ГОУВПО

ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ

к индивидуальной работе по дисциплине

«Организация баз данных и знаний»

(для студентов направлений подготовки 09.03.02 "Информационные системы и технологии" (специальностей "Информационные технологии в медиаиндустрии и дизайне", "Системы автоматизированного проектирования"), 02.03.01 "Математика и компьютерные науки" (специальность "Компьютерное моделирование и дизайн")

Донецк-ДонНТУ-2016

ГОУВПО

ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ

к индивидуальной работе по дисциплине

«Организация баз данных и знаний»

(для студентов направлений подготовки 09.03.02 "Информационные системы и технологии" (специальностей "Информационные технологии в медиаиндустрии и дизайне", "Системы автоматизированного проектирования"), 02.03.01 "Математика и компьютерные науки" (специальность "Компьютерное моделирование и дизайн")

Рассмотрено на заседании кафедры
программной инженерии
Протокол № 1 от 30.08.2016
Утверждено на заседании
учебно-издательского Совета ДонНТУ
протокол № от

Донецк –2016

УДК 681.3

Методические указания и задания к индивидуальной работе по дисциплине «Организация баз данных и знаний» для студентов направления подготовки 09.03.02 "Информационные системы и технологии" (специальностей "Информационные технологии в медиаиндустрии и дизайне", "Системы автоматизированного проектирования"), 02.03.01 "Математика и компьютерные науки" (специальность "Компьютерное моделирование и дизайн"), Чернышова А.В., Донецк, ДонНТУ, 2016 - стр.

Приведены методические указания и задания к выполнению индивидуальной работы по дисциплине «Организация баз данных и знаний». Излагаются вопросы, связанные с организацией работы серверной СУБД MySQL. Предлагается к изучению и практической реализации вопросы, связанные с созданием базы данных, созданием таблиц, реализацией простых SQL запросов в СУБД MySQL.

Методические указания предназначены для усвоения теоретических основ и формирования практических навыков по курсу «Организация баз данных и знаний».

Составители: ст.преп. каф. ПИ Чернышова А.В.

Тема: Создание базы данных, таблиц, запросов с использованием серверной СУБД MySQL. Анализ возможностей используемой СУБД.

Цель работы: Познакомиться с основами работы с серверной СУБД MySQL. Проанализировать возможности СУБД MySQL.

Методические указания к индивидуальной работе

Система Управления Базой Данных (СУБД) - комплекс языков и программ, позволяющий создавать БД и управлять ее функционированием. СУБД обрабатывает обращения к базе данных, поступающие от пользователей, прикладных процессов и выдает необходимые им сведения.

СУБД характеризуется используемой моделью и средствами администрирования, разработки прикладных процессов, работы в информационной сети.

Типовая организация СУБД:

- ядро, которое отвечает за управление данными во внешней и оперативной памяти, управление транзакциями и журнализацию. При использовании архитектуры "клиент-сервер" ядро является основной составляющей серверной части системы;

- компилятор языка SQL;

- подсистема поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД;

- сервисные программы (внешние утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

Классификация СУБД

1.По модели данных

- иерархические;

- сетевые;
- реляционные;
- объектно-ориентированные;
- объектно-реляционные;
- пост-реляционные;

2. По степени распределенности:

- локальные СУБД;
- распределенные СУБД;

3. По способу доступа к БД:

- файл-серверные;
- клиент-серверные;
- встраиваемые.

Иерархическая БД

Иерархическая модель базы данных состоит из объектов с указателями от родительских объектов к потомкам, соединяя вместе связанную информацию.

Иерархические базы данных могут быть представлены как дерево, состоящее из объектов различных уровней.

Сетевая БД

К основным понятиям сетевой модели базы данных относятся: уровень, элемент (узел), связь.

Реляционная БД - база данных, логически организованная в виде набора отношений ее компонентов.

Характерной особенностью РБД является структура, выполненная в виде таблиц. Строки таких таблиц соответствуют записям, столбцы - атрибутам (признакам хранимых данных).

Объектно-ориентированные базы данных (ООБД) появились совсем недавно как естественное развитие объектно-ориентированных языков программирования.

Данные представлены в виде объектов различных классов.

Как правило, имеются возможности создавать новые классы, наследовать их от уже имеющихся, задавать произвольные атрибуты и методы для классов.

Для доступа к объектам, в каждой ООБД обычно предусматривается свой собственный язык, либо расширение другого языка. Пока еще ООБД недостаточно развиты и не представляют серьезной конкуренции SQL-серверам. Примеры: Cache, FastObjects,GemStone/S, Jasmine, ObjectStore и др

Объектно-реляционные БД

Разработчики многих реляционных БД включают в свои базы средства работы с объектными типами данных.

Такие базы данных получили название объектно-реляционных.

По этому пути, в частности, развивается и Oracle. Бывшая ранее чисто реляционной базой, Oracle начиная с 8 версии поддерживает возможность хранения и обработки объектов и безо всякой натяжки может быть отнесена к объектно-реляционному классу баз данных.

Пост-реляционными, часто называют многомерные базы данных.

Данные в многомерных базах, представляются в виде разреженных многомерных массивов, а не плоских таблиц, как в реляционных базах. Для определенных задач, многомерные базы могут давать значительный выигрыш в быстродействии, по сравнению с реляционными.

