицы syscomments.

FOR INSERT, UPDATE и DELETE – это ключевые слова, определяющие операцию модификации таблицы, при выполнении которой будет активизирован триггер.

WITH APPEND – эта опция необходима, только если установленный уровень совместимости не

превышает 65 и используется для создания дополнительных триггеров.

NOT FOR REPLICATION – показывает, что триггер не активизируется при модификации таблицы в процессе репликации.

AS – ключевое слово, задающее начало определения триггера.

Инструкции\_SQL – в T-SQL триггер может содержать любое количество инструкций SQL, если они заключены в ключевые слова begin и end.

IF UPDATE (столбец) – для операций добавления и обновления данных можно определить дополнительные условия на конкретный столбец таблицы; при указании нескольких столбцов они разделяются логическими операторами.

IF (COLUMNS\_UPDATED()) – выше было показано, как можно с помощью конструкции If update (столбец) определять какие столбцы затрагиваются изменениями. Если необходимо проверять изменяется ли один конкретный столбец, то она очень удобно, однако, если надо построить сложное условие, включающее много столбцов, то хотя с ее помощью и можно достигнуть требуемого оператора, но полученная конструкция будет слишком громоздкой. Для таких случаев предназначена конструкция IF (COLUMNS\_UPDATED()...). Результатом функции COLUMNS\_UPDATED() является набор битов, каждый из которых отвечает за один столбец таблицы (младший бит соответствует первому столбцу; старший – последнему). Если в операции, вызвавшей срабатывание триггера была попытка изменить некоторый столбец, то соответствующий бит будет установлен в 1.

Побитовый\_оператор – побитовый оператор, определяющий операцию выделения нужных битов, полученных с помощью COLUMNS\_UPDATED(). Обычно используется оператор &.

Битовая маска – В сочетании с побитовым оператором, битовая маска позволяет выделить интересующие разработчика биты, т.е. определить изменялись ли в операции, вызвавшей срабатывание триггера интересующие его столбцы.

Оператор\_сравнения и битовая\_маска\_столбца – итак, функция COLUMNS\_UPDATED() дает набор битов, соответствующий изменяемым столбцам. С помощью битовой маски и побитового оператора над этим набором битов производится преобразование и получается некий промежуточный результат. С помощью оператора сравнения теперь этот промежуточный результат сравнивается с битовой маской столбца. Если результат сравнения – истина, то тело триггера, т.е. набор инструкций SQL будет выполнен; иначе нет.

Пример: пусть таблица имеет следующую структуру: create table mytable (a int, b int, c int, d int, e int). Пять столбцов соответствуют 5 битам, из которых младший соответствует столбцу a, старший – столбцу e. Пусть операция, приведшая к срабатыванию триггера изменяет столбцы a, b и e. Тогда функция Columns\_updated даст значение 10011. Пусть нас не интересует изменение столбцов b и d, но интересует изменение всех остальных столбцов (a, c и e, т.е. маска будет 10101) (напомним, что на момент написания триггера мы не знаем какие столбцы затронет на самом деле та или иная операция изменения или вставки, т.е. какой результат даст функция columns\_updated()). Задав побитовый оператор сравнения во время выполнения получим 10011 & 10101, что даст в результате 10001, что в десятичном представлении составляет 17. Сравнив это значение с помощью оператора сравнения и битовой маски столбца получим ответ – удовлетворяет ли операция изменения/вставки требуемым условиям. Так, например, если бизнес-

логика требует, чтобы триггер сработал при изменении все интересующих нас столбцов(а, с, е), то естественно битовая маска и битовая маска столбца должны иметь одинаковые значения, а оператор сравнения должен быть =. Т.е. в этом случае для нашего примера, вся конструкция будет иметь вид if (columns\_updated & 17)=17. Если же требуется, чтобы изменился хотя бы один из интересующих нас столбцов, то она, очевидно, будет иметь вид if (columns\_updated & 17)>0. С помощью битовых операций можно достичь большой гибкости при составлении таких конструкций.

Создавать триггеры можно и с помощью SQL Server Enterprise Manager. Для этого: Запустите SQL Server Enterprise Manager. Щелкните правой кнопкой мыши на таблице, для которой хотите создать триггер, и в контекстном меню выберите команду Task|Manage Triggers. В результате этих действий появится диалоговое окно, в котором можно ввести текст триггера и присвоить ему имя. После окончания ввода можно выполнить проверку синтаксиса и нажать кнопку ОК для сохранения триггера в базе данных.

Ограничения при создании триггеров:

Оператор create trigger может применяться только в одной таблице.

Триггер можно создавать только в текущей базе данных, но в нем можно ссылаться на внешние объекты.

В одном операторе создания триггера можно указывать несколько действий, на которые он будет реагировать.

В тексте триггера нельзя использовать следующие инструкции: ALTER DATABASE, ALTER PROCEDURE, ALTER TABLE, CREATE DEFAULT, CREATE PROCEDURE, ALTER TRIGGER, ALTER VIEW, CREATE DATABASE, CREATE RULE, CREATE SCHEMA, CREATE TRIGGER, CREATE VIEW, DISK INIT, DISK RESIZE, DROP DATABASE, DROP DEFAULT, DROP PROCEDURE, DROP RULE, DROP TRIGGER, DROP VIEW, RESOTRE DATABASE, RESTORE LOG, RECONFIGURE, UPDATE STATISTICS

Кроме того:

Любая правильная операция SET работает только в период существования триггера.

Нельзя выполнить триггер, анализируя в столбцах таблиц INSERTED и DELETED состояние большого двоичного объекта(BLOB), имеющего тип данных text или image, независимо от того, записывается эта процедура в журнал или нет.

Не следует применять инструкции SELECT, возвращающие результирующие наборы из триггера, для приложения-клиента, требующего специального управления результирующими наборами, независимо от того, делается это в хранимой процедуре или нет.

Нельзя создавать INSTEAD OF UPDATE и DELETE триггеры на таблицы, имеющие внешние ключи с установленными опциями каскадного изменения или удаления соответственно

Примеры использования триггеров

Пример1: Триггеры вставки и обновления

Эти триггеры удобны, поскольку они могут поддерживать условия ссылочной целостности и обеспечивать правильность данных перед вводом в таблицу. Обычно триггеры применяются для обновления столбцов отсчета времени или для проверки данных в определенных столбцах на соответствие требуемому критерию. Триггеры следует применять, когда критерий проверки более сложен, чем условие декларативной

целостности.

В приведенном ниже примере триггер выполняется всегда, когда в таблицу Sales вставляется строка или выполняется ее модификация. Если дата заказа не находится в пределах первых 15 дней месяца, строка в таблицу не вводится.

```
CREATE TRIGGER Tri_Ins_Sales
ON Sales
FOR INSERT, UPDATE
AS
/* Объявить необходимые локальные переменные */
DECLARE @nDayOfMonth TINYINT
/* Найти информацию о добавленной записи */
SELECT @nDayOfMonth = DatePart (day, i.ord_date)
FROM Sales s, Inserted i
WHERE s.stor_id = i.stor_id
AND s.ord_num = i.ord_num
AND s.title_id = i.title_id
/* Проверить критерий отказа и в случае необходимости
послать сообщение об ошибке */
IF @nDayOfMonth > 15
BEGIN
/* Примечание; всегда сначала производите откат. Вы можете не знать,
какого рода ошибка обработки произошла, что может вызвать
неоправданно продолжительное время блокировки */
ROLLBACK TRAN
RAISERROR('Выполняются только заказы, поданные в первые
15 дней месяца', 16, 10 )
END
```

Если теперь попытаться вставить или обновить запись в таблице, при несоблюдении заданного условия получим соответствующее сообщение об ошибке.

Обратите внимание, что фрагмент кода обращается к новой таблице, причем в списке таблиц базы данных эта таблица отсутствует. В данном случае таблица inserted содержит копию каждой строки, которая будет добавлена только в случае успешного завершения транзакции. Эта таблица и ее значения применяются при выполнении любой операции сравнения для проверки правильности транзакции.

Столбцы таблицы inserted в точности совпадают со столбцами рабочей таблицы. Сравнение можно выполнить по столбцам, как это сделано в данном примере, где для проверки правильности дат продажи сравниваются столбцы таблицы базы данных sales.

Можно также создать триггеры, выполняющие работу только в случае обновления конкретного столбца. Для принятия решения о продолжении обработки в триггере может быть применена инструкция IF

UPDATE:

```
IF UPDATE(au_lname)
AND (@@ROWCOUNT=1)
BEGIN
    ...
END
```

Код внутри блока выполняется только в том случае, если обновляется столбец au\_lname. Всегда помните, что обновляемый столбец изменяется не во всех случаях. Если возникает необходимость в каких-либо изменениях, многие приложения, включая большинство корпоративных систем, просто обновляют всю строку.

Операция UPDATE задействует обе системные таблицы, в таблице Inserted хранятся новые значения, а в таблице Deleted – старые. Поэтому Вы при анализе изменений можете использовать обе эти таблицы.

Часто бывает необходимо заменить некоторые значения на неопределенные, это делается элементарной операцией присваивания, например ¶ NUM\_READER=NULL

Пример2: Триггеры удаления.

Триггеры удаления (delete triggers) обычно применяются в двух случаях: предотвращение удаления строк, которое может вызвать проблемы с целостностью данных, например, строки, используемой в качестве внешнего ключа к другим таблицам и выполнение каскадных операций удаления дочерних (children) строк главной (master) строки. Такой триггер можно использовать для удаления всей информации о заказах из главной строки продаж

Триггеры учитывают общую сумму всех строк, на которые действует запрошенная операция. Таким образом, они должны иметь возможность работать с различными комбинациями информации в таблице и возвращать необходимые данные. Например, при выполнении инструкции DELETE FROM Authors, триггер должен учесть, что инструкция уничтожит все строки из таблицы.

В следующем примере использование переменной @@ROWCOUNT позволяет предотвратить удаление более одной строки. Этот триггер выполняется всегда, когда пользователь пытается удалить строку из таблицы stores. Если информация касается продаж, то триггер препятствует выполнению этого запроса.

```
CREATE TRIGGER Tri_Del_Stores
ON Stores
FOR DELETE
AS
/* Проверка количества модифицируемых строк и запрещение удаления более одной строки за один
раз */
IF @@ROWCOUNT > 1
BEGIN
    ROLLBACK TRAN
    RAISERROR('За один раз можно удалить только одну строку.', 16, 10 )
END
```

```

/* Объявление временной переменной для сохранения уничтожаемой информации*/
DECLARE @StorID char(4)
/* Получение значения удаляемой строки */
SELECT @StorID = d.stor_id
FROM Stores s, Deleted d
WHERE s.stor_id *= d.stor_id
IF EXISTS (SELECT *
           FROM Sales
           WHERE stor_id = @storID )
BEGIN
ROLLBACK TRAN
RAISERROR ( 'Эта информация не может быть удалена, поскольку имеется соответствующая запись
в таблице Sales.', 16, 10)
END

```

Примечание: Применение `raiserror` — это самый простой способ посылки вызывающему процессу или пользователю подробной и конкретной информации об ошибке. `Raiserror` дает возможность указать текст сообщения, уровень опасности, состояние информации и скомбинировать все это для пользователя в описательное сообщение. Эта инструкция также облегчает написание общих блоков обработки ошибок в приложениях-клиентах.

В этом примере используются также несколько инструкций управления транзакциями, которые позволяют остановить выполнение операций. Обратите внимание, что фрагмент кода обращается к новой таблице. В списке таблиц базы данных эта таблица отсутствует. В данном случае таблица `Deleted` содержит копию каждой строки, которая будет добавлена только в случае успешного завершения транзакции. Эта таблица и ее значения применяются при выполнении любого сравнения для проверки правильности транзакции.

Столбцы в таблице `Deleted` в точности совпадают со столбцами рабочей таблицы. Сравнение можно выполнить по столбцам, как это показано в примере, где столбцы таблицы `Deleted` сравниваются со столбцами базы данных `Sales`. При этом осуществляется проверка того, что предназначенная для удаления информация не включает данных о продажах.

### Пример3: INSTEAD OF триггеры

`INSTEAD OF` триггеры отличаются от обычных (`AFTER`) триггеров тем, что выполняются не после выполнения операции, приведшей к его срабатыванию, а вместо нее со всеми вытекающими последствиями (такими например, как возможность их использования совместно с ограничениями целостности). Системные таблицы `Inserted` и `Deleted` используются в них также, как и в `AFTER` триггерах. Тело триггера может дублировать операцию, которая вызвала его срабатывание, а может и нет. Другими словами, если мы описываем `INSTEAD OF DELETE` триггер, то ничто не мешает нам в нем выполнить операцию `DELETE`, удаляющую все строки, которые и должны были быть удалены в соответствии с вызвавшей триггер операцией, хотя можно этого и не делать. Приведем пример использования `INSTEAD OF` триггера.

Таблица jobs связана отношением 1:M с таблицей employees. Поэтому невозможно удалить работу, если на нее уже назначены сотрудники. Создадим триггер, который при удалении работы будет проверять назначены ли на нее сотрудники или нет. Если назначены, то работа не будет удаляться. В связи с тем, что имеется ограничение целостности (DRI), то работа AFTER триггера совместно с ним невозможна. Т.е. можно создать такой триггер:

```
CREATE TRIGGER Check_Job ON JOBS
FOR DELETE
AS
IF EXISTS (SELECT * FROM Employee e JOIN Deleted d ON e.job_id=d.job_id) BEGIN
ROLLBACK TRAN
END
```

Кстати, отметим, что в отличие от примера2 этот триггер позволяет удалять сразу несколько строк. Однако, такой триггер сможет работать корректно, только если разорвать связь между таблицами Employees и Jobs, чтобы перед выполнением триггера не обрабатывались DRI.

Но можно создать INSTEAD OF триггер:

```
CREATE TRIGGER Check_Job ON JOBS
INSTEAD OF DELETE
AS
DELETE FROM Jobs FROM Jobs j JOIN deleted d on d.job_id=j.job_id
WHERE j.job_id NOT IN (SELECT DISTINCT Job_id FROM Employee)
```

Такой триггер не будет иметь конфликтов с DRI и будет выполняться.

Проверка DRI выполняется сразу при выполнении операции, т.е. раньше чем выполнение AFTER-триггера; при использовании INSTEAD OF триггера операция по сути не выполняется, а управление передается триггеру, поэтому DRI не будет выполняться.

Как уже было сказано, таблица inserted содержит добавленные строки, а таблица deleted – удаленные. Нетрудно догадаться, что при выполнении операции изменения, будет использована и таблица inserted и таблица deleted. В этом случае старые значения окажутся в таблице Deleted, а новые – в таблице Inserted. Объединяя их по ключевому столбцу (столбцам), нетрудно определить какие значения на какие изменены.

Использование вложенных триггеров

Триггеры можно встраивать друг в друга, допускается 32 уровней вложенности. Если операции вложенного триггера нежелательны, SQL Server можно сконфигурировать так, чтобы отключить их.

Примечание: уровень вложенности триггера можно проверить в любое время, опросив значение, установленное в переменной @@NESTLEVEL. Оно должно находиться в пределах от 0 до 32.

Вложенные триггера могут привести к рекурсии. Рекурсия бывает двух видов – прямая и косвенная.



Прямая рекурсия получается в случае, если срабатывание триггера приводит к изменениям, которые вновь вызывают его же. Косвенная рекурсия получается когда срабатывание триггера приводит к изменениям, которые приводят к срабатыванию другого триггера, что в свою очередь приводит к изменениям, вызывающим срабатывание первого триггера. Конечно же цепочка может состоять не только из двух триггеров, но из большего числа. Прямую рекурсию можно отключить (и включить) с помощью опции БД `recursive triggers`. Отключить (и включить) косвенную рекурсию, равно как и вложенность триггеров вообще, можно с помощью серверной опции `nested_triggers`. Эта опция определяет возможность вложенности триггеров не для одной конкретной БД, а для всего сервера.

Следует отметить, что `INSTEAD OF` триггеры по своей природе не подвержены прямой рекурсии.

При создании триггера SQL Server не может самостоятельно распознать, что некоторая вложенная конструкция вызывает бесконечный цикл. Подобный факт может быть установлен только во время выполнения этого триггера. Предположим, что таблица `Table_A` включает триггер `trigger_A`, который выполняется, когда происходит обновление `Table_A`. При выполнении `trigger_a` вызывает обновление таблицы `table B`. Эта таблица включает в себя триггер `trigger_b`, который выполняется, когда обновляется `table_v`, и вызывает обновление таблицы `table_A`. Таким образом, если пользователь обновляет любую из этих двух таблиц, два триггера продолжают бесконечно вызывать выполнение друг друга. При возникновении такой ситуации SQL Server закрывает или отменяет выполнение триггера.