Примеры, Cache, Teradata

Файл-серверные СУБД

В файл-серверных СУБД файлы данных располагаются централизованно на файл-сервере. СУБД располагается на каждом клиентском компьютере (рабочей станции). Доступ СУБД к данным осуществляется через локальную сеть.

Примеры: Microsoft Access, Paradox, dBase.

Клиент-серверные СУБД

Клиент-серверная СУБД располагается на сервере вместе с БД и осуществляет доступ к БД непосредственно, в монопольном режиме. Все

клиентские запросы на обработку данных обрабатываются клиент-серверной СУБД централизованно.

Примеры: Oracle, Firebird, Interbase, IBM DB2, MS SQL Server, Sybase, PostgreSQL, MySQL.

Встраиваемая СУБД — библиотека, которая позволяет унифицированным образом хранить большие объёмы данных на локальной машине. Доступ к данным может происходить через SQL либо через особые функции СУБД. Встраиваемые СУБД быстрее обычных клиент-серверных и не требуют установки сервера, поэтому востребованы в локальном ПО, которое имеет дело с большими объёмами данных (например, геоинформационные системы).

Примеры: OpenEdge, SQLite, BerkeleyDB, один из вариантов Firebird, MySQL, Sav Zigzag, Microsoft SQL Server Compact.

Транзакция - это последовательность операций над БД, рассматриваемых СУБД как единое целое. Либо транзакция успешно выполняется, и СУБД фиксирует (COMMIT) изменения БД, произведенные этой транзакцией, во внешней памяти, либо ни одно из этих изменений никак не отражается на состоянии БД. Понятие транзакции необходимо для поддержания логической целостности БД. Поддержание механизма транзакций является обязательным условием даже однопользовательских СУБД (если, конечно, такая система заслуживает названия СУБД). Но понятие транзакции гораздо более важно в многопользовательских СУБД.

То свойство, что каждая транзакция начинается при целостном состоянии БД и оставляет это состояние целостным после своего завершения, делает очень удобным использование понятия транзакции как единицы активности пользователя по отношению к БД. При соответствующем управлении параллельно выполняющимися транзакциями со стороны СУБД каждый из пользователей может в принципе ощущать себя единственным пользователем СУБД (на самом деле, это несколько идеализированное

представление, поскольку в некоторых случаях пользователи многопользовательских СУБД могут ощутить присутствие своих коллег).

С управлением транзакциями в многопользовательской СУБД связаны важные понятия **сериализации транзакций и сериального плана выполнения смеси транзакций**

Под **сериализаций** параллельно выполняющихся транзакций понимается такой **порядок планирования их работы**, при котором суммарный эффект смеси транзакций эквивалентен эффекту их некоторого последовательного выполнения. **Сериальный план** выполнения смеси транзакций - это такой план, который приводит к сериализации транзакций. Понятно, что если удается добиться действительно сериального выполнения смеси транзакций, то для каждого пользователя, по инициативе которого образована транзакция, присутствие других транзакций будет незаметно (если не считать некоторого замедления работы по сравнению с однопользовательским режимом).

Существует несколько базовых алгоритмов сериализации транзакций. В централизованных СУБД наиболее распространены алгоритмы, основанные на синхронизационных захватах объектов БД. При использовании любого алгоритма сериализации возможны ситуации конфликтов между двумя или более транзакциями по доступу к объектам БД. В этом случае для поддержания сериализации необходимо выполнить откат (ликвидировать все изменения, произведенные в БД) одной или более транзакций. Это один из случаев, когда пользователь многопользовательской СУБД может реально (и достаточно неприятно) ощутить присутствие в системе транзакций других пользователей.

Журнализация

СУБД должна обеспечивать надежность хранения данных, т.е. быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя. Обычно рассматриваются два возможных вида аппаратных сбоев: так называемые мягкие сбои, которые

можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания), и жесткие сбои, характеризуемые потерей информации на носителях внешней памяти. Примерами программных сбоев могут быть: аварийное завершение работы СУБД (по причине ошибки в программе или в результате некоторого аппаратного сбоя) или аварийное завершение пользовательской программы, в результате чего некоторая транзакция остается незавершенной. Первую ситуацию можно рассматривать как особый вид мягкого аппаратного сбоя; при возникновении последней требуется ликвидировать последствия только одной транзакции.

Понятно, что в любом случае для восстановления БД нужно располагать некоторой дополнительной информацией. Другими словами, поддержание надежности хранения данных в БД требует избыточности хранения данных, причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение журнала изменений БД.

Журнал- это особая **часть БД**, недоступная пользователям СУБД и поддерживаемая с особой тщательностью (иногда поддерживаются две копии журнала, располагаемые на разных физических дисках), в которую поступают записи обо всех **изменениях**основной части БД.

В разных СУБД изменения БД журнализуются на разных уровнях: иногда запись в журнале соответствует некоторой логической операции изменения БД (например, операции удаления строки из таблицы реляционной БД), иногда - минимальной внутренней операции модификации страницы внешней памяти; в некоторых системах одновременно используются оба подхода.

Во всех случаях придерживаются стратегии "упреждающей" записи в журнал (так называемого протокола Write Ahead Log - WAL). Грубо говоря, эта стратегия заключается в том, что запись об изменении любого объекта БД должна попасть во внешнюю память журнала раньше, чем измененный

объект попадет во внешнюю память основной части БД. Известно, что если в СУБД корректно соблюдается протокол WAL, то с помощью журнала можно решить все проблемы восстановления БД после любого сбоя.

Для восстановления БД после **жесткого сбоя** используют **журнали архивную копию БД**. Архивная копия - это полная копия БД к моменту начала заполнения журнала.

Поддержка языков БД

Для работы с базами данных используются специальные языки, в целом называемые **языками баз данных**. В ранних СУБД поддерживалось несколько специализированных по своим функциям языков. Чаще всего выделялись два языка - **язык определения схемы БД (SDL - Schema Definition Language) и язык манипулирования данными (DML - Data Manipulation Language).** SDL служил главным образом для определения логической структуры БД, т.е. той структуры БД, какой она представляется пользователям. DML содержал набор операторов манипулирования данными, т.е. операторов, позволяющих заносить данные в БД, удалять, модифицировать или выбирать существующие данные.

В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания, и обеспечивающий базовый пользовательский интерфейс с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (**Structured Query Language**).

Язык SQL включает язык определения данных, язык манипулирования данными, язык запросов, язык управления данными. Более детальную информацию можно получить по ссылке: http://poisk-ru.ru/s54483t1.html

Установка и настройка Open Server

Руководство пользователя

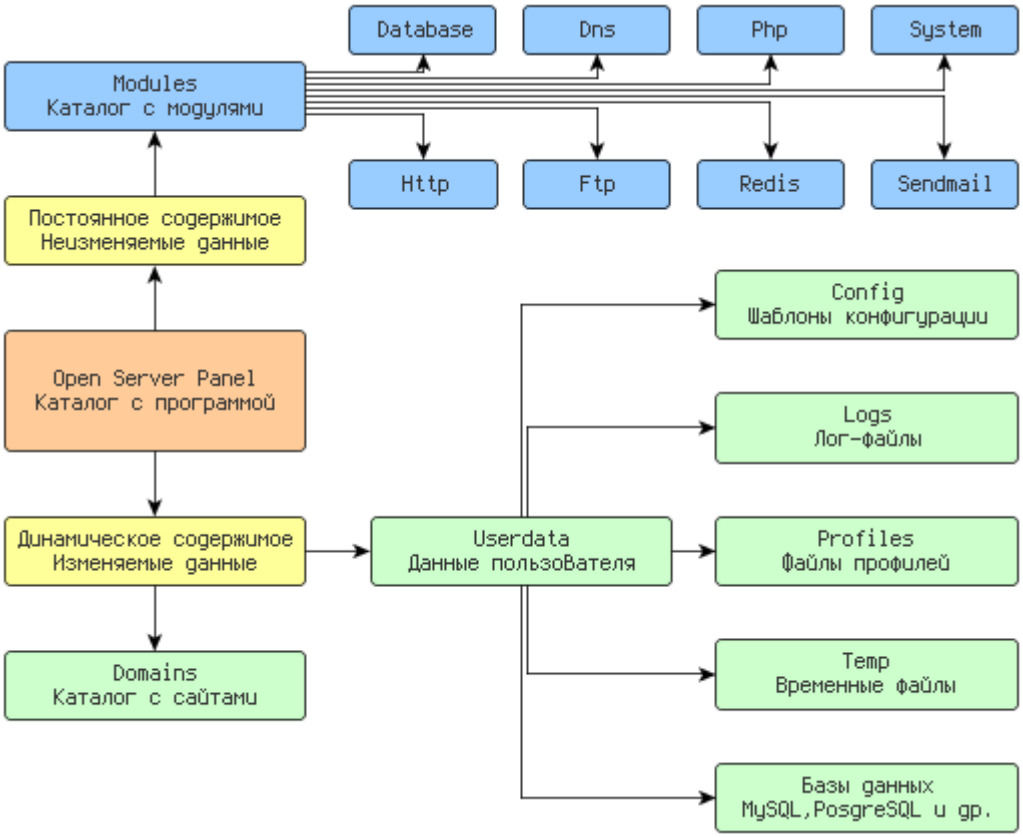

Рисунок 1 – Архитектура Open Server

Архитектура расположения каталогов программного комплекса подразумевает под собой чёткое разделение двух различных типов данных: динамических данных пользователя (настройки, временные файлы, логи т.д.) и статичных данных (модули, программы, служебные файлы).

Если говорить простым языком, то в папке modules никакие файлы никогда не изменяются, не создаются и не удаляются, а в папках domains и userdata напротив, данные постоянно меняются. Такая архитектура создана специально для упрощения синхронизации данных между различными копиями комплекса и экономии места при резервном копировании.

Установка и запуск

Установка

OSPanel является портативным программным комплексом и не требует установки. Сборку можно разместить на внешнем жёстком диске или флэш-

накопителе, это позволит использовать OSPanel на любом компьютере, который отвечает системным требованиям.

Дистрибутив представляет собой самораспаковывающийся архив в формате 7ZIP (расширение .exe). Запустите исполняемый файл дистрибутива и выберите путь для распаковки файлов.

Системные требования

- Необходимый минимум системных ресурсов: 200 Мб RAM и 1 Гб на HDD;
- Windows (32-bit или 64-bit): Windows 8 / Windows 7 / Windows Server 2008 / Windows Vista / Windows XP SP3;
- Установленный набор библиотек Microsoft Visual C++ 2005-2008-2010 Redistributable Package x86;

Скачать можно по ссылке с официального сайта: https://ospanel.io/download/

Запуск

Для запуска OSPanel используйте файл OSPanel.exe. При наличии возможности программу следует запускать только от имени администратора. После старта программы вы увидите красный флажок в трее Windows (область возле системных часов). Чтобы включить непосредственно сам веб-сервер нажмите на флажок, далее выберите пункт меню [Меню → Запустить].