Пример1: Представим, что таблица `Sales` включает один триггер, а таблица `stores` — другой. Ниже показано определение двух вложенных триггеров, которые выполняются, если в таблице `sales` производится операция удаления:

```
/* Первый триггер уничтожает строки в таблице Stores,
если уничтожаются строки таблицы Sales */
CREATE TRIGGER Tri_Del_Sales
ON Sales
FOR DELETE
AS
/* Объявление выполняемого триггера */
PRINT 'Выполняется триггер удаления для таблицы Sales ...'
/* Объявление временной переменной для хранения удаляемой информации */
DECLARE @sStorID char(4),@sMsg varchar(40)
/* Получение значения ID удаляемой строки */
SELECT TOP 1 @sStorID = stor_id
FROM Deleted
/* Deleted — это вспомогательная таблица, которую SQL Server
использует для хранения уничтоженных записей */
/* Удаление строки */
SELECT @sMsg = 'магазин ' + @sStorID + ' удален'
```

```

PRINT @sMsg
DELETE FROM Stores
WHERE stor_id = @sStorID
PRINT 'Конец выполнения триггера для таблицы Sales'
GO

```

/\* Второй триггер уничтожает строки одной таблицы,  
если уничтожаются строки другой \*/

```

CREATE TRIGGER Tri_Del_Stores
ON Stores
FOR DELETE
AS
/* Объявление выполняемого триггера */
PRINT 'Выполняется триггер удаления для таблицы Stores ...'
/* Объявление временной переменной для хранения информации,
уничтожаемой из таблицы */
DECLARE @sStorID char(4), @sMsg varchar (200)
/* Получение уничтожаемого значения */
SELECT TOP 1 @sStorID = stor_id
FROM Deleted
/* Deleted — это вспомогательная таблица, которую SQL Server
использует для хранения уничтоженных записей */
IF @@ROWCOUNT = 0
BEGIN
PRINT 'В таблице Stores нет соответствующих строк'
RETURN
END
/* Удаление записи */
SELECT @sMsg = 'Удаление скидок, относящихся к магазину ' + @sStorID
PRINT @sMsg
DELETE Discounts
WHERE Stor_id = @sStorID
PRINT 'Количество удаленных скидок: ' + CONVERT(VARCHAR(4), @@ROWCOUNT)
PRINT 'Конец выполнения триггера для таблицы Stores'

```

Если инструкция DELETE выполняется в таблице Sales, как показано в следующем примере, активизируется триггер, что в свою очередь вызывает выполнение триггера таблицы stores.

Выполним :

```
DELETE FROM Sales WHERE stor_id = '8042'
```

Результат:

Выполняется триггер удаления для таблицы Sales ...

магазин 8042 удален

Выполняется триггер удаления для таблицы Stores ...

Удаление скидок, относящихся к магазину 8042

(1 row(s) affected)

Количество удаленных скидок: 1

Конец выполнения триггера для таблицы Stores

(1 row(s) affected)

(4 row(s) affected)

Конец выполнения триггера для таблицы Sales

Обратите внимание на порядок выдаваемых сообщений. Сначала запускается триггер для таблицы Sales. Он удаляет строку из таблицы Stores, запуская таким образом для нее триггер. При этом фактически еще ни из таблицы Sales ни из таблицы Stores удаления не произошло (удаление в процессе) – об этом говорит то, что не выдано автоматическое сообщение сервера "(N row(s) affected)", которое выдается при удалении из любой строки и показывает сколько строк было удалено.

После запуска триггер на таблицу Stores удаляет связанные строки из таблицы скидок (Discounts), о чем выдается сообщение (1 row(s) affected). После чего он выдает соответствующие сообщения и заканчивает свою работу. Как только он закончил свою работу, собственно удаляется строка из таблицы Stores, удаление которой и вызвало его работу. Далее, поскольку эта строка удалена, происходит возврат к работе триггера на таблицу Stores. Этот триггер выдает свое последнее сообщение об окончании работы и завершается. Как только он завершился выдается сообщение (1 row(s) affected), свидетельствующее об удалении строки из таблицы Stores. И уже только после этого окончательно удаляются строки из таблицы Sales.

Примечание: Триггеры и механизм декларативной ссылочной целостности обычно не могут работать вместе. Например, в предыдущем примере показано, что перед выполнением инструкции DELETE необходимо сначала удалить условие на значение FOREIGN KEY в таблице Discounts. Везде, где это возможно, следует применять либо триггер, либо условие ссылочной целостности. Однако, как уже говорилось, в MS SQL SERVER 2000 появились INSTEAD OF триггеры. Их можно использовать совместно с механизмами декларативной целостности, нельзя только использовать при этом каскадные операции в связях на ту же операцию, на которую создан INSTEAD OF триггер. Например, если создан INSTEAD OF

DELETE триггер, то нельзя в связях, в которых эта таблица является подчиненной использовать конструкцию ON DELETE CASCADE.

Пример2: Теперь приведем пример прямой рекурсии в триггерах: создадим триггер, который при удалении сотрудника удалял бы также тех сотрудников, у которых такая же фамилия, как у удаляемого или такое же имя. Причем сотрудники удаляются триггером, то на это удаление опять же срабатывает этот же триггер и опять удаляет людей с такой же фамилией или именем.

```
CREATE TRIGGER Del_Empl_Tr ON Employee
FOR DELETE
AS
IF EXISTS (SELECT * FROM Employee e
JOIN Deleted d on e.lname=d.lname OR e.Fname=d.fname
)
DELETE FROM Employee
FROM Employee e JOIN Deleted d on e.lname=d.lname OR e.Fname=d.fname
```

В БД pubs нет сотрудников, имеющих одинаковые фамилии или имена, но Вы можете сами добавить таких сотрудников и проверить что будет, удалив одного из них. Пусть, например, были сотрудники

Fname	Lname
Петр	Васильев
Иван	Иванов
Михаил	Иванов
Иван	Сергеев
Петр	Сергеев

Если теперь выполнить инструкцию

```
DELETE FROM Employee WHERE Fname= 'Иван' AND Lname= 'Иванов'
```

То кроме триггера, который запустится при удалении, удалит также Ивана Сергеева и Михаила Иванова. После чего на это удаление будет опять же запущен триггер, который будет искать всех Иванов и Михайлов, а также Ивановых и Сергеевых. В результате его работы будут удален Петр Сергеев. Опять запустится этот же триггер и удалит Петра Васильева. После этого опять запустится этот триггер и будет искать Петров и Васильевых, но поскольку их больше нет в таблице, то на этом выполнение закончится.

Заметьте, что тут обязательно надо делать проверку IF EXISTS. Если ее не сделать, то, когда дело дойдет до удаления Петра Васильева, то будет выполнена инструкция DELETE и, хотя она фактически никого не удалит, но вновь вызванный триггер опять вызовет самого себя (опять никого фактически не удаляя) и т.д., до превышения максимального уровня вложенности – 32. После достижения уровня вложенности 32 произойдет ошибка и все действия будут отменены.

Пример3: Косвенная рекурсия. Изменим пример1 таким образом, чтобы если удаляется строка из

таблицы Sales, то удалялся бы и магазин, в котором была сделана эта продажа. А поскольку отношение между ними 1:M, то в удаляемом магазине может быть множество продаж, а не только та, которую мы пытаемся удалять. Поэтому цепочка должна быть следующая: Удаляем продажу → удаляется магазин в котором она была сделана → удаляются все остальные продажи, сделанные в этом магазине и удаляются все скидки, привязанные к этому магазину. Кроме того, реализуем эти триггера в виде INSTEAD OF – триггеров, чтобы не было необходимости разрывать связи между таблицами.

```
CREATE TRIGGER Tri_Del_Sales
ON Sales
instead of DELETE
AS
DELETE FROM Sales FROM Sales s JOIN Deleted d on d.ord_num=s.ord_num
IF EXISTS (SELECT * FROM Stores s JOIN Deleted d ON d.stor_id = s.stor_id )
DELETE FROM Stores FROM Stores s JOIN Deleted d ON d.stor_id = s.stor_id
GO
```

```
CREATE TRIGGER Tri_Del_Stores
ON Stores
INSTEAD OF DELETE
AS
DELETE FROM Discounts FROM Discounts di JOIN Deleted de on di.stor_id=de.stor_id
IF EXISTS(SELECT * FROM Sales s JOIN Deleted d on d.stor_id=s.stor_id)
DELETE FROM Sales FROM Sales s JOIN Deleted d on d.stor_id=s.stor_id
DELETE FROM Stores FROM Stores s JOIN Deleted d on d.stor_id=s.stor_id
```

Для проверки можно выполнить команду

```
DELETE FROM Sales WHERE ord_num='P723'
```

В результате из таблицы Sales будет удалена не только строка с кодом заказа 'P723', но и три другие строки, относящиеся к тому же магазину (код 8042). Также будет удален сам магазин 8042 и относящаяся к нему скидка.

В приведенном примере, кроме всего прочего, удалены все выводы сообщений и изменены вызовы операторов DELETE – поскольку выводов сообщений нет, то нет и необходимости формировать значение локальной переменной @sStroId. Использование этой переменной в операторе DELETE несколько ограничивало применимость триггеров – так триггеры в примере2 были рассчитаны на то, что будут удаляться записи только для одного магазина, и при удалении записей, относящихся сразу к нескольким магазинам работали некорректно. Теперь же такого ограничения нет, поскольку удаляются все записи, связанные с записями в таблице Deleted (т.е. со всеми фактически удаляемыми строками).

Можно задать вопрос – зачем использовать рекурсию и не проще ли было бы при удалении из таблицы Sales удалять в триггере на нее все записи из самой себя, относящиеся к тому же магазину, как и удаляемая строка продажи. После этого удалять строку из таблицы Stores. А в триггере на таблицу Stores удалять связанные записи только из таблицы Discounts. Да, так можно сделать, но только в том случае, если мы всегда будем давать команду удаления именно из таблицы Sales (как например это сделано выше при проведении проверки). Однако, мы можем дать команду удаления и из таблицы Stores, например так:

```
DELETE FROM stores WHERE stor_id=8042
```

И в этом случае мы также хотим, чтобы команда отработала корректно. Если же в триггере на таблицу Stores, как было предложено в вопросе, не будет включено удаление из Sales, то если для удаляемого магазина есть продажи, то такое удаление приведет к ошибке. Наш же пример позволяет решить эту проблему. Ну а если уж в триггер на Stores включена команда удаления из Sales, то в триггере на Sales нет необходимости включать удаление продаж в том же магазине, что и удаляемый, поскольку это будет автоматически выполнено через рекурсию.

Замечание1: чтобы уже созданные в предыдущих примерах триггера не мешались – надо их удалить с помощью инструкции DROP TRIGGER имя\_триггера.

Замечание2: Еще раз обращаем Ваше внимание, что для того, чтобы рекурсия работала должны быть выставлены соответствующие опции базы данных и сервера.

Пример4: В последнем примере рассмотрим случай определения нескольких триггеров для одной операции модификации таблицы.

```
CREATE TRIGGER trig_del_1 ON Authors FOR DELETE AS
PRINT 'Триггер удаления №1 '
GO
CREATE TRIGGER trig_del_2 ON Authors FOR DELETE AS
PRINT 'Триггер удаления №2 '
GO
CREATE TRIGGER trig_upd_1 ON Authors FOR UPDATE AS
PRINT 'Триггер обновления №1'
GO
CREATE TRIGGER trig_upd_3 ON Authors FOR UPDATE AS
PRINT 'Триггер обновления №3' '
GO
CREATE TRIGGER trig_upd_2 ON Authors FOR UPDATE AS
PRINT 'Триггер обновления №2'
GO
```

А теперь попробуем изменить какую-либо запись в таблице:

```
UPDATE Authors  
SET au_fname = 'Юрий' WHERE au_lname = 'Тихомиров' ;
```

сработают все три триггера обновления:

Триггер обновления №1

Триггер обновления №3

Триггер обновления №2

Обратите внимание на последовательность выполнения триггеров — она определяется порядком их создания. Если теперь удалить триггер `trig_upd_3`, а потом создать его снова, то при обновлении таблицы получим следующий результат:

Триггер обновления N'1

Триггер обновления №2

Триггер обновления №3

Множественные триггеры достаточно активно используются при репликации.

Отображение информации о триггере и его изменения.

Для выяснения назначения триггера таблицы необходимо отобразить информацию, описывающую любой триггер, которым владеет таблица. Существует несколько путей получения информации о триггере конкретной таблицы. Одним из них является SQL Server Enterprise Manager, другим- системные процедуры `sp_help` и `sp_depends`. Для того, чтобы посмотреть текст триггера через Enterprise Manager выполните следующее:

1. В ЕМ выберите сервер и БД, с которой вы хотите работать.
2. Откройте таблицу в режиме проектирования — выполните команду `Design Table` и в ее окне нажмите кнопку `Triggers` на панели инструментов.
3. Появится диалоговое окно создания триггера, где можно посмотреть текст любого из установленных триггеров.

Системные хранимые процедуры `sp_help` и `sp_depends` были уже описаны в разделе хранимые процедуры.

Для того, чтобы изменить функциональность триггера, можно либо удалить его и создать новый с соответствующими изменениями, либо сразу изменить уже существующий. Для того, чтобы изменить существующий триггер в T-SQL существует соответствующая команда `ALTER TRIGGER`. Ее синтаксис идентичен синтаксису `CREATE TRIGGER` (конечно же за исключением ключевого слова `ALTER` вместо `CREATE`), поэтому мы не будем его здесь приводить.

Можно также изменить триггер с помощью ЕМ. Для этого после того как вошли в него (см. выше),

надо просто внести изменения и применить их.

#### Удаление триггеров

Иногда нужно удалить триггеры из таблицы (или таблиц). Например, при перемещении приложения в рабочую среду может потребоваться удаление триггеров, обеспечивавших высокое качество обработки, но сильно уменьшавших производительность. Можно просто удалить триггеры для замены их более новой версией. Для удаления триггера применяется инструкция DROP TRIGGER. Ее синтаксис:

```
DROP TRIGGER [владелец.]имя_ триггера  
[, ... n]
```

Удаление триггера необязательно, если новый триггер замещает существующий. При удалении таблицы автоматически уничтожаются все связанные с ней объекты, включая триггеры.

Пример удаления триггера Tri\_Dei\_Autnors:

```
DROP TRIGGER Tri_Del_Authors
```

#### Приостановка и возобновление работы триггеров

Часто бывает необходимым отключить на некоторое время работу триггера без его фактического удаления. Этого можно достигнуть используя конструкцию

ALTER TABLE <имя\_таблицы> DISABLE TRIGGER <имя триггера> - для отключения триггера и  
ALTER TABLE <имя\_таблицы> ENABLE TRIGGER <имя триггера> - для возобновления его работы

#### Задание 1.

Разработать триггер, который бы удалял запись о книге, в том случае если удаляется последний экземпляр данной книги. Для какой таблицы Вы будете писать этот триггер? При написании триггера помним, что с таблицей книги у нас связаны таблицы авторы и системный каталог. Однако они связаны отношением «многие-ко-многим», для чего используются связующие таблицы. Удалить данные о книге нельзя, если на нее есть ссылки в этих связующих таблицах. Предусмотрите предварительное удаление данных из связующих таблиц.

Проверить работу данного триггера.

#### Технология работы.

Помним, что триггеры – это системные хранимые процедуры, которые связаны с конкретной таблицей. Для вызова редактора триггеров необходимо выделить таблицу, по правой кнопке контекстного меню перейти в раздел «Все задачи» и далее в «Manage triggers» и Вы попадаете в редактор триггеров (см. рис. 1).



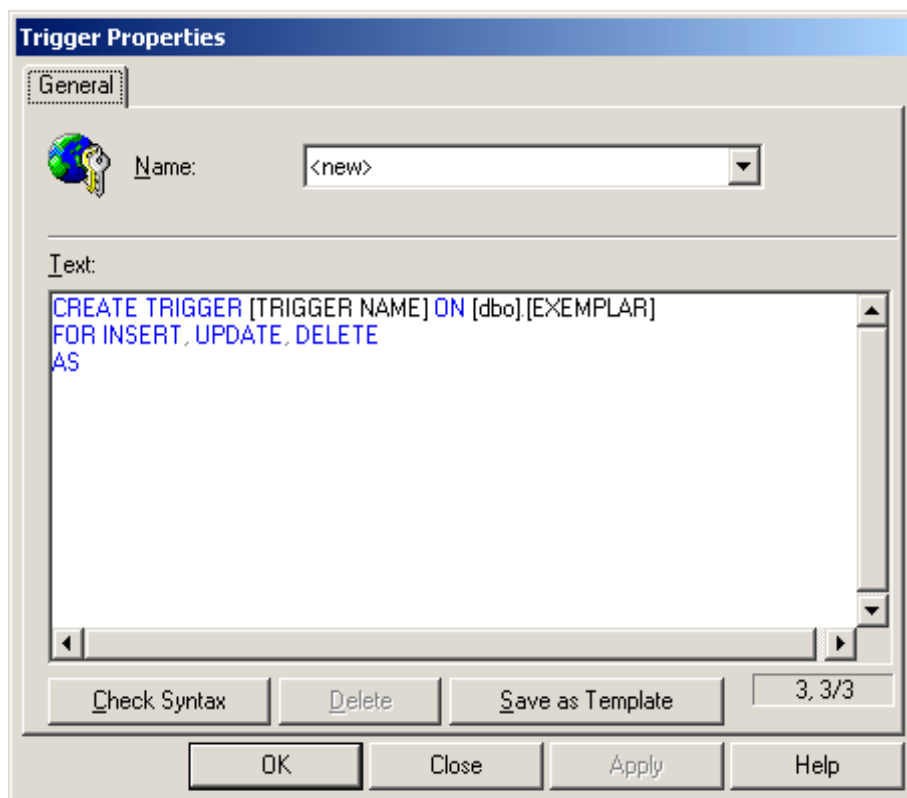


Рис. 1 Начальное состояние редактора триггеров при создании нового триггера.