Если сервер не запускается, обратитесь по ссылке https://ospanel.io/docs/#reshenie-problem

Внимание

Перед началом использования OSPanel выполните [Меню → Дополнительно → Первый запуск]. Наличие установленного набора библиотек Microsoft Visual C++ 2005-2008-2010 Redistributable Package x86 является обязательным системным требованием, без их наличия OSPanel работать не будет.

Запуск с CD/DVD

OSPanel умеет запускаться с CD/DVD дисков так же как и с обычного HDD диска или flash-накопителя с той лишь разницей, что при работе с оптического диска никакие изменения и файлы после выхода из программы сохранены не будут.

Как известно CD/DVD диск доступен только в режиме чтения, что делает невозможным привычный запуск и использование сервера на таком носителе. В OSPanel встроен достаточно простой механизм для выхода из этой ситуации: при запуске сервер копируется во временную папку компьютера, а во время выхода из программы эта папка полностью удаляется. Таким образом, вся работа OSPanel происходит на компьютере пользователя, а не на оптическом диске.

Запуск с оптических дисков рекомендуется использовать только при создании демонстрационных сборок и автономных программных пакетов (см. Создание сборок).

Установка обновлений

OSPanel это достаточно сложный программный комплекс с постоянно совершенствующейся архитектурой. Как таковой процедуры обновления не предусмотрено. При выходе новой версии OSPanel необходимо заново выполнить все настройки, скопировать папки ваших сайтов и выполнить перенос баз данных. Не распаковывайте файлы дистрибутива поверх существующей версии, а так же не пытайтесь скопировать файлы конфигурации и профилей из старой версии программы в новую! Поскольку выход новых версий OSPanel иногда может быть довольно частым, то не стоит обновлять вашу сборку каждый раз, лучше пропустите несколько версий.

Если есть возможность обновиться простым копированием файлов, то ссылка на патч всегда публикуется в новостях на сайте вместе с аннотацией к новой версии.

Совместимость

Часть новых модулей OSPanel (PHP 5.5, MongoDB, Apache 2.4) несовместима с устаревшими операционными системами, такими как Windows XP, Windows 2003, Windows Vista. При использовании таких модулей и запуске сервера на устаревших системах вы получите сообщение об ошибке. Так же следует знать, что некоторые модули могут работать только совместно, например это Apache 2.4 и PHP 5.5.

Подключение

Ниже представлены параметры для подключения к модулям установленные в OSPanel по умолчанию. Вы всегда можете самостоятельно изменить эти настройки по своему усмотрению.

Подключение к MySQL

- Адрес: домен вашего сайта*
- Порт: 3306
- Пользователь: mysql
- Пароль: mysql

ROOT подключение к MySQL

- Пользователь: root
- Пароль: (пусто)

Информация

Кодировка, установленная по умолчанию в настройках MySQL сервера, не действует на пользователя ROOT. Кодировку нужно будет явно указывать в скриптах подключения к БД, потому использовать пользователя ROOT не рекомендуется.

Подключение к PostgreSQL

- Адрес: домен вашего сайта*
- Порт: 5432
- Пользователь: postgres
- Пароль: (пусто)

Подключение к MongoDB

- Адрес: домен вашего сайта*
- Порт: 27017
- Пользователь: (пусто)
- Пароль: (пусто)

Подключение к Memcache

- Адрес: домен вашего сайта*
- Порт: 11211

Макс. размер памяти используемой сервером Memcache по умолчанию равен 64 Мб. Данный параметр можно изменить в настройках OSPanel [Меню → Настройки → Разное].

Подключение к FTP

- Адрес: домен вашего сайта*
- Порт: 21 (990 для FTPS)
- Пользователь: ftp
- Пароль: ftp

***Например, если ваш скрипт размещен по адресу test.server.local/mysql.php, то хостом (адресом) MySQL, PostgreSQL, FTP и Memcache сервера будет домен: test.server.local**

Домен localhost

Если вы хотите использовать привычный адрес localhost для подключения к MySQL, PostgreSQL, FTP или Memcache серверу, то достаточно создать стандартный домен или алиас с именем localhost.

Автоматизация подключения

Когда вы работаете с локальными копиями действующих веб-проектов, возникают трудности с постоянным редактированием файлов конфигурации,

в основном это касается настроек подключения к базе данных. Чтобы этого избежать рекомендуется:

1. Локально создать пользователя базы данных с теми же именем, паролем и привилегиями, что используются на удалённом сервере.
2. Создать алиас с тем же именем, что используется в качестве хоста базы данных на удалённом сервере.

Домены и алиасы

Режимы управления доменами

В OSPanel существует три режима управления списком доменов: автопоиск, ручное управление и ручное+автопоиск. По умолчанию используется первый режим автоматического подключения папок из корневой директории указанной в настройках.

Как работает автопоиск

Программа сканирует заданную веб-директорию на наличие папок с доменами, после чего в каждой найденной папке производится поиск подпапок (корневой папки домена) которые указаны в настройках для автосканирования. Если ни одна из предполагаемых корневых подпапок не найдена, то корнем домена становится сама папка с доменом.