#### Задание 2.

Разработать триггер, который бы не позволял удалить экземпляр книги, если этот экземпляр в данный момент находится на руках у читателя. Для отмены команды удаления применить команду отката транзакций ROLLBACK .

Проверить работу триггера в независимом режиме, для этого попробовать удалить не последний экземпляр книги, который имеет отметку о том, что он находится у читателя.

Попробовать удалить экземпляр книги, который не находится на руках у читателя.

Проверить работу двух триггеров, для этого попробовать удалить последний экземпляр книги, который находится на руках у читателя.

#### Задание 3.

Разработать триггер, который бы контролировал выдачу книг читателю и при превышении количества 3-х не сданных книг на руках не позволял бы выдать данному читателю еще одну книгу.

#### Задание 4.

Разработать триггер, который бы добавлял один экземпляр при вводе новой книги. Действительно мы определили, что книги у нас в каталоге присутствуют только в том случае, если они есть в нашей библиотеке. Поэтому при вводе новой книги в таблицу экземпляр должен добавляться один экземпляр данной книги.

#### Задание 5.

Разработать триггер типа INSTEAD OF для таблицы «Читатели». Данный триггер должен проверять, есть ли информация хотя бы об одном из телефонов для оперативной связи с читателем, и если такой информации нет, то не вводить данные о читателе.

#### Задание 6

Разработать триггер, который при изменении поля, символизирующего присутствие экземпляра книги в библиотеке, например YES\_NO, со значения '1' на значение '0' автоматически заменит бы значение в поле «даты выдачи», «дата возврата» и номер читательского билета на неопределенное.

#### Задание 7

Разработать триггер, который не позволяет удалить читателя, если за ним числится хотя бы одна книга из библиотеки.

#### Задание 8.

Разработать триггер, который при удалении экземпляра книги проверял бы, сколько экземпляров данной книги осталось в библиотеке, и если остался только один экземпляр, то повышал бы стоимость данной книги на 15% как редкой и ценной.

### Практическое занятие «Создание запросов и процедур на изменение структуры базы данных»

Цель: научиться создавать запросы и формировать выборку данных SQL.

Команда SELECT – выборка данных

Общий синтаксис:

```
SELECT [{ ALL | DISTINCT }] { список_вывода | * }  
FROM имя_таблицы1 [ алиас1 ]      [, имя_таблицы2 [ алиас2 ],...] [ WHERE  
    условие_отбора_записей ]  
[ GROUP BY { имя_поля | выражение } ,... ] [ HAVING      условие_отбора_групп ]  
[ UNION [ALL] SELECT ...]  
[ ORDER BY имя_поля1 | целое [ ASC | DESC ] [, имя_поля2 | целое [ ASC | DESC ],...]];
```

Примеры:

```
select * from departs;
```

```
select  name, post      from emp;
```

Общий синтаксис списка вывода:

[ {all | distinct} ] { \* | выражение1 [алиас1] [, выражение2 [алиас2] ,...} Список ввода находится между ключевыми словами SELECT и FROM.

- Вывести все поля всех записей из таблицы Проекты (Project):

```
select * from project;
```

2. Вывести список сотрудников с указанием их должности и № отдела:

```
select depno, name, post from emp;
```

3. Вывести список сотрудников с указанием их должности и зарплаты:

```
select name 'ФИО', post 'Должность', salary*0.87 'Зарплата' from emp;
```

Установить другой формат вывода даты:

```
alter session set nls_date_format = 'dd/mm/yyyy';
```

1. Вывести должности и оклады сотрудников:

```
select post, salary from emp;
```

2. Вывести должности и оклады сотрудников без повторов: `select DISTINCT post, salary from emp;`

3. Вывести отделы и должности сотрудников без повторов: `select DISTINCT depno, post from emp;`

4. Задание: вывести список сотрудников с указанием ФИО, даты рождения и адреса.

```
select name 'ФИО', born 'Дата рождения', adr 'Адрес'
from emp;
```

Упорядочение результата: `order by`

1. Вывести данные из таблицы Проекты в порядке даты начала проекта:

```
select *
from Project order by dbegin;
```

2. Упорядочить список сотрудников по отделам и по ФИО:

```
select depno, name, post
from emp
order by depno, name; -- order by 1,2;
```

3. Упорядочить сотрудников по зарплате (от большей к меньшей):

```
select name 'ФИО', post 'Должность', salary 'Зарплата'
from emp
order by 3 DESC;
```

4. Упорядочить данные об отделах, должностях и зарплатах:

```
select depno 'Номер отдела', post 'Должность', salary 'Зарплата'
from emp
order by 1, 3 DESC, 2;
```

Выбор данных из таблицы (селекция)

WHERE – содержит условия выбора отдельных записей. Условие является логическим выражением и может принимать одно из 3-х значений:

- TRUE – истина,
- FALSE – ложь,
- NULL – неизвестное, неопределённое значение (интерпретируется как ложь).

Условие формируется путём применения различных операторов и предикатов.

Операторы сравнения:

= равно, <>, != не равно, > больше,  
 >= больше или равно, <= меньше или равно, < меньше.

1. Вывести список сотрудников 2-го отдела:

```
select * from emp where depno = 2;
```

2. Вывести список текущих проектов:

```
select * from project
```

```
where dend > sysdate;
```

-- sysdate – функция, возвращающая текущую дату

Логические операторы

Для формирования условий используются следующие логические операторы:

AND – логическое произведение (И), OR – логическая сумма (ИЛИ), NOT – отрицание (НЕ).

Операция И: Операция ИЛИ:

Операция НЕ:

a NOT a

0 1

1 0

1. Вывести список сотрудников 2-го отдела с зарплатой больше 30000 рублей:

```
select * from emp
```

```
where depno = 2 AND salary > 30000 ;
```

2. Вывести список сотрудников-мужчин, родившихся после 1979 года:

```
select * from emp
```

```
where born > '31/12/1979' AND sex = 'м';
```

3. Вывести список сотрудников 2-го и 5-го отделов:

```
select * from emp
```

```
where depno=2 OR depno = 5;
```

4. Вывести список сотрудников 2-го и 5-го отделов в зарплатой не менее 30000:

```
select * from emp
```

```
where (depno=2 OR depno = 5) AND salary >= 30000 ;
```

5. Вывести список всех сотрудников, кроме сотрудников 2-го и 5-го:

```
select * from emp
where NOT (deptno=2 OR deptno = 5);
```

Задание 1: вывести список текущих проектов стоимостью более 2 млн. рублей.

```
select *
from project
where end_date > sysdate AND cost > 2000000;
```

Задание 2: вывести список сотрудников, работающих в должностях 'инженер' и 'ведущий инженер'.

```
select *
from emp
where job = 'инженер' OR job = 'ведущий инженер' ;
```

Задание 3: вывести список сотрудников, работающих в должности 'охранник', с зарплатой более 20000 рублей.

```
select *
from emp
where job = 'охранник' AND salary > 20000;
```

Предикат вхождения в список значений:

имя\_поля IN ( значение1 [, значение2,... ] )

выражение IN ( значение1 [, значение2,... ] )

Примеры:

- Список сотрудников отделов 5, 8 и 9:

```
select *
from emp
where deptno IN ( 5, 8, 9 ) ;
```

- Список сотрудников, работающих в должностях 'инженер' и 'ведущий инженер' :

```
select *
from emp
where job IN ( 'инженер', 'ведущий инженер' );
```

Предикат вхождения в диапазон:

имя\_поля BETWEEN минимальное\_значение AND максимальное\_значение

выражение BETWEEN минимальное\_значение AND максимальное\_значение

Минимальное значение должно быть меньше либо равно максимальному.

Примеры:

- Список всех сотрудников со 2-го по 5-й отделы:

```
select *  
from emp  
where depno BETWEEN 2 AND 5 ;
```

- Список сотрудников с чистой зарплатой от 20 до 30 тысяч рублей:

```
select *  
from emp  
where salary*0.87 BETWEEN 20000 AND 30000;
```

Предикат поиска подстроки: имя\_поля LIKE 'шаблон'

Этот предикат применяется только к полям типа CHAR и VARCHAR. Возможно использование шаблонов:

'\_' – один любой символ,

'%' – произвольное количество любых символов (в т.ч., ни одного).

Примеры:

- Список всех сотрудников-экономистов:

```
select * from emp  
where post LIKE '%экономист%';
```

- Список всех инженеров-специалистов (кроме просто инженеров):

```
select * from emp  
where post LIKE 'инженер_%';
```

Экранировать специальное значение символов '\_' и '%' можно так:

```
where <строка> LIKE '_#%' ESCAPE '#';
```

Символ экранирования (escape) может быть любым. В примере первый символ % будет искаться как символ, а второй имеет специальное значение.

Предикат поиска неопределенного значения:

значение IS [NOT] NULL

Если значения является неопределенным (NULL), то предикат IS NULL выдаст истину, а предикат IS NOT NULL – ложь.

Примеры:

- Список всех сотрудников, у которых нет телефона (номер телефона неопределен):

```
select *  
from emp  
where phone IS NULL ;
```

- Список все проекты, у которых определена стоимость:

```
select *  
from project  
where cost IS NOT NULL ;
```

#### Использование предикатов

Задание 1: вывести список сотрудников, которых зовут 'ЮРИЙ'.

```
select *  
from emp  
where name LIKE '%ЮРИЙ%';
```

Задание 2: вывести список проектов стоимостью от 1 до 2 млн. рублей.

```
select *  
from project  
where cost BETWEEN 1000000 AND 2000000;
```

Задание 3: вывести список сотрудников, которые являются начальниками отделов.

```
select *  
from emp  
where post LIKE 'нач%отдел%';
```

#### Агрегирующие функции

COUNT – подсчёт количества строк (значений). Применяется к записям и полям любого типа. Имеет 3 формата вызова:

- ☐ count (\*) – количество строк результата;
- ☐ count (имя\_поля) – количество значений указанного поля, не являющихся NULL-значениями.
- ☐ count (distinct имя\_поля) – количество разных не-NULL значений

указанного поля.

MAX, MIN – определяет максимальное (минимальное) значение указанного поля в результирующем множестве. Применяется к полям любого типа.

SUM – определяет арифметическую сумму значений указанного числового поля в результирующем множестве записей.

AVG – определяет среднее арифметическое значений указанного числового поля в результирующем множестве записей. Не учитывает NULL-значения, и сумма значений поля делится на количество определённых значений.

#### Примеры использования агрегирующих функций

1. Вывести максимальную и минимальную стоимость проектов:

```
select max(cost) "Максимальная цена", min(cost) "Минимальная цена"  
from project;
```

2. Вывести сумму зарплаты сотрудников 8-го отдела:

```
select sum(salary)  
from emp
```

```
where depno = 8;
```

3. Вывести среднюю зарплату сотрудниц предприятия:

```
select avg(salary)  
from emp
```

```
where sex = 'Ж';
```

4. Вывести даты начала работы над первым проектом и завершения работы над последним проектом:

```
select min(dbegin), max(dend)  
from project;
```

#### Примеры использования функции COUNT

1. Вывести количество сотрудников:

```
select count(*)  
from emp;
```

2. Вывести количество сотрудников с телефонами:

```
select count( phone )  
from emp;
```

3. Вывести количество разных должностей сотрудников:

```
select count (DISTINCT post) from emp;
```

4. Задание: вывести количество сотрудников 6-го отдела.

```
select count(*)  
from emp  
where depno = 6;
```

#### Группировка данных: предложение GROUP BY

Агрегирующие функции обычно используются совместно с предложением GROUP BY.

Например, следующая команда считает количество сотрудников по отделам:

```
select depno, count(*)
```



```
from emp
group by depno;
```

#### Примеры использования GROUP BY

1. Вывести минимальную и максимальную зарплату в каждом отделе:

```
select depno, MIN(salary) minsal, MAX(salary) maxsal from emp
group by depno;
```

2. Вывести количество разных должностей в каждом отделе:

```
select depno, COUNT(distinct post) cnt from emp
group by depno;
```

3. Посчитать сумму зарплат в каждом отделе:

```
select depno, SUM(salary) allsal from emp
group by depno;
```

4. Посчитать среднюю зарплату по каждой должности:

```
select post, AVG(salary) avgsal from emp
group by post;
```

#### Использование GROUP BY

##### Правило использования GROUP BY :

В списке вывода при использовании GROUP BY могут быть указаны только функции агрегирования, константы и поля, перечисленные в GROUP BY.

Если включить в список выбора поля, не указанные в GROUP BY, то СУБД не будет выполнять такой запрос и выдаст ошибку "нарушение условия группирования" (not a GROUP BY expression).

Например, нельзя получить сведения о том, у каких сотрудников самая высокая зарплата в своём отделе с помощью такого запроса:

```
select depno, name, max(salary) as max_sal from emp
group by depno;
```

Этот запрос синтаксически неверен!

#### Группировка по нескольким полям

1. Сумма зарплат по отделам и по должностям:

```
select depno, post, count(*), sum(salary) from emp
group by depno, post;
```

2. Количество мужчин и женщин по отделам:

```
select depno, sex, count(*) from emp  
group by depno, sex;
```

Задание: вывести информацию о зарплате и количестве сотрудников, которые получают такую зарплату.

```
select salary, count(*) from emp  
group by salary;
```

### **Практическое занятие «Работа с журналом аудита базы данных»**

Цель работы: научиться работать с журналом аудита базы данных Oracle.

Управление доступом к базам данных

Аудит баз данных

Иногда необходимо регистрировать информацию об операциях, выполняемых в системе баз данных. Это называется аудитом (audit). Например, может потребоваться статистика о том, какие действия в системе выполняют пользователи. Аудит можно применять также для слежения за базой данных с целью обнаружения потенциальных нарушений ее защиты.

#### ***Избирательный аудит***

Средство аудита Oracle можно включать и выключать. При включении аудита требуется определенный расход ресурсов, необходимый для генерации записей аудита. Чтобы свести к минимуму работу, выполняемую Oracle для аудита базы данных, нужно точно определить, для чего нужен аудит:

- Можно выполнять аудит конкретных SQL-операторов безотносительно к конкретным объектам. Например, можно просить Oracle генерировать запись аудита всякий раз, когда какой-либо пользователь выполняет оператор DROP TABLE, и не учитывать, к какой таблице относится этот оператор.
- Можно выполнять аудит использования мощных системных привилегий. Например, просить Oracle генерировать запись аудита всякий раз, когда какой-либо пользователь применяет системную привилегию SELECT ANY TABLE для запроса к таблице базы данных.
- Можно выполнять аудит определенных SQL-операторов для конкретных объектов базы данных. Например, просить Oracle генерировать запись аудита всякий раз, когда какой-либо пользователь удаляет запись из таблицы SALES.CUSTOMERS.
- Для каждой разрешенной опции аудита можно просить Oracle генерировать запись аудита для успешного, неуспешного или любого выполнения операторов. Более того, можно установить каждую разрешенную опцию аудита для всех либо для некоторых пользователей базы данных.
- Для каждой разрешенной опции аудита можно просить Oracle генерировать запись аудита один раз за все время сеанса связи с базой данных независимо от того, сколько раз за это время выполняется оператор, подлежащий аудиту. И наоборот, можно просить Oracle генерировать запись аудита всякий раз, когда во время сеанса выполняется оператор, подлежащий аудиту.

## Записи аудита и журнал аудита

Oracle генерирует записи аудита только после того, как аудит разрешен и для него установлены соответствующие опции. Например, можно установить опции аудита для конкретных операторов, объектов и пользователей. В каждую запись аудита включается информация о контролируемом операторе: какая операция выполнялась, какой пользователь ее выполнял, а также дата и время ее выполнения.

Oracle хранит сгенерированные записи аудита в журнале аудита (audit trail), называемом также контрольным журналом. Журнал аудита — это место хранения информации аудита. В Oracle разрешается сохранять генерируемые записи аудита либо в журнале аудита базы данных, либо в таком же журнале операционной системы, в которой работает Oracle Server. При использовании журнала аудита базы данных можно легко просматривать и находить записи аудита с помощью SQL-запросов и предварительно созданных для этого журнала представлений словаря данных. При использовании журнала аудита операционной системы можно объединять информацию аудита базы данных с информацией аудита операционной системы.