Как работает совмещённое управление (ручное + автопоиск)

При использовании совмещенного режима управления доменами программа сначала подключает домены созданные вручную, после чего производится автоматическое сканирование по процедуре описанной выше.

Создание домена в автоматическом режиме

Чтобы создать домен или поддомен откройте [Меню → Папка с сайтами] и создайте папку с именем будущего домена. После создания домена перезапустите сервер.

Создание домена в ручном режиме

Чтобы создать домен или поддомен перейдите в раздел [Меню → Настройки → Домены] и создайте запись вида: домен => папка. В качестве

папки домена можно выбрать уже существующую папку на диске или создать её непосредственно в окне выбора каталога. После создания домена сохраните настройки.

Создание кириллического домена

OSPanel поддерживает кириллические домены, однако будьте внимательны, папку с доменом нужно называть его реальным именем, а не псевдо названием на кириллице. Для пиво.рф реальным названием (punycode формат) будет xn--b1altb.xn--p1ai и создав такой домен вы получите доступ к http://пиво.рф. Для конвертации доменных имён в punycode формат и обратно используйте [Меню → Дополнительно → IDN конвертер].

Создание поддомена

Процесс создания поддомена аналогичен процедуре создания обычного домена. При создании только поддомена доступность основного домена существующего в сети Интернет не теряется, т.е. вы сможете работать с локальным поддоменом имея при этом доступ к рабочему домену в сети Интернет.

Создание алиаса

Чтобы создать алиас перейдите в раздел [Меню → Настройки → Алиасы] и создайте запись вида: исходный домен => конечный домен. После создания алиаса сохраните настройки.

Обратите внимание - создание алиаса вида *.xxx.xx не означает то, что вам станут доступны любые домены вида test.xxx.xx, mail.xxx.xx и т.д. Всё равно необходимо создать конкретный алиас или домен чтобы он стал доступен, это особенность операционной системы Windows.

Иконка сайта в меню доменов

При наличии корректного файла favicon.ico в корневой папке домена иконка сайта будет отображаться в меню программы.

Ограниченный режим работы

В некоторых случаях управление доменами и алиасами недоступно (см. Ограниченный режим).

Веб-инструменты

Открыть стартовую страницу со ссылками на домены и списком инструментов можно дописав приставку /openserver/ к любому существующему локальному домену. Например: http://localhost/openserver/

Работа с MySQL

Создание пользователя MySQL

1. Откройте [Меню → Дополнительно → PHPMyAdmin]
2. Введите имя пользователя root без пароля (по умолчанию)
3. В PHPMyAdmin откройте раздел [Привилегии]
4. Нажмите ссылку [Добавить нового пользователя]
5. Заполните форму и нажмите кнопку [Создать пользователя]

Информация

В том случае, если помимо прочих привилегий для пользователя будет отмечена привилегия SUPER, то кодировка, установленная по умолчанию в настройках MySQL сервера, не будет на него действовать. Кодировку нужно будет указывать в ваших скриптах персонально для каждого подключения к MySQL, потому отмечать привилегию SUPER не рекомендуется.

Создание базы данных MySQL

1. Откройте [Меню → Дополнительно → PHPMyAdmin]
2. Используйте для входа root без пароля (по умолчанию)
3. В PHPMyAdmin откройте раздел [Базы данных]
4. Введите название новой базы данных и выберите её кодировку
5. Нажмите кнопку [Создать]

Как войти в PHPMyAdmin / MySQL менеджер

1. Откройте [Меню → Дополнительно → PHPMyAdmin / MySQL менеджер]
2. Введите имя пользователя root без пароля
3. Нажмите кнопку [Вход]

Указание кодировки подключения к MySQL серверу

```
<?php
... здесь ваш php код подключения к mysql серверу ...
... далее вставьте строки указанные ниже ...
mysql_query("set names cp1251");
mysql_query("set character_set_server=cp1251");
?>
```

Кодировка cp1251 - русская, её можно заменить на имя любой нужной вам кодировки.

Переключение модулей баз данных

Будьте внимательны при смене активного модуля базы данных. Каждый модуль имеет свое отдельное хранилище баз и настроек, они никак не связаны друг с другом, поэтому вы не увидите созданные вами базы данных при переключении на другой модуль.

Длительное подключение к MySQL (более 1 сек)

Перед началом использования OSPanel следует отключить протокол IPv6 через [Меню → Закладки → Фиксы реестра → Отключение IPv6]. Если этого не сделать, то процесс подключения в MySQL серверу может выполняться очень долго (более 1 сек) и скорость выполнения php скриптов вас не порадует.

Работа с Path

Для добавления собственных путей в переменную окружения PATH можно использовать файл ./userdata/config/path.txt

Пути необходимо добавлять по одному вписывая каждый с новой строки, например:

```
C:\Windows
D:\My Programs
%realprogdir%\data\dll
C:\Windows\System32
```

Информация

По умолчанию файл path.txt не подключается. См. [Меню → Настройки → Сервер].