### Установка опций аудита

После того как администратор разрешает аудит на системном уровне, можно устанавливать опции аудита при помощи SQL-команды AUDIT.

Синтаксис её таков:

```
AUDIT {sql_statement_clause | scheme_object_clause}
[BY {SESSION | ACCESS} ]
[WHENEVER [NOT] SUCCESSFUL ];
```

где sql\_statement\_clause - это

```
{ {statement_option | ALL} [, ...]
| {system_privilege | ALL PRIVILEGES} [, ...]
}
[BY {proxy [,proxy]... | user [,user]...}]
```

А обозначение scheme\_object\_clause - это

```
{object_option [,object_option]... | ALL}
ON { [schema.]object | DIRECTORY directory_name
| DEFAULT }
```

На примере, ниже показано, как включить некоторые опции аудита:

```
-- Следующий оператор аудита фиксирует все неуспешные
-- попытки применения системной привилегии
-- SELECT ANY TABLE пользователями SROGERS и JGIBBS.
-- Oracle генерирует только одну запись аудита за
-- сеанс независимо от того, сколько раз встречается
```

```

-- критерий аудита.
AUDIT SELECT ANY TABLE
  BY srogers, jgibbs
  BY SESSION WHENEVER NOT SUCCESSFUL;
-- Следующий оператор фиксирует все успешные и
-- неуспешные попытки выполнения оператора SELECT
-- для таблицы SALES.CUSTOMERS.
-- Oracle генерирует запись аудита всякий раз,
-- когда встречается критерий аудита.
AUDIT SELECT ON sales.customers
  BY ACCESS;

```

Если аудит какой-либо операции больше не нужен, можно выключить его при помощи SQL-команды NOAUDIT. Синтаксис:

```

NOAUDIT { sql_statement_clause

    [,sql_statement_clause] ...

    | scheme_object_clause

    [,scheme_object_clause] ...

}

[WHENEVER [NOT] SUCCESSFUL ];

```

Пример.

```
NOAUDIT SELECT ON sales.customers;
```

### Практическое занятие «Мониторинг нагрузки сервера»

Цель: узнать, как добавить SignalR в приложение и отправлять уведомления с сервера на подключенные клиенты с помощью концентраторов. Кроме того, вы узнали, как масштабировать приложение с помощью компонента объединительной платы при развертывании приложения в нескольких ЭКЗЕМПЛЯРАХ служб IIS.

Задание.

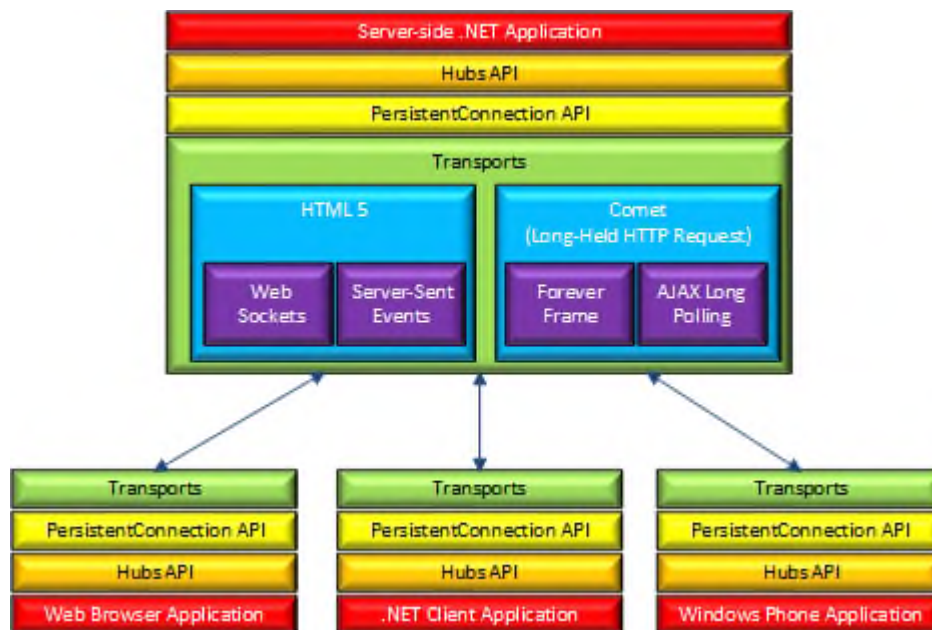
Веб-приложения в режиме реального времени позволяют отправлять содержимое на стороне сервера в подключенные клиенты, как это происходит в режиме реального времени. Для разработчиков ASP.NET ASP.NET SignalR — это библиотека для добавления в приложения веб-функций в режиме реального времени. Она использует преимущества нескольких транспортов,

автоматически выбирая оптимальную доступную транспортную передачу, используя оптимальный доступный транспорт клиента и сервера. Он использует преимущества WebSocket, API HTML5, который обеспечивает двунаправленный обмен данными между браузером и сервером.

SignalR также предоставляет простой, высокоуровневый API для выполнения СЕРВЕРНОЙ RPC-службы (вызов функций JavaScript в браузерах клиентов из серверного кода .NET) в приложении ASP.NET, а также добавление полезных обработчиков для управления подключениями, таких как события подключения и отключения, группирование соединений и авторизация.

SignalR является абстракцией для некоторых транспортов, необходимых для работы в режиме реального времени между клиентом и сервером. Подключение SignalR запускается как HTTP, а затем передается в подключение WebSocket, если оно доступно. WebSocket — это идеальный транспорт для SignalR, так как он обеспечивает наиболее эффективное использование памяти сервера, имеет наименьшую задержку и обладает самыми основными функциями (например, полным дуплексным обменом данными между клиентом и сервером), но обладает самыми строгими требованиями: для WebSocket требуется, чтобы сервер использовал Windows Server 2012 или Windows 8, а также .NET Framework 4,5. Если эти требования не выполняются, то SignalR попытается использовать другие транспорты для подключения (например, длительный опрос Ajax).

API SignalR содержит две модели взаимодействия между клиентами и серверами: постоянные подключения и концентраторы. Соединение представляет собой простую конечную точку для отправки сообщений одного получателя, сгруппированных или широковещательных. Концентратор — это высокоуровневый конвейер, построенный на основе API подключения, который позволяет клиенту и серверу вызывать методы напрямую.



Все примеры кода и фрагментов включены в набор средств для обучения Web Camp, <https://github.com/Microsoft-Web/WebCampTrainingKit/releases/tag/v2015.10.13> выпущенный в октябре 2015. Обратите внимание, что ссылка на этот установщик на этой странице больше не работает; Вместо этого используйте одну из ссылок в разделе активы.

Чтобы выполнить упражнения в этой практической работе, сначала необходимо настроить среду.

1. Откройте окно проводника Windows и перейдите к исходной папке лаборатории.
2. Щелкните правой кнопкой мыши Setup. cmd и выберите Запуск от имени администратора, чтобы запустить процесс установки, который будет настраивать среду и установить фрагменты кода Visual Studio для этой работы.
3. Если отображается диалоговое окно Контроль учетных записей пользователей, подтвердите действие для продолжения.

#### Примечание

Убедитесь, что проверены все зависимости для этой работы перед запуском программы установки.

### Использование фрагментов кода

На протяжении всего лабораторного документа вы получите инструкции по вставке блоков кода. Для удобства большая часть этого кода предоставляется как Visual Studio Code фрагменты, к которым можно получить доступ из Visual Studio 2013, чтобы не добавлять их вручную.

## Примечание

Каждое упражнение сопровождается начальным решением, расположенным в начальной папке упражнения, которое позволяет выполнять каждое упражнение независимо от других. Имейте в виду, что фрагменты кода, добавленные во время упражнения, отсутствуют в этих начальных решениях и могут не работать, пока вы не завершите упражнение. В исходном коде для упражнения также будет найдена Конечная папка, содержащая решение Visual Studio с кодом, полученным в результате выполнения шагов в соответствующем упражнении. Эти решения можно использовать в качестве руководства, если вам нужна дополнительная помощь во время работы с этой практической работой.

При первом запуске Visual Studio необходимо выбрать одну из предварительно определенных коллекций параметров. Каждая предопределенная коллекция разработана для соответствия определенному стилю разработки и определяет макеты окон, поведение редактора, фрагменты кода IntelliSense и параметры диалоговых окон. Процедуры в этом лабораторном занятии описывают действия, необходимые для выполнения данной задачи в Visual Studio при использовании коллекции общих параметров разработки. При выборе другой коллекции параметров для среды разработки могут возникнуть различия в действиях, которые необходимо учитывать.

## **Упражнение 1. Работа с данными в режиме реального времени с помощью SignalR**

Хотя беседа часто используется в качестве примера, вы можете сделать многое больше с помощью веб-функций в режиме реального времени. Каждый раз, когда пользователь обновляет веб-страницу для просмотра новых данных, или страница реализует длительный опрос Ajax для получения новых данных, вы можете использовать SignalR.

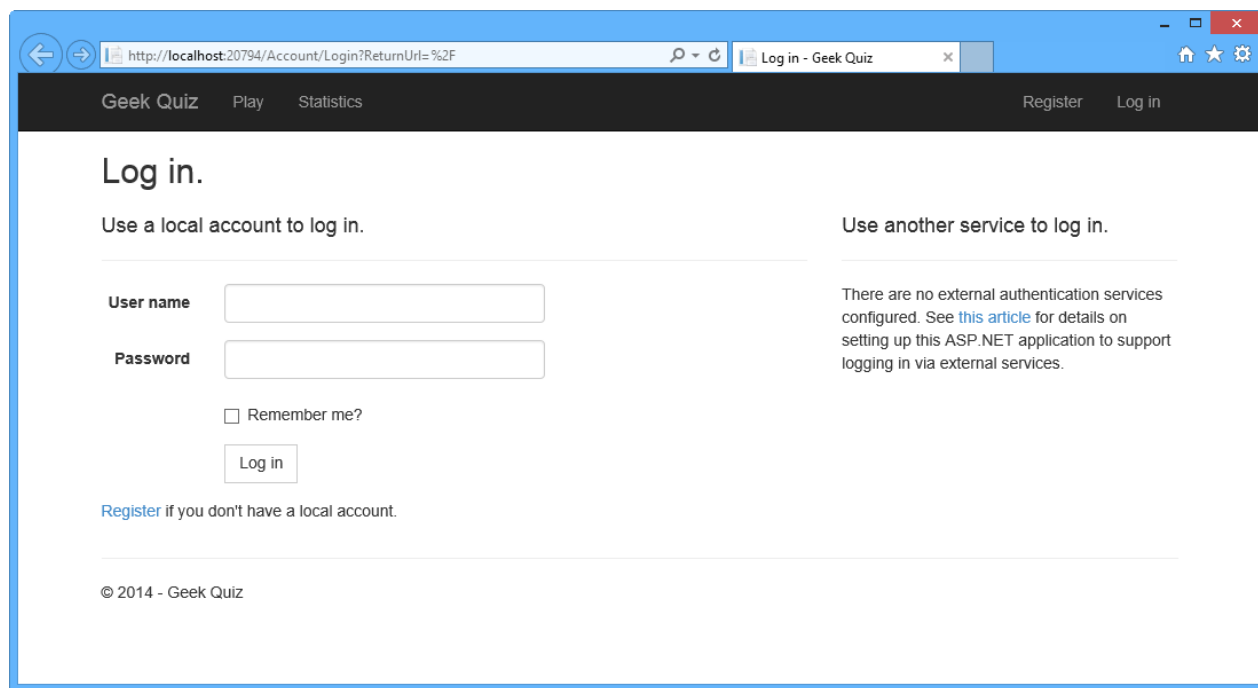
SignalR поддерживает функцию принудительной или широковещательной передачи сервера; он автоматически обрабатывает управление подключениями. В классических HTTP-подключениях для обмена данными между клиентом и сервером устанавливается соединение повторно для каждого запроса, но SignalR обеспечивает постоянное подключение между клиентом и сервером. В SignalR серверный код вызывает клиентский код в браузере с помощью удаленных вызовов процедур (RPC), а не модели «запрос-ответ», которая уже известна.

В этом упражнении вы настроите приложение " Викторина ", которое будет использовать SignalR для вывода панели мониторинга статистики с обновленными метриками без необходимости обновлять всю страницу.

#### Задача 1. Обзор страницы "Статистика головоломки"

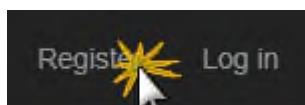
В этой задаче вы просматриваете приложение и проверите, как отображается страница статистики, и как можно улучшить способ обновления информации.

1. Откройте Visual Studio Express 2013 для Web и откройте решение жееккуиз. sln , расположенное в папке Source\Ex1-WorkingWithRealTimeData\Begin .
2. Чтобы запустить решение, нажмите клавишу F5. Страница входа должна появиться в браузере.



#### Запуск решения

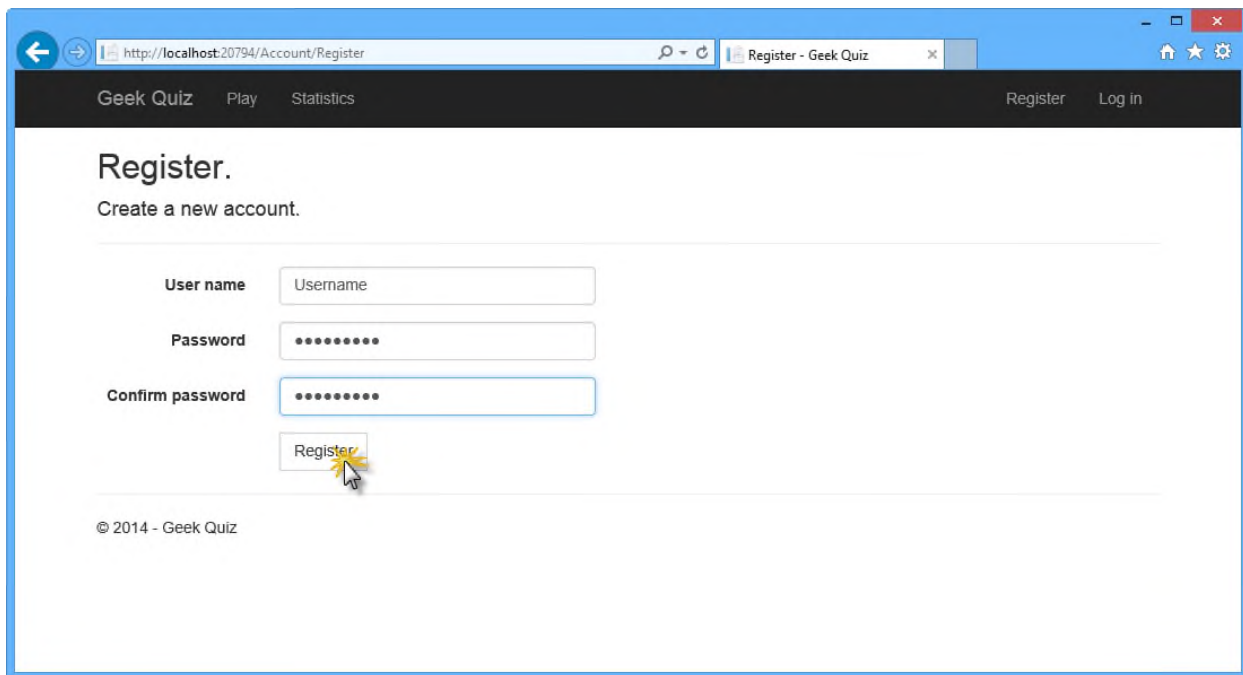
3. Нажмите кнопку зарегистрировать в правом верхнем углу страницы, чтобы создать нового пользователя в приложении.



#### Ссылка для регистрации



4. На странице Регистрация введите имя пользователя и пароль, а затем нажмите кнопку зарегистрировать.