Использование переменных в качестве подстановок

Переменная	Описание переменной
%realprogdir%	Реальный путь до папки с OSPanel (обратный слеш "\")
%progdir%	Генерируемый путь до папки с OSPanel с учетом виртуального диска (обратный слеш "\")
%sprogdir%	Генерируемый путь до папки с OSPanel с учетом виртуального диска (слеш "/")
%dprogdir%	Генерируемый путь до папки с OSPanel с учетом виртуального диска (двойной обратный слеш "\\")
%dsprogdir%	Генерируемый путь до папки с OSPanel с учетом виртуального диска (двойной слеш "//")
%realsitedir%	Реальный путь до корневой папки доменов (обратный слеш "\")
%sitedir%	Генерируемый путь до корневой папки доменов с учетом виртуального диска (обратный слеш "\")
%ssitedir%	Генерируемый путь до корневой папки доменов с учетом виртуального диска (слеш "/")
%httpport%	Порт HTTP сервера
%httpsport%	Порт HTTPS сервера
%postgresqlport%	Порт PostgreSQL сервера
%mysqlport%	Порт MySQL сервера
%mongodbport%	Порт MongoDB сервера
%memcacheport%	Порт Memcache сервера
%ftpport%	Порт FTP сервера
%httpdriver%	Название модуля HTTP
%phpdriver%	Название модуля PHP
%mysql_driver%	Название модуля MySQL / MariaDB
%pg_driver%	Название модуля PostgreSQL
%mongo_driver%	Название модуля MongoDB
%memcachedriver%	Название модуля Memchache
%dnsdriver%	Название модуля DNS
%ip%	IP адрес сервера
%disk%	Буква диска из генерируемого пути до папки с OSPanel с учетом виртуального диска (только буква)
%osdisk%	Буква диска из реального пути до папки с OSPanel (только буква)
%sysdisk%	Системный диск Windows (только буква)

DNS сервер

Встроенный DNS сервер предназначен для использования в локальных сетях или для отладки веб-приложений. Для детальной настройки доступна общая конфигурация сервера, а так же конфигурация доменов.

Для использования встроенного DNS сервера необходимо выполнить настройку сетевого интерфейса на каждом компьютере ДО запуска самого сервера. Выполнить настройку необходимо как на локальной машине, так и на других компьютерах в локальной сети, которые хотят получить доступ к вашим доменам. Без указания локального DNS сервера в настройках сетевого подключения запуск сервера будет невозможен (если модуль DNS активирован в настройках OSPanel).

По умолчанию параметр TTL установлен в значение 60 (секунд), вы можете изменить это значение в файле ./userdata/init.ini однако следует иметь ввиду, что бездумное изменения TTL может спровоцировать кэширование неактуальных записей другими компьютерами в вашей сети. Изменяйте этот параметр только в том случае, если вы действительно понимаете его предназначение.

Рекомендации по настройке

1. В качестве IP адреса сервера выберите в настройках OSPanel IP адрес вашего компьютера в локальной сети или сети Интернет (не выбирайте параметр Все доступные IP).
2. Откройте свойства нужного сетевого подключения:
   Центр управления сетями → Подключение xxx → Свойства →
   Протокол Интернета версии 4 → Свойства → Общие → Использовать следующие адреса DNS-серверов.

   Пропишите следующие адреса NS серверов:

1. xxx.xxx.xxx.xxx
2. 8.8.8.8 (или любой другой реальный резервный DNS)

Вместо xxx.xxx.xxx.xxx впишите IP адрес вашего компьютера в локальной сети или сети Интернет.

3. Повторите процедуру настройки на других компьютерах в локальной сети.
4. Не выполняйте настройку алиасов из инструкции раздела Внешний доступ данного руководства. Подобная настройка не требуется, поскольку удалённые компьютеры будут напрямую работать с вашим DNS сервером и получат доступ ко всем доменам.
5. Сохраните настройки и выполните запуск сервера OSPanel.

После правильной настройки все компьютеры в вашей локальной сети смогут получить доступ к доменам OSPanel.

Ограниченный режим

При недоступном на запись HOSTS файле (как правило такое происходит в случае запуска программы без прав администратора) программа переходит в ограниченный режим работы с урезанной функциональностью.

В ограниченном режиме вам будет недоступна следующая функциональность:

- Использование своих алиасов и доменов (кроме домена localhost);
- Указание IP адреса сервера (кроме адресов 127.0.0.1 и *);

Таким образом, в ограниченном режиме вам будет доступен один из двух IP адресов: 127.0.0.1 или *, и только один домен localhost. Другие созданные вами алиасы и домены не будут обработаны программой.

Работать в ограниченном режиме можно только при полном отсутствии прав администратора. Если же у вас есть возможность запускать программу с нужными правами, то рекомендуется непременно этим воспользоваться. В случае, если вы постоянно забываете запускать программу с необходимыми правами, включите опцию [Требовать учётную запись Администратора].

Если в логах запуска вы видите сообщение о том что Hosts файл недоступен для записи и OSPanel запущен с правами администратора, то это означает что доступ к этому файлу блокируется антивирусами/файерволами (даже для доверенных программ) либо действуют ограничения прав доступа Windows.