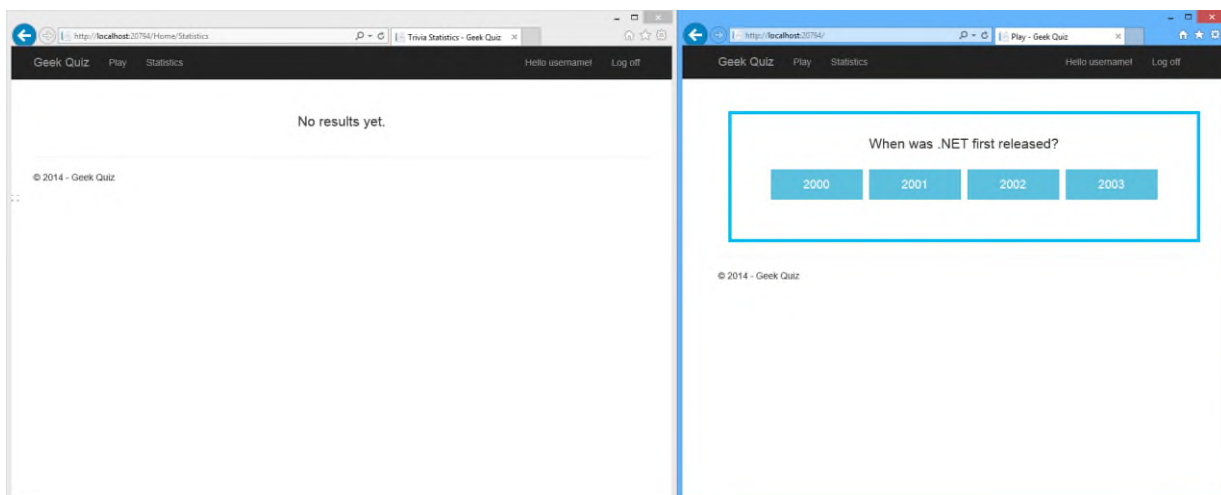


The screenshot shows a web browser window with the address bar displaying `http://localhost:20794/Account/Register`. The page title is "Register - Geek Quiz". The page has a dark header with "Geek Quiz", "Play", and "Statistics" links, and "Register" and "Log in" buttons. The main content area is titled "Register." with the subtitle "Create a new account." Below this, there are three input fields: "User name" with the placeholder "Username", "Password" with masked characters, and "Confirm password" with masked characters. A "Register" button is located below the password fields. A mouse cursor is hovering over the "Register" button. At the bottom left, there is a copyright notice: "© 2014 - Geek Quiz".

#### Регистрация пользователя

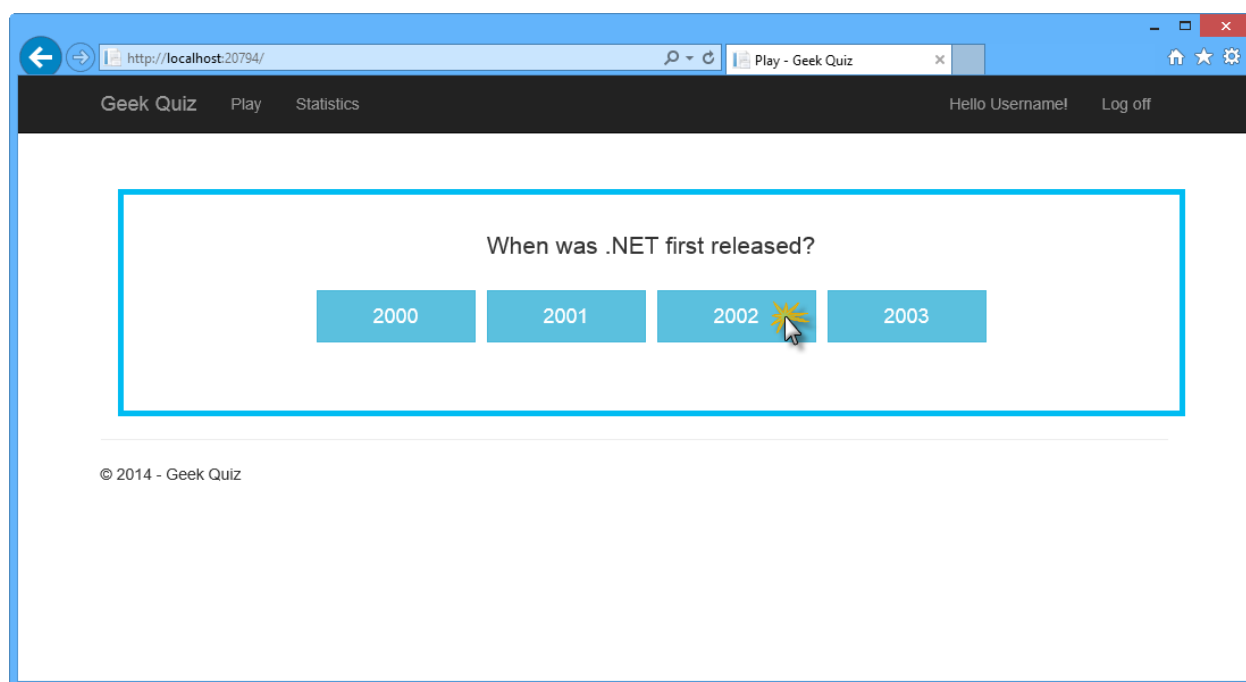
5. Приложение регистрирует новую учетную запись, и пользователь проходит проверку подлинности и перенаправляется обратно на домашнюю страницу, где отображается первый контрольный вопрос.

6. Откройте страницу Статистика в новом окне и разместите страницу Домашняя страница и Статистика параллельно.



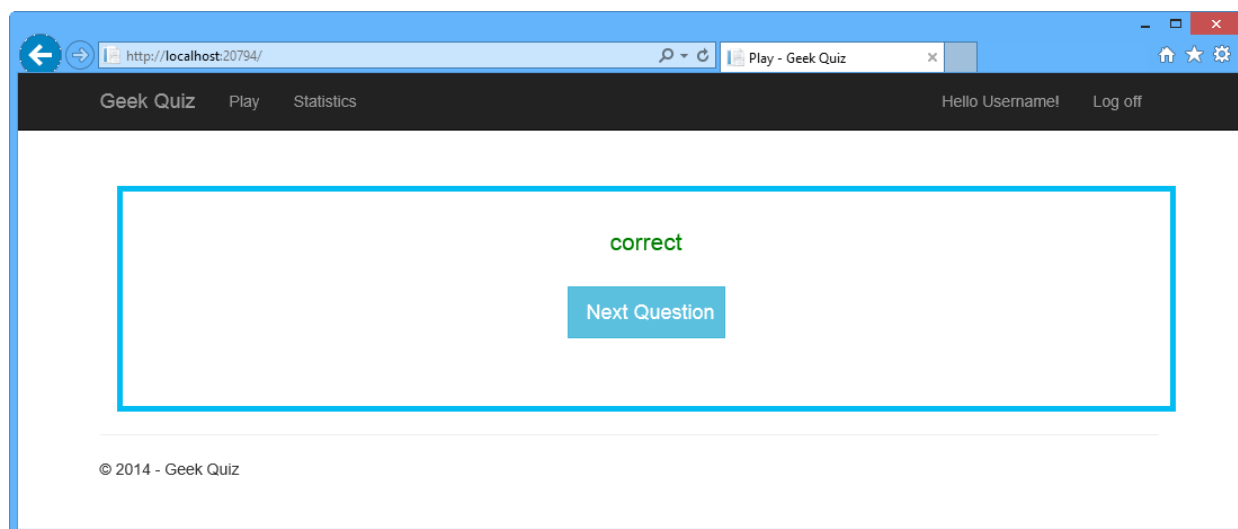
#### Параллельные окна

7. На домашней странице ответьте на вопрос, щелкнув один из вариантов.



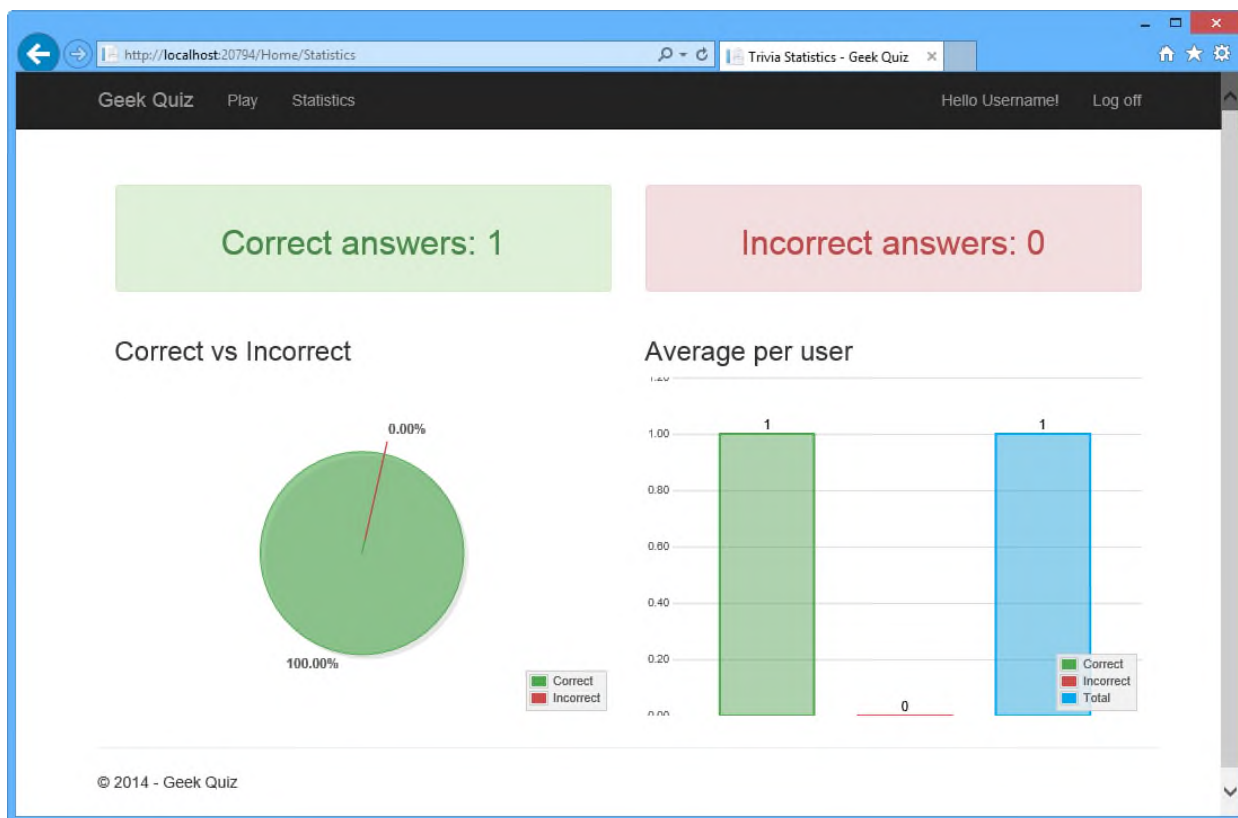
Ответ на вопрос

8. После нажатия одной из кнопок появится ответ.



Ответ на вопрос правильно

9. Обратите внимание, что сведения, приведенные на странице Статистика, устарели. Обновите страницу, чтобы просмотреть обновленные результаты.



### Страница «Статистика»

10. Вернитесь в Visual Studio и завершите отладку.

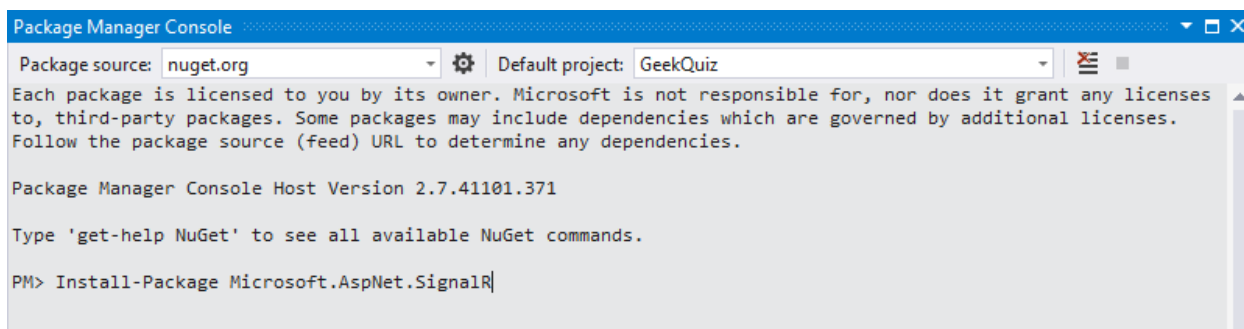
### Задача 2. Добавление SignalR в головоломку для отображения в сети диаграмм

В этой задаче вы добавите в решение SignalR и автоматически отправите обновления клиентам при отправке на сервер нового ответа.

1. В меню Сервис в Visual Studio выберите Диспетчер пакетов NuGet, а затем щелкните консоль диспетчера пакетов.
2. В окне консоли диспетчера пакетов выполните следующую команду:

PowerShell

```
Install-Package Microsoft.AspNet.SignalR
```



## Установка пакета SignalR

### Примечание

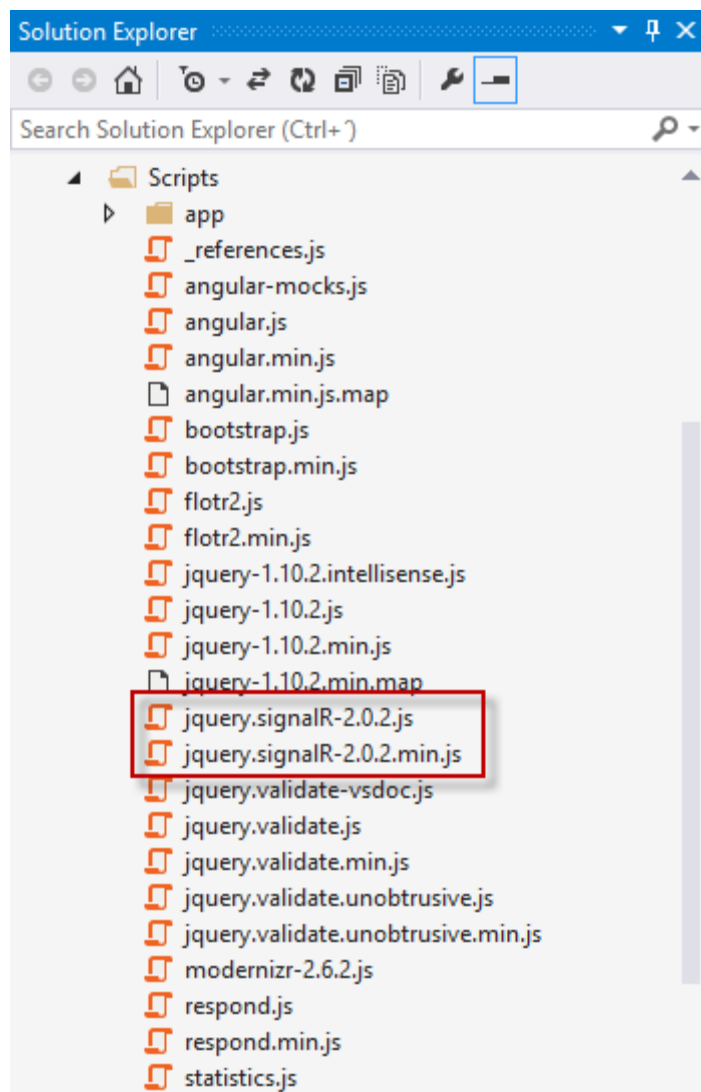
При установке пакетов NuGet SignalR версии 2.0.2 из нового приложения MVC 5 необходимо вручную обновить пакеты OWIN до версии 2.0.1 (или выше) перед установкой SignalR. Для этого можно выполнить следующий скрипт в консоли диспетчера пакетов:

#### PowerShell

```
❑ get-package | where-object { $_.Id -like "Microsoft.Owin*" } | Update-Package
```

В будущих выпусках SignalR зависимости OWIN будут автоматически обновлены.

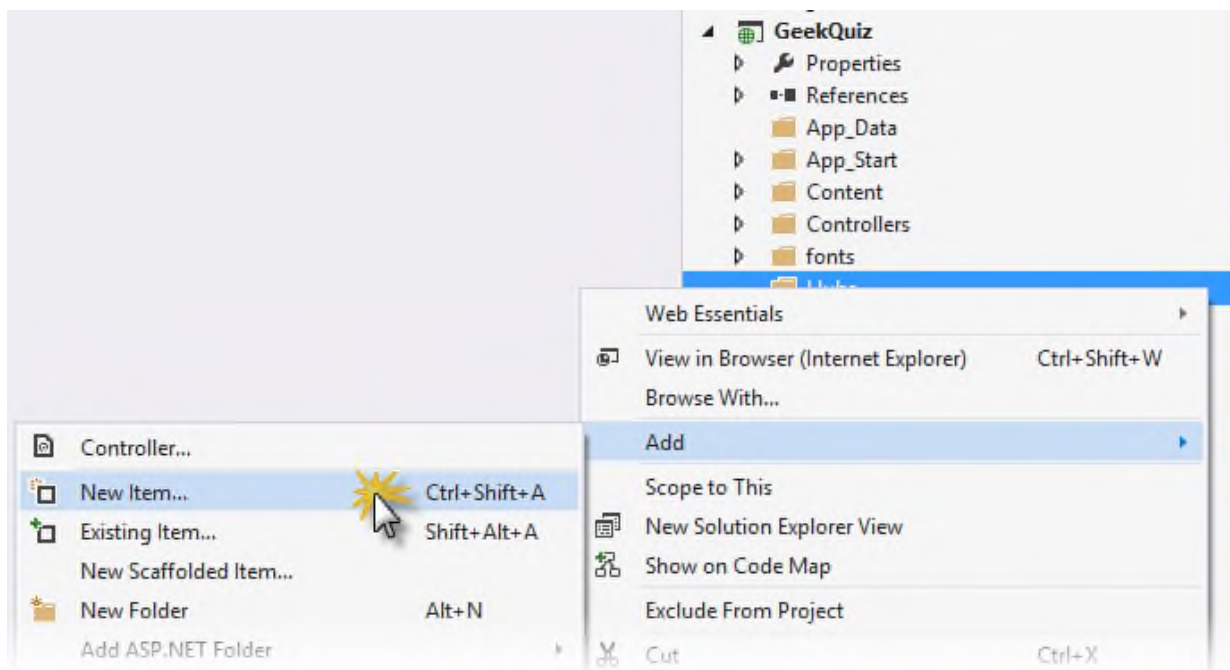
❑ В Обозреватель решений разверните папку « скрипты » и обратите внимание, что файлы SignalR. JS добавлены в решение.



## Ссылки на JavaScript SignalR

В Обзорщике решений щелкните правой кнопкой мыши проект Жееккуиз , выберите добавить | создать папку и назовите ее центры.

Щелкните правой кнопкой мыши папку концентраторы и выберите Добавить | Новый элемент.



### Добавить новый элемент

В диалоговом окне "Добавление нового элемента" выберите C# визуальный элемент | Интернет | Узел SignalR в левой области выберите класс концентратора SignalR (v2) в центральной области, назовите файл StatisticsHub.CS и нажмите кнопку Добавить.

![Диалоговое окно "Добавление нового элемента"](real-time-web-applications-with-signalr/\_static/image12.png "Диалоговое окно "Добавление нового элемента")

### Диалоговое окно "Добавление нового элемента"

- ☐ Замените код в классе статистикшуб следующим кодом.

(Фрагмент кода — реалтимесигналр-EX1-статистикшубкласс)

C#

```

namespace GeekQuiz.Hubs
{
    using Microsoft.AspNet.SignalR;

    public class StatisticsHub : Hub
    {
    }
}

```

Откройте Startup.CS и добавьте следующую строку в конец метода конфигурации .

(Фрагмент кода — реалтисигналр-EX1-мапсигналр)

C#

```
□ public void Configuration(IApplicationBuilder app)
{
    this.ConfigureAuth(app);
    app.MapSignalR();
}
```

Откройте страницу StatisticsService.CS в папке Services и добавьте следующие директивы using.

(Фрагмент кода — реалтисигналр-EX1-усингдирективес)

C#

```
using Microsoft.AspNet.SignalR;
using GeekQuiz.Hubs;
```

□ Чтобы уведомить подключенных клиентов об обновлениях, сначала необходимо получить объект контекста для текущего соединения. Объект Hub содержит методы для отправки сообщений одному клиенту или вещания всем подключенным клиентам. Добавьте следующий метод в класс статистикссервице для передачи статистических данных.

(Фрагмент кода — реалтисигналр-EX1-нотифюдатесмесод)

C#

```
public async Task NotifyUpdates()
{
    var hubContext =
GlobalHost.ConnectionManager.GetHubContext<StatisticsHub>();
    if (hubContext != null)
    {
        var stats = await this.GenerateStatistics();
        hubContext.Clients.All.updateStatistics(stats);
    }
}
```

## Примечание

В приведенном выше коде вы используете произвольное имя метода для вызова функции на клиенте (т. е. `updateStatistics`). Указанное имя метода интерпретируется как динамический объект, что означает отсутствие в нем IntelliSense или проверку во время компиляции. Выражение вычисляется во время выполнения. При выполнении вызова метода SignalR отправляет клиенту имя метода и значения параметров. Если клиент имеет метод, совпадающий с именем, вызывается этот метод и ему передаются значения параметров. Если на клиенте не найден соответствующий метод, ошибка не возникает. Дополнительные сведения см. в [руководстве по API концентраторов signalr ASP.NET](#).

Откройте страницу `TriviaController.CS` в папке `Controllers` и добавьте следующие директивы `using`.

```
C#  
using GeekQuiz.Services;
```

Добавьте следующий выделенный код в метод действия `POST`.

(Фрагмент кода — реалтисигналр-EX1-нотифюдатескалл)

```
C#  
public async Task<IHttpActionResult> Post(TriviaAnswer answer)  
{  
    if (!ModelState.IsValid)  
    {  
        return this.BadRequest(this.ModelState);  
    }  
  
    answer.UserId = User.Identity.Name;  
  
    var isCorrect = await this.StoreAsync(answer);  
  
    var statisticsService = new StatisticsService(this.db);  
    await statisticsService.NotifyUpdates();  
  
    return this.Ok<bool>(isCorrect);  
}
```



```
}
```

Откройте страницу Statistics.cshtml внутри представлений | Домашняя папка. Откройте раздел Scripts и добавьте следующие ссылки на скрипты в начале раздела.

(Фрагмент кода — реалтисигналр-EX1-сигналрскриптреференцес)

CSHTML

```
@section Scripts {  
    @Scripts.Render("~/Scripts/jquery.signalR-2.0.2.min.js");  
    @Scripts.Render("~/signalr/hubs");  
    ...  
}
```

#### Примечание

При добавлении в проект Visual Studio SignalR и других библиотек сценариев диспетчер пакетов может установить версию файла сценария SignalR, более позднюю по сравнению с версией, приведенной в этом разделе. Убедитесь, что ссылка на скрипт в коде совпадает с версией библиотеки скриптов, установленной в проекте.

□ Добавьте выделенный ниже код, чтобы подключить клиент к концентратору SignalR и обновить данные статистики при получении нового сообщения от концентратора.

(Фрагмент кода — реалтисигналр-EX1-сигналрклиенткоде)

CSHTML

```
@section Scripts {  
    ...  
    <script>  
        ...  
  
        var connection = $.hubConnection();  
        var hub = connection.createHubProxy("StatisticsHub");  
        hub.on("updateStatistics", function (statistics) {  
            statisticsData = statistics;  
  
            $("#correctAnswersCounter").text(statistics.CorrectAnswers);  
        });  
    </script>  
}
```

```

$ ("#incorrectAnswersCounter") .text (statistics.IncorrectAnswers) ;

        showCharts (statisticsData) ;

    });

    connection.start();
</script>
}

```

В этом коде вы создаете прокси-сервер концентратора и регистрируете обработчик событий для прослушивания сообщений, отправленных сервером. В этом случае Вы прослушиваете сообщения, отправленные с помощью метода `updateStatistics` .

### Задача 3. Запуск решения

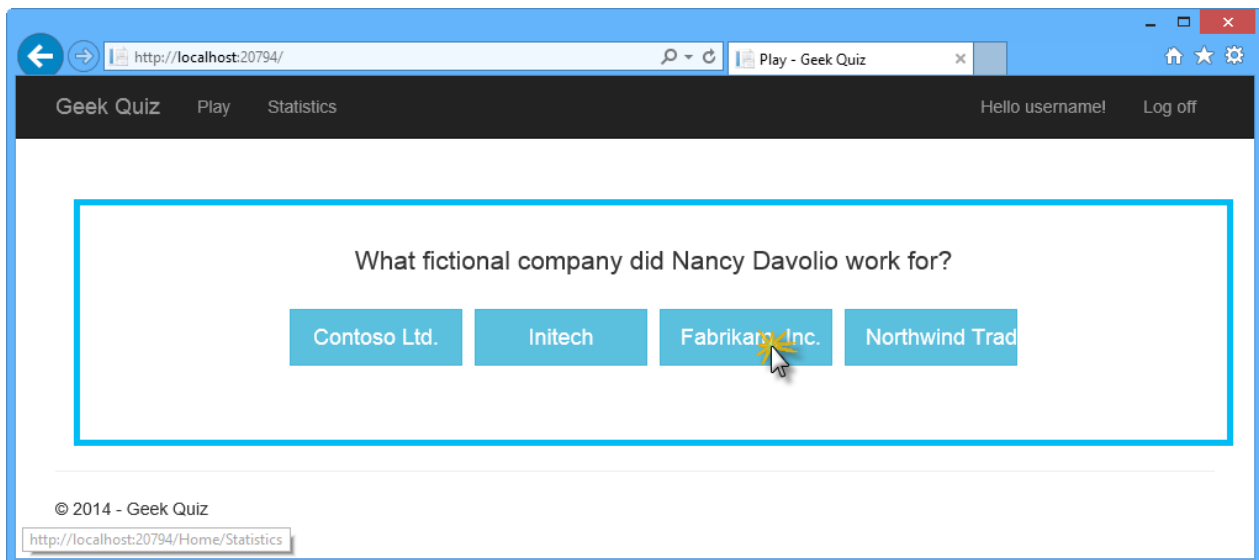
В этой задаче будет запущено решение, чтобы убедиться, что представление статистики автоматически обновляется с помощью SignalR после ответа на новый вопрос.

1. Чтобы запустить решение, нажмите клавишу F5.

#### Примечание

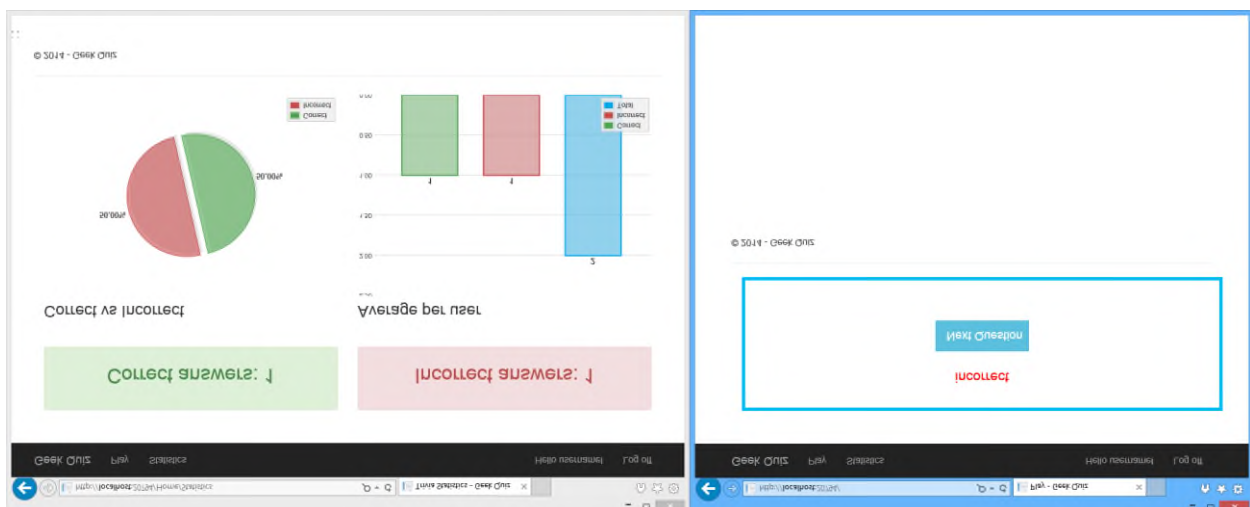
Если вы еще не вошли в приложение, войдите в систему с учетной записью пользователя, созданного в задаче 1.

2. Откройте страницу Статистика в новом окне и разместите страницу Домашняя страница и Статистика параллельно, как показано в задаче 1.
3. На домашней странице ответьте на вопрос, щелкнув один из вариантов.



Ответ на другой вопрос

4. После нажатия одной из кнопок появится ответ. Обратите внимание, что статистические данные на странице обновляются автоматически после ответа на вопрос с обновленной информацией без необходимости обновления всей страницы.



Страница статистики обновлена после ответа

## Упражнение 2. масштабирование с помощью SQL Server

При масштабировании веб-приложения обычно можно выбирать между возможностями масштабирования и масштабирования. Увеличение масштаба означает использование более крупного сервера с большим количеством ресурсов (ЦП, ОЗУ и т. д.), а масштабирование означает добавление дополнительных серверов для работы с нагрузкой. Проблема в последнем заключается

в том, что клиенты могут перенаправляться на разные серверы. Клиент, подключенный к одному серверу, не будет принимать сообщения, отправленные с другого сервера.

Эти проблемы можно решить с помощью компонента, называемого объединительной платой, для пересылки сообщений между серверами. После включения объединительной платы каждый экземпляр приложения отправляет сообщения в объединительную плату, а Объединительная плата перенаправляет их в другие экземпляры приложения.

В настоящее время существует три типа объединительных плат для SignalR:

- Служебная шина Windows Azure. Служебная шина — это инфраструктура обмена сообщениями, позволяющая компонентам обмениваться слабо связанными сообщениями.
- SQL Server. SQL Serverная Объединительная плата записывает сообщения в таблицы SQL. Объединительная плата использует Service Broker для эффективного обмена сообщениями. Однако он также работает, если Service Broker не включена.
- Redis. Redis — это хранилище "ключ — значение" в памяти. Redis поддерживает шаблон публикации/подписки ("Pub/resubscribe") для отправки сообщений.

Каждое сообщение отправляется через шину сообщений. Шина сообщений реализует интерфейс имессажебус , который предоставляет абстракцию публикации и подписки. Объединительные платы работают, заменяя имессажебус по умолчанию на шину, предназначенную для этой объединительной платы.

Каждый экземпляр сервера подключается к объединительной плате через шину. При отправке сообщения он переходит на объединительную плату, а Объединительная плата отправляет его на каждый сервер. Когда сервер получает сообщение от объединительной платы, он сохраняет сообщение в его локальном кэше. Затем сервер доставляет сообщения клиентам из своего локального кэша.

Дополнительные сведения о работе объединительной платы SignalR см. в этой статье.

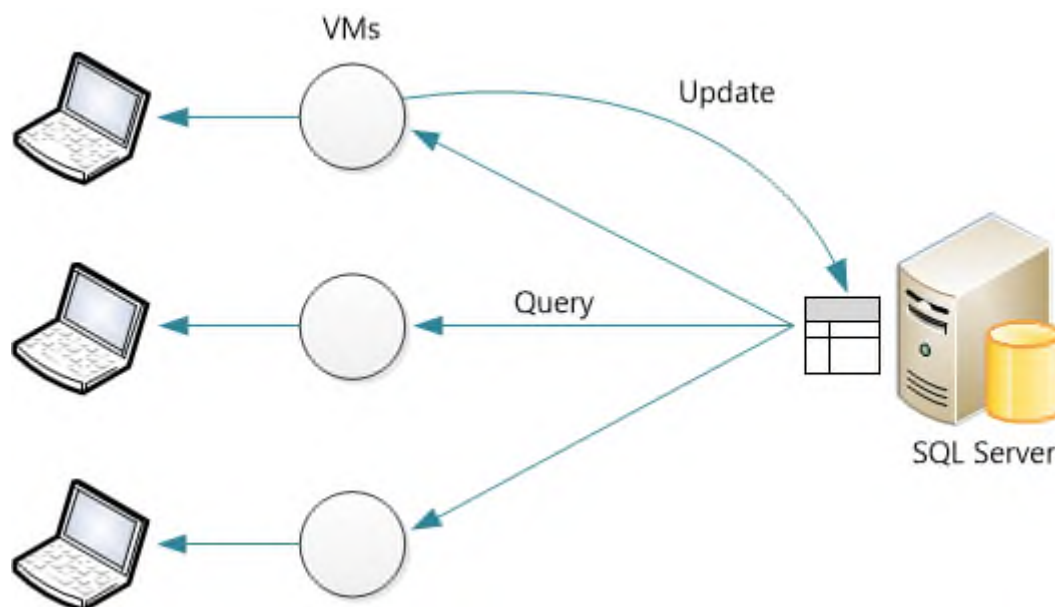
#### Примечание

Существует несколько сценариев, в которых Объединительная плата может стать узким местом. Ниже приведены некоторые типичные сценарии SignalR.

- Широковещательная рассылка сервера (например, биржевая тикер). в этом сценарии хорошо работает Объединительная плата, поскольку сервер управляет скоростью отправки сообщений.

- Клиент-клиент (например, чат). в этом сценарии Объединительная плата может стать узким местом, если количество сообщений масштабируется с учетом количества клиентов; то есть, если скорость передачи сообщений пропорционально увеличению числа клиентов, присоединяемых к нему.
- Высокая частота (например, игры в реальном времени). в этом сценарии не рекомендуется использовать объединительную плату.

В этом упражнении вы будете использовать SQL Server для распространения сообщений в приложении для головоломки . Эти задачи будут выполняться на одном тестовом компьютере, чтобы узнать, как настроить конфигурацию, но для получения полного результата потребуется развернуть приложение SignalR на двух или более серверах. Необходимо также установить SQL Server на одном из серверов или на отдельном выделенном сервере.



#### Задача 1. Основные сведения о сценарии

В этой задаче будут запущены 2 экземпляра класса " головоломка ", моделирующие несколько экземпляров IIS на локальном компьютере. В этом сценарии при ответе на вопросы Trivia в одном приложении обновление не будет уведомлено на странице статистики второго экземпляра. Такое моделирование напоминает среду, в которой приложение разворачивается на нескольких экземплярах и использует подсистему балансировки нагрузки для взаимодействия с ними.