Добавьте OSPanel, а так же все остальные компоненты о которых будет спрашивать антивирус/файервол, в доверенные программы. Отключите защиту HOSTS файла (или системных файлов) в настройках вашего антивируса/файервола, если такая защита присутствует. Попробуйте удалить файл C:\Windows\System32\Drivers\etc\hosts и заново создать со следующим содержимым:

```
1.  127.0.0.1 localhost
```

В случае работы без прав администратора, но с доступным на запись HOSTS файлом, программа работает в нормальном режиме без каких-либо ограничений. Разрешить запись в HOSTS файл для всех пользователей можно выполнив через консоль (запускать от имени Администратора) следующую команду:

```
1.  attrib -s -r -h -a C:\Windows\system32\drivers\etc\hosts
```

Внимание

При включённой службе контроля учётных записей пользователей (UAC) и запуске без прав администратора OSPanel не будет иметь доступа к HOSTS файлу и автоматически перейдёт в ограниченный режим работы.

Запуск без внесения записей в HOSTS файл

BOSPanel реализована возможность полноценного запуска без внесения записей в HOSTS файл. Эта возможность будет полезна пользователям офисных сетей и терминалов, где доступ к HOSTS файлу имеет только старший администратор. Если опция [Не вносить изменения в HOSTS файл] включена, то запуск сервера происходит без редактирования HOSTS файла, т.е. доступ к этому файлу не требуется вовсе.

Следует знать и понимать, что во время запуска программа делает DNS запрос к каждому созданному вами домену и если в HOSTS файле не будет записи любого из доменов, то это приведёт к ошибке [Сбой запуска]. Не забывайте обращаться к администратору вашей сети после каждого создания домена, администратор должен внести нужные записи в HOSTS файл, иначе вы не сможете запустить сервер.

Формат внесения записей в HOSTS файл стандартный - *ip пробел домен*, например:

```
1.   192.168.5.10  rhino.acme.com
2.   192.168.5.10  x.acme.com
```

Защита сервера

Настройка защиты

Когда сервер открыт для доступа из сети Интернет он становится крайне уязвимым, особенно с настройками которые установлены по умолчанию. Множество пауков и вирусов постоянно сканируют Интернет на предмет открытых портов и, как правило, незащищенный сервер оказывается взломанным уже через несколько часов после появления в сети.

Несколько шагов по защите веб-сервера от несанкционированного доступа:

1. Смените стандартные пароли FTP пользователя [Меню → Настройки → FTP сервер];
2. **Установите собственные пароли для root (и других) пользователей всех модулей СУБД;**
3. **Включите защиту веб-инструментов и диска от доступа из внешних сетей в настройках программы [Меню → Настройки → Сервер];**
4. Теперь перезапустите саму управляющую программу (не сервер);

5. Выполните настройку фаервола, закрыв на доступ извне ВСЕ порты кроме тех, которые планируется использовать (например: 80,443,21,990,53);

Внимание

Веб-сервер работает от имени администратора вашего компьютера, при работе на внешних IP адресах или IP = * безопасность вашего компьютера будет под угрозой! Не допускайте использования уязвимых скриптов, некорректной конфигурации модулей, простых паролей.

Встроенная защита от внешнего доступа

Выбор опции [Защитить сервер от внешнего доступа] отключит часть опасных функций PHP, доступ к веб-инструментам извне будет заблокирован, а доступ к диску для PHP скриптов будет ограничен корневой папкой доменов. Данная опция снижает производительность PHP скриптов в 3-4 раза.

Более детальную информацию по настройке сервера можно прочитать по ссылке https://ospanel.io/docs/

Учебное пособие по MySQL

Использование клиентской программы `mysql` для создания несложной базы данных и работы с ней. Утилита `mysql` (иногда называемая также «терминальным монитором'» или просто «монитором'») представляет собой интерактивную программу, позволяющую подсоединяться к MySQL-серверу, запускать запросы, и просматривать результаты. Программа `mysql` может работать и в пакетном режиме: для этого необходимо записать все запросы в файл, а затем передать его содержимое на исполнение `mysql`. Ниже описаны оба способа использования `mysql`.

Увидеть список команд программы mysql можно, запустив ее с параметром --help:

```
shell> mysql --help
```

При подключении к серверу с помощью `mysql` обычно нужно ввести имя пользователя MySQL и, в большинстве случаев, пароль. Если сервер запущен не на том компьютере, с которого вы вошли в систему, необходимо также указать имя хоста. Параметры соединения (а именно - соответствующее имя хоста, пользователя и пароль) вы сможете узнать у администратора. Получив соответствующие параметры, подсоединиться к серверу можно следующим образом:

```
shell> mysql -h host -u user -p
Enter password: ********
```

Символы ******** обозначают ваш пароль; введите его, когда `mysql` выведет на экран запрос `Enter password:`.

Если все сработает, на экране должна появиться следующая информация и метка командной строки `mysql>`:

```
shell> mysql -h host -u user -p
Enter password: ********

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 459 to server version: 3.22.20a-log

Type 'help' for help.

mysql>
```

Метка обозначает, что программа `mysql` готова к вводу команд.

В некоторых вариантах установки MySQL возможно подсоединение к запущенному на локальном хосте серверу без ввода имени пользователя (пользователь anonymous). Если ваша система настроена именно так, подсоединиться к серверу вы сможете, запустив mysql со следующими параметрами:

```
shell> mysql
```

После установки соединения можно в любой момент отключиться от сервера, набрав в командной строке `mysql>` команду `QUIT`:

```
mysql> QUIT
Bye
```

Отсоединиться от сервера можно и при помощи сочетания клавиш Control-D.

Большая часть приведенных ниже примеров построена с учетом того, что соединение с сервером уже установлено. Это видно по наличию в них командной строки mysql>.