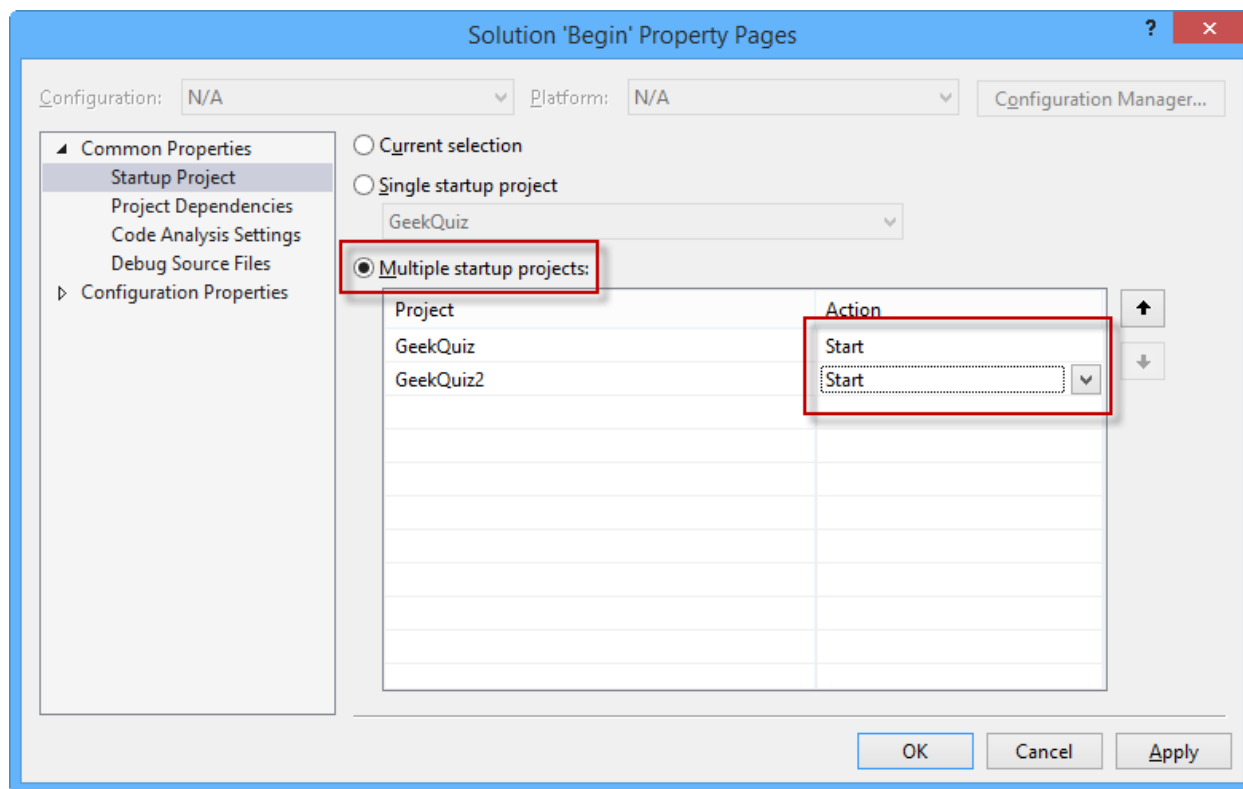
1. Откройте решение `Begin. sln` , расположенное в папке `Source/EX2-скалингаутвиссклсервер/Begin` . После загрузки вы обратите внимание на Обзоратель сервера , что решение имеет два проекта с идентичными структурами, но разными именами.

Это позволит имитировать выполнение двух экземпляров одного приложения на локальном компьютере.

![Начало решения для имитации 2 экземпляров "высоко" головоломка](real-time-web-applications-with-signalr/\_static/image16.png "Начало решения для имитации 2 экземпляров "высоко" головоломка")

Начало решения для имитации 2 экземпляров "высоко" головоломка

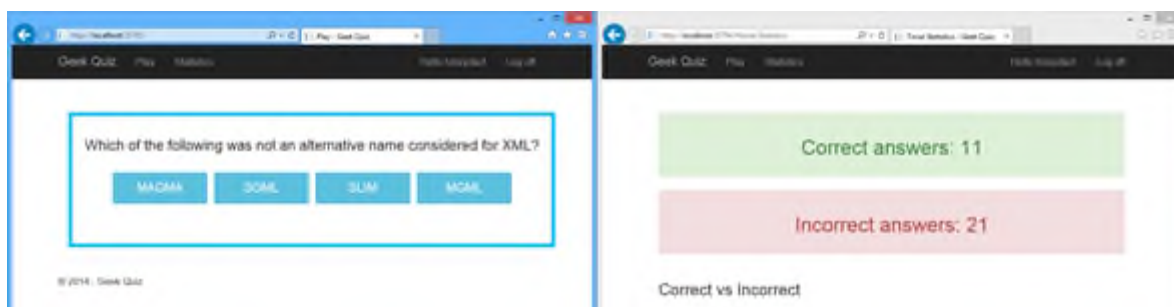
2. Откройте страницу свойств решения, щелкнув правой кнопкой мыши узел решения и выбрав пункт Свойства. В разделе запускаемый проект выберите Несколько запускаемых проектов и измените значение параметра действие для обоих проектов на Запуск.



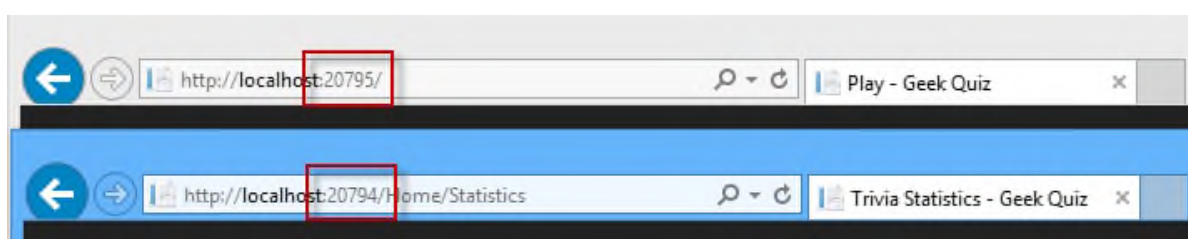
Запуск нескольких проектов

3. Чтобы запустить решение, нажмите клавишу F5. Приложение запустит два экземпляра класса "высокообъектный тест " на разных портах, имитируя несколько экземпляров одного приложения. Закрепите один из браузеров слева, а другой — справа от экрана. Выполните вход с использованием учетных данных или Зарегистрируйте нового

пользователя. После входа в систему откройте страницу Trivia слева и перейдите на страницу Статистика в браузере справа.



Параллельный контрольный опрос



Головоломка в разных портах

4. Начните отвечать на вопросы в браузере слева, и вы увидите, что страница статистики в правильном браузере не обновляется. Это обусловлено тем, что SignalR использует локальный кэш для распределения сообщений между клиентами, и этот сценарий имитирует несколько экземпляров, поэтому кэш не используется совместно. Можно проверить работоспособность SignalR, проверив те же действия, но используя одно приложение. В следующих задачах будет настроена Объединительная плата для репликации сообщений между экземплярами.

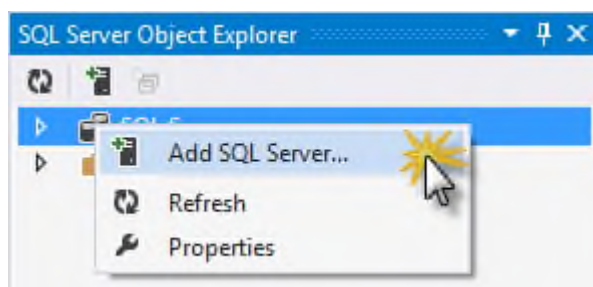
5. Вернитесь в Visual Studio и завершите отладку.

## Задача 2. Создание объединительной платы SQL Server

В этой задаче вы создадите базу данных, которая будет использоваться в качестве объединительной платы для приложения для головоломки. Для просмотра сервера и инициализации базы данных будет использоваться Обзорщик объектов SQL Server. Кроме того, будет включен Service Broker.

1. В Visual Studio откройте представление меню и выберите Обзорщик объектов SQL Server.

2. Подключитесь к экземпляру LocalDB, щелкнув правой кнопкой мыши узел SQL Server и выбрав пункт Добавить SQL Server... .



Добавление SQL Server экземпляра в обозреватель объектов SQL Server

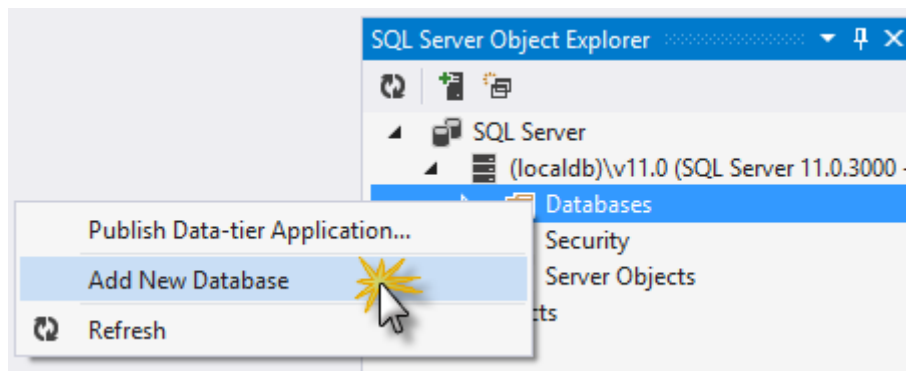
3. Задайте имя сервера (LocalDB) \V11.0 и оставьте проверку подлинности Windows в качестве режима проверки подлинности. Чтобы продолжить, щелкните Подключить .



Подключение к LocalDB

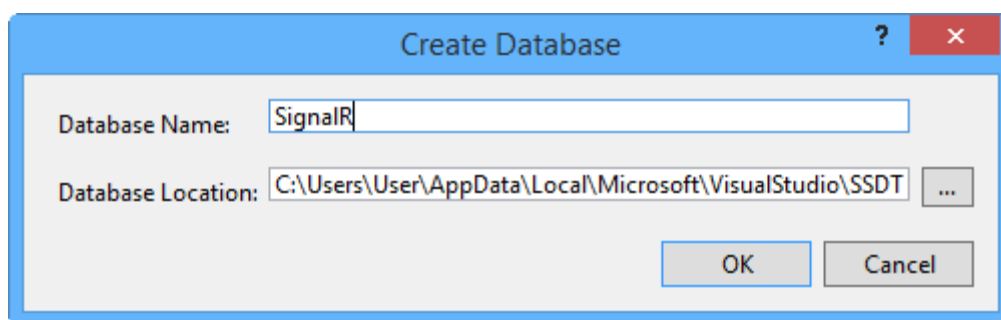
4. Теперь, когда вы подключены к экземпляру LocalDB, необходимо создать базу данных, которая будет представлять SQL Serverную объединительную платформу для SignalR. Для этого щелкните правой кнопкой мыши узел базы данных и выберите команду Добавить новую базу данных.





Добавление новой базы данных

5.        Задайте для базы данных имя SignalR и нажмите кнопку ОК , чтобы создать его.

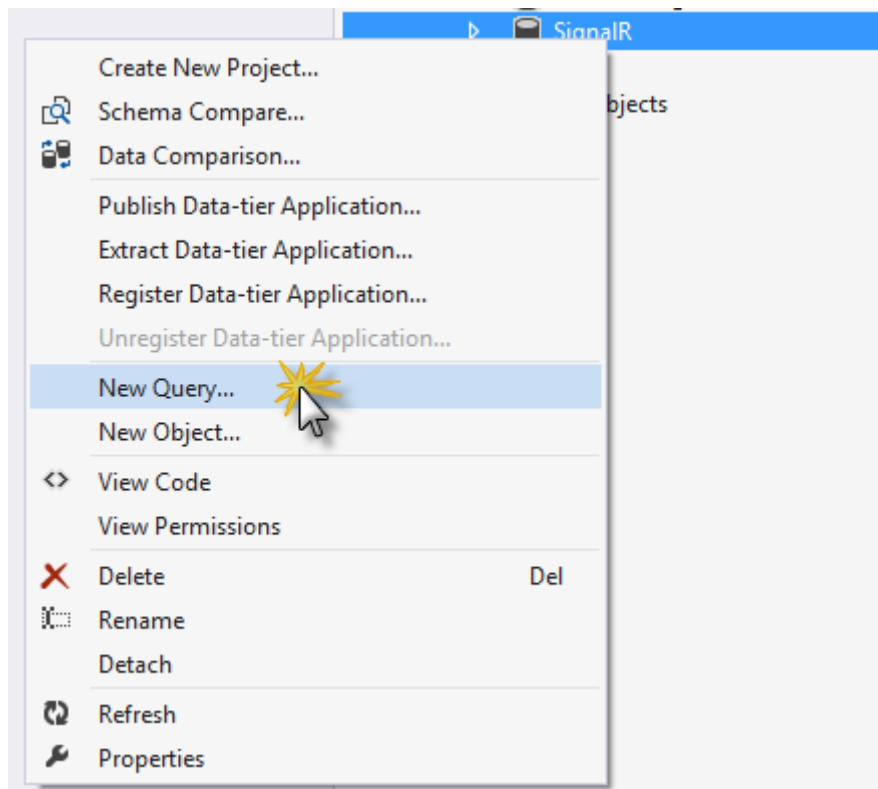


Создание базы данных SignalR

Примечание

Можно выбрать любое имя для базы данных.

6.        Для более эффективного получения обновлений от объединительной платы рекомендуется включить Service Broker для базы данных. Service Broker обеспечивает собственную поддержку обмена сообщениями и очередей в SQL Server. Задняя панель также работает без Service Broker. Откройте новый запрос, щелкнув правой кнопкой мыши базу данных и выбрав пункт создать запрос.

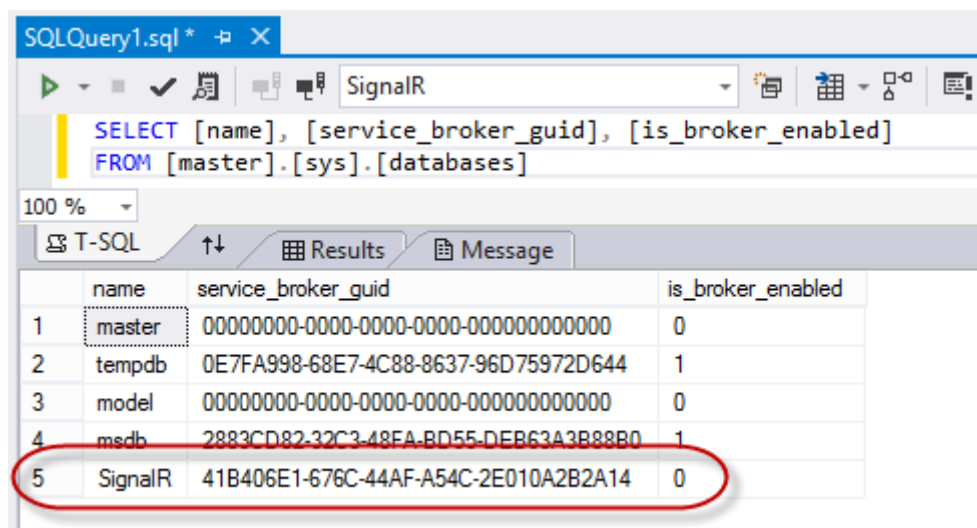


Открытие нового запроса

7. Чтобы проверить, включена ли Service Broker, запросите столбец `_Broker_` включен в представлении каталога `sys. databases`. Выполните следующий скрипт в недавно открытом окне запроса.

SQL

```
SELECT [name], [service_broker_guid], [is_broker_enabled] FROM
[master].[sys].[databases]
```

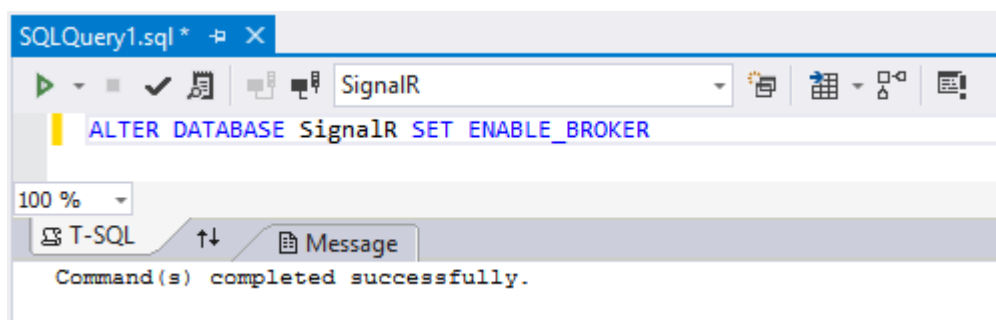


## Запрос состояния Service Broker

□ Если значение столбца `_broker_Enabled` в базе данных равно "0", используйте следующую команду, чтобы включить ее. Замените <базы данных именем, заданным при создании базы данных (например, SignalR).

## SQL

8. `ALTER DATABASE <YOUR-DATABASE> SET ENABLE_BROKER`



9.

## Включение Service Broker

### Примечание

Если этот запрос выглядит как взаимоблокировка, убедитесь, что к базе данных не подключено ни одного приложения.

## Задача 3. Настройка приложения SignalR

В этой задаче вы настроите свой высокодоступный тест для подключения к объединительной плате SQL Server. Сначала необходимо добавить пакет NuGet SignalR. SqlServer и задать строку подключения для базы данных объединительной платы.

Откройте консоль диспетчера пакетов из меню Инструменты > Диспетчер пакетов NuGet. Убедитесь, что в раскрывающемся списке проект по умолчанию выбран проект жееккуиз . Введите следующую команду, чтобы установить пакет NuGet Microsoft. AspNet. SignalR. SqlServer .

### PowerShell

```
Install-Package Microsoft.AspNet.SignalR.SqlServer
```

Повторите предыдущий шаг, но на этот раз для проекта GeekQuiz2.

Чтобы настроить SQL Serverную объединительную плату, откройте файл Startup.CS проекта жееккуиз и добавьте следующий код в метод Configure . Замените <базы данных именем базы данных, которое использовалось при создании SQL Server объединительной платы. Повторите этот шаг для проекта GeekQuiz2 .

(Фрагмент кода — реалтисигналр-EX2-стартапконфигурацион)

```
C#  
  
public class Startup  
{  
    public void Configuration(IAppBuilder app)  
    {  
        var sqlConnectionString =  
@"Server=(localdb)\v11.0;Database=<YOUR-DATABASE>;Integrated Security=True;" ;  
  
GlobalHost.DependencyResolver.UseSqlServer(sqlConnectionStri  
ng) ;  
  
        this.ConfigureAuth(app) ;  
        app.MapSignalR() ;  
    }  
}
```

Теперь, когда оба проекта настроены для использования объединительной платы SQL Server, нажмите клавишу F5 , чтобы запустить их одновременно.

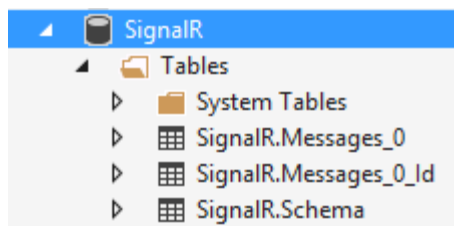
Опять же, Visual Studio запустит два экземпляра класса « визуальный опрос » на разных портах. Закрепите один из браузеров слева, а другой — справа от экрана и войдите с помощью своих учетных данных. Не задерживайте страницу Trivia слева и перейдите к странице статистики в правильном браузере.

Начните отвечать на вопросы в браузере слева. На этот раз страница статистики обновляется благодаря объединительной плате. Переключайтесь между приложениями (Статистика отображается слева, а Trivia — справа) и повторите проверку, чтобы убедиться, что она работает для обоих экземпляров. Задняя панель выступает в качестве общего кэша сообщений для каждого

подключенного сервера, и каждый сервер хранит сообщения в собственном локальном кэше для распространения подключенным клиентам.

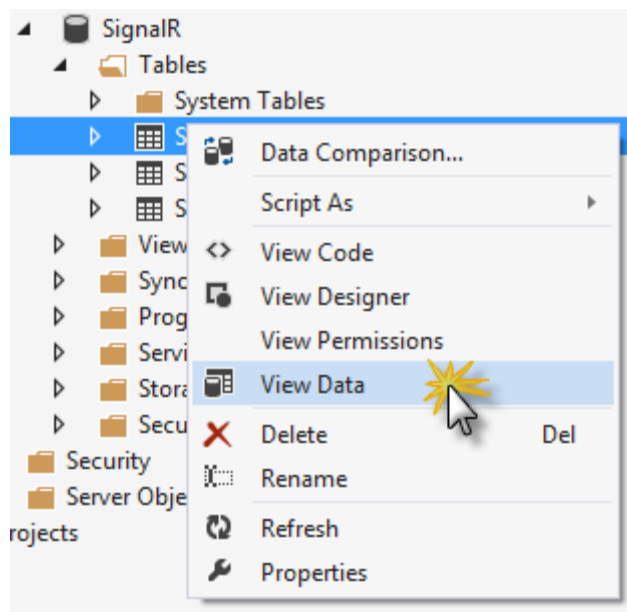
### 3. Вернитесь в Visual Studio и завершите отладку.

Компонент объединительной платы SQL Server автоматически создает необходимые таблицы в указанной базе данных. На панели Обозреватель объектов SQL Server откройте базу данных, созданную для объединительной платы (например, SignalR), и разверните ее таблицы. Должны отобразиться следующие таблицы:



Таблицы, созданные в объединительной плате

Щелкните правой кнопкой мыши элемент SignalR. messages\_0 Table и выберите Просмотреть данные.



Просмотр таблицы сообщений о объединительной плате SignalR

При ответе на вопросы Trivia можно увидеть различные сообщения, отправляемые в центр . Объединительная плата распространяет эти сообщения на любой подключенный экземпляр.

SignalR.Messages_0 [Data] ▢ ✕			
<div> <span>■</span> <span>🔍</span> <span>🔊</span> Max Rows: 1000 <span>🔄</span> <span>📄</span> </div>			
	PayloadId	Payload	InsertedOn
▶	1	<Binary data>	25/02/2014 05:3...
	2	<Binary data>	25/02/2014 05:3...
	3	<Binary data>	25/02/2014 05:3...
	4	<Binary data>	25/02/2014 05:3...
	5	<Binary data>	25/02/2014 05:3...
	6	<Binary data>	25/02/2014 05:3...
	7	<Binary data>	25/02/2014 05:3...
	8	<Binary data>	25/02/2014 05:3...
	9	<Binary data>	25/02/2014 05:3...
	10	<Binary data>	25/02/2014 05:3...
	11	<Binary data>	25/02/2014 05:3...
	12	<Binary data>	25/02/2014 05:3...
	13	<Binary data>	25/02/2014 05:3...
	14	<Binary data>	25/02/2014 05:3...
	15	<Binary data>	25/02/2014 05:3...
*	NULL	NULL	NULL

Таблица сообщений о объединительной плате

### Контрольные вопросы.

1. Распределенные и централизованные базы данных. Архитектура файл-сервер. Архитектура клиент-сервер.
2. Иерархическая и сетевая модели данных.
3. Реляционная модель данных. История развития. Основные понятия (тип данных, домен, отношение, кортеж, атрибут, ключ).
4. Реляционная база данных.
5. Функции системы управления базами данных (СУБД): управления данными во внешней памяти, управление буферами оперативной памяти, управление транзакциями.
6. Функции системы управления базами данных: журнализация, поддержка языков баз данных.
7. Типовая организация современной СУБД.
8. Базовые средства манипулирования реляционными данными.
9. Реляционная алгебра. Общая интерпретация реляционных операций.
10. Особенности теоретико-множественных операций реляционной алгебры.
11. Реляционное исчисление.

12. Схема отношения. Функциональные зависимости. Декомпозиция отношений, транзитивные зависимости.
13. Нормализация отношений. Проектирование с использованием метода сущность - связь.
14. 1, 2, 3 и 4 нормальные формы. Нормальная форма Бойса-Кодда. Приведение базы данных к нормализованному виду.
15. Целостность сущностей и ссылок.
16. История развития SQL. Функции и основные возможности SQL. ANSI SQL; T-SQL; PL/SQL; Jet SQL
17. Выражения в SQL.
18. СУБД в архитектуре клиент-сервер. Открытые системы.
19. Клиенты и серверы локальных сетей.
20. Системная архитектура клиент-сервер. Удаленный вызов процедур.
21. Сервера баз данных.
22. Типичное распределение функций между клиентом и сервером. Распределенные базы данных.
23. Создание и модификация базы данных в MS SQL Server.
24. Сортировка и поиск данных в MS SQL Server.
25. Язык T-SQL. Числовые и денежные типы данных. Типы данных для хранения информации о времени.
26. Язык T-SQL. Символьные и текстовые типы данных.
27. Язык T-SQL. Специальные типы данных. Конвертирование типов данных.
28. Процесс проектирования таблиц в реляционной базе данных. Определение идентификационной колонки.
29. Создание таблиц средствами T-SQL.
30. Изменение структуры таблицы средствами T-SQL. Удаление таблиц.
31. Добавление данных в таблицу средствами T-SQL. Использование INSERT и SELECT...INTO.
32. Извлечение данных средствами T-SQL. Команда SELECT. Разделы SELECT и INTO.
33. Извлечение данных средствами T-SQL. Команда SELECT. Раздел FROM.
34. Извлечение данных средствами T-SQL. Команда SELECT. Разделы WHERE, GROUP BY, HAVING, ORDER BY.
35. Изменение данных в таблице средствами T-SQL. Команда UPDATE.
36. Удаление данных средствами T-SQL. Команда DELETE.
37. Transact-SQL ODBC и MS SQL Server.
38. Использование представлений.

39. Хранимые процедуры. Этапы создания.
40. Создание, модификация и удаление хранимых процедур средствами T-SQL.
41. Создание, изменение и удаление представлений средствами T-SQL.
42. Современные промышленно-сопровождаемые СУБД
43. Системы управления базами данных следующего поколения.

### **Тестовое задание.**

1 Предметная область - это:

- 1) совокупность таблиц, состоящих из записей и полей; информации об индексах и связях; хранимых процедур;
- 2) совокупности таблиц, объединенных связями; экранных форм, отчетов, запросов
- 3) некоторая часть реально существующей системы, функционирующая как самостоятельная единица;
- 4) поименованная совокупность структурированных данных, относящихся к определенной предметной области;
- 5) набор правил, обеспечивающих соответствие ключевых значений в связанных таблицах.

2 Система управления базой данных (СУБД) - это:

- 1) регулярная структура, состоящая из однотипных записей, разбитых на поля;
- 2) комплекс программных и языковых средств, необходимых для создания и модификации базы данных;
- 3) поименованная совокупность структурированных данных, относящихся к определенной предметной области;
- 4) служебная информация, содержащая упорядоченные сведения о ключевых значениях;
- 5) программно-аппаратный комплекс, предназначенный для хранения и обработки информации какой-либо предметной области.

3. База данных - это:

- 1) комплекс программных и языковых средств, необходимых для добавления, модификации, удаления, поиска и отбора информации;
- 2) совокупности таблиц, объединенных связями; экранных форм, отчетов, запросов;
- 3) некоторая часть реально существующей системы, функционирующая как самостоятельная единица;
- 4) поименованная совокупность структурированных данных, относящихся к определенной предметной области;
- 5) программно-аппаратный комплекс, предназначенный для хранения и обработки информации какой-либо предметной области.

4 Реляционная модель базы - это:



- 1) совокупность таблиц, состоящих из записей и полей; информации об индексах и связях; хранимых процедур;
- 2) совокупности таблиц, объединенных связями; экранных форм, отчетов, запросов;
- 3) некоторая часть реально существующей системы, функционирующая как самостоятельная единица;
- 4) поименованная совокупность структурированных данных, относящихся к определенной предметной области;
- 5) набор правил программно-аппаратный комплекс, предназначенный для хранения и обработки информации какой-либо предметной области.

5 Таблица базы данных - это:

- 1) регулярная структура, состоящая из однотипных записей, разбитых на поля;
- 2) комплекс программных и языковых средств, необходимых для создания и модификации базы данных;
- 3) поименованная совокупность структурированных данных, относящихся к определенной предметной области;
- 4) служебная информация, содержащая упорядоченные сведения о ключевых значениях;
- 5) функциональная зависимость между объектами.

6. Ключ таблицы базы данных - это:

- 1) поле или строковое выражение, образованное из значений нескольких полей, по которому можно определить значения других полей для одной или нескольких записей таблицы;
- 2) поле или строковое выражение, образованное из значений нескольких полей, по которому можно однозначно идентифицировать строку в таблице;
- 3) программный модуль, сохраняемый в базе данных для выполнения определенных операций с информацией базы;
- 4) поименованная совокупность структурированных данных, относящихся к определенной предметной области;
- 5) набор правил, обеспечивающих связи между таблицами в базе данных.

7. Отношение в теории реляционных баз данных - это:

- 1) основной объект базы данных, состоящий из кортежей и имеющий определенный набор свойств – атрибутов;
- 2) набор всех допустимых значений, которые может содержать атрибут;
- 3) формальный метод анализа отношений на основе их первичного ключа и существующих функциональных зависимостей;
- 4) функциональная зависимость между объектами;
- 5) математические принципы, вытекающие из теории множеств и логики предикатов/

8. Связи между ключевыми значениями в реляционной модели бывают:

- 1) "один к одному", "один ко многим", "многие ко многим";
- 2) только "один к одному";
- 3) только "один ко многим";
- 4) только "многие ко многим".

9. Сетевая модель данных состоит из:

- 1) набора экземпляров одного типа, образующих дерево с одним корневым объектом;
- 2) набора записей и набора связей с любым числом других записей;
- 3) совокупности таблиц со связями по ключевым значениям;
- 4) многомерных таблиц, созданных с использованием объектно-ориентированных методов;
- 5) множества баз данных, управляемых одной СУБД.

10. Реляционная модель данных состоит из:

- 1) набора экземпляров одного типа, образующих дерево с одним корневым объектом;
- 2) набора записей и набора связей с любым числом других записей;
- 3) совокупности таблиц со связями по ключевым значениям;
- 4) многомерных таблиц, созданных с использованием объектно-ориентированных методов;
- 5) множества баз данных, управляемых одной СУБД.

11. Иерархическая модель данных состоит из:

- 1) набора экземпляров одного типа, образующих дерево с одним корневым объектом;
- 2) набора записей и набора связей с любым числом других записей;
- 3) совокупности таблиц со связями по ключевым значениям;
- 4) многомерных таблиц, созданных с использованием объектно-ориентированных методов;
- 5) множества баз данных, управляемых одной СУБД.

12. Использование каких моделей данных наиболее эффективно в системах класса OLAP:

- 1) Реляционных;
- 2) Объектно-ориентированных;
- 3) Многомерных;
- 4) Сетевых.

13. Предметно-ориентированный, интегрированный, неизменяемый и поддерживающий хронологию набор данных, предназначенный для обеспечения принятия управленческих решений, называется:

- 1) Банком данных;
- 2) Информационным массивом;
- 3) Хранилищем данных;
- 4) Информационной системой.

14. Какое из перечисленных высказываний не является верным по отношению к объектно-ориентированным базам данных (ООБД):
- 1) При ссылке на объекты необходимо повторять пользовательские ключи;
  - 2) Все объекты ООБД идентифицируются одинаковым образом;
  - 3) Идентификаторы никогда не изменяются до тех пор, пока существуют объекты, которые они идентифицируют;
  - 4) Идентификаторы не характеризуются излишней сложностью.
15. В случае, если СУБД по отношению к базе данных выполняет не только те действия, которые явно указывает пользователь, но и дополнительные действия в соответствии с правилами, заложенными в саму СУБД, база данных называется:
- 1) Многомерной;
  - 2) Активной;
  - 3) Реляционной;
  - 4) Дедуктивной.
16. Оператор SQL, выполняющий проверку на диапазон значений:
- 1) FROM...TO;
  - 2) BETWEEN...AND;
  - 3) FROM...AND;
  - 4) BETWEEN...TO.
17. Оператор IN в языке SQL выполняет:
- 1) Проверку выражения на NULL;
  - 2) Проверку выражения на совпадение с любым из элементов списка;
  - 3) Проверку выражения на совпадение со всеми элементами списка;
  - 4) Логическую импликацию выражений.
18. Какая команда SQL осуществляет выбор пяти первых фамилий студентов, упорядоченных по учебным группам:
- 1) SELECT Имя, Фамилия FROM Студент[Группа] LIMIT 5;
  - 2) SELECT Имя, Фамилия ORDER BY [Группа] FROM Студент LIMIT 5;
  - 3) SELECT Имя, Фамилия FROM Студент ORDER BY[Группа] LIMIT 5;
  - 4) SELECT Имя, Фамилия ORDER BY[Группа] WHERE Студент LIMIT 5.
19. Чем отличаются подходы, применяемые в реляционной алгебре и реляционном исчислении?
- 1) Реляционная алгебра использует описательный подход, а реляционное исчисление предписывающий;
  - 2) Оба подхода описательные;
  - 3) Оба подхода предписывающие;

- 4) Реляционная алгебра использует предписывающий подход, а реляционное исчисление описательный.

20. Какой из перечисленных тестов для СУБД измеряет насколько быстро СУБД может выполнять однотабличный запрос, ответ на который содержит определенную процентную долю строк таблицы:

- 1) Полное сканирование;
- 2) Обновление;
- 3) Чтение с произвольной выборкой;
- 4) Выборка.

## ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

<b>№ п.п.</b>	<b>Содержание изменения</b>	<b>Дата, номер протокола заседания кафедры, подпись зав.кафедрой</b>
1	2	3
1		
2		
3		
4		