Создание и выбор базы данных

Если администратор при выдаче разрешения создаст для вас базу, с ней можно сразу начинать работу. В противном случае вам придется создать ее самостоятельно:

```
mysql> CREATE DATABASE menagerie;
```

В Unix имеет значение регистр символов в именах баз данных (в отличие от ключевых слов SQL), так что в этой ОС вам всегда придется называть свою базу menagerie, а не Menagerie, MENAGERIE или еще как-нибудь. Это же правило распространяется и на имена таблиц (в Windows данное ограничение не действует, однако при обращении к базам и таблицам в пределах одного запроса, тем не менее, можно использовать только один регистр).

При создании базы данных она автоматически не выбирается; выбирать ее нужно отдельно. Сделать menagerie текущей базой можно с помощью следующей команды:

```
mysql> USE menagerie
Database changed
```

Создавать базу нужно только однажды, но выбирать ее приходится в каждом сеансе работы с `mysql`. Делать это можно с помощью команды USE, представленной выше. А можно выбирать базу и из командной строки при запуске `mysql`. Для этого достаточно лишь ввести ее имя после параметров соединения, которые нужно вводить в любом случае. Например:

```
shell> mysql -h host -u user -p menagerie
Enter password: ********
```

Обратите внимание: в вышеприведенной команде menagerie не является паролем. Ввести пароль в командной строке после параметра -p можно без пробела (например, -pmypassword, а не -p mypassword). Впрочем, пароль в командной строке все равно лучше не вводить, так как таким образом его могут и подсмотреть.

Создание таблицы

Как вы уже успели убедиться, создать базу данных было просто. Однако пока что в ней ничего нет - в этом можно удостовериться при помощи команды SHOW TABLES:

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

Гораздо труднее определиться со структурой своей базы, т.е. с тем, какие могут понадобиться таблицы, и какие столбцы должны содержаться в каждой из них.

Вам обязательно будет нужна таблица, содержащая по записи на каждое из животных. Назвать ее можно `pet`, и храниться в ней будут, как минимум, имена. Но так как само по себе имя неинформативно, в таблице должны будут присутствовать и другие данные. Например, если домашние животные есть более чем у одного члена вашей семьи, в таблицу можно добавить и имя владельца каждого животного. Кроме того, в базу стоит внести и описательную информацию - например, вид и пол животного.

Но вот как быть с возрастом? Эта информация тоже может оказаться полезной, но хранить такие данные в базе неудобно. Возраст со временем меняется, а это значит, что придется довольно часто обновлять записи. Значительно удобнее хранить фиксированные значения - например, даты рождения. В таком случае возраст всегда можно получить, вычислив разницу между текущей датой и датой рождения. В MySQL есть функции для арифметических действий над данными, так что это совсем несложно. Хранение даты рождения имеет и другие преимущества:

- Базу данных можно использовать для выдачи напоминаний о приближающихся днях рождения.

- Возраст можно подсчитывать относительно любой даты, а не только для текущей. Например, если записать в базу и дату смерти животного, всегда можно будет узнать, сколько лет ему было на момент смерти.

Можно было бы придумать и еще какие-нибудь данные, которые неплохо было бы хранить в таблице pet, но пока что мы ограничимся уже выбранными: именем (`name`), именем владельца (`owner`), видом (`species`), полом (`sex`), датой рождения (`birth`) и датой смерти (`death`).

При помощи команды `CREATE TABLE` определим структуру новой таблицы:

```
mysql> CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20),
    -> species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);
```

Тип `VARCHAR` отлично подойдет для хранения имени животного, имени владельца и названия вида, так как длина данных этого типа может варьироваться. Конечно, длины таких столбцов вовсе не должны совпадать и не должны быть равны 20 - можно выбрать любое значение в пределах от 1 до 255 (если при выборе длины столбца вы ошибетесь, и при работе с базой окажется, что столбец маловат, можно будет исправить ошибку при помощи команды `ALTER TABLE`).

Пол животного можно обозначать несколькими способами, например буквами "m" и "f", или словами male (мужской) и female (женский). С буквами "m" и "f" будет проще.

Применение типа данных `DATE` для хранения дат рождения и смерти вполне очевидно.

Теперь, когда таблица создана, команда `SHOW TABLES` должна вывести следующее:

```
mysql> SHOW TABLES;
       +---------------------+
       | Tables in menagerie |
       +---------------------+
       | pet                 |
       +---------------------+
```

Проверить, правильно была ли таблица создана в соответствии с планом, можно при помощи команды `DESCRIBE`:

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | YES  |     | NULL    |       |
| owner   | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex     | char(1)     | YES  |     | NULL    |       |
| birth   | date        | YES  |     | NULL    |       |
| death   | date        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

Загрузка данных в таблицу

Создав таблицу, нужно позаботиться об ее заполнении. Для этого предназначены команды `LOAD DATA` и `INSERT`.

Предположим, ваши записи соответствуют приведенным в этой таблице (обратите внимание: MySQL принимает даты в формате `ГГГГ-ММ-ДД`; возможно, к такой записи вы не привыкли).

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1998-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	

Так как вы начинаете работу с пустой таблицей, заполнить ее будет проще всего, если создать текстовый файл, содержащий по строке на каждое из животных, а затем загрузить его содержимое в таблицу одной командой.

Создайте текстовый файл с именем `` `pet.txt' ``, содержащий по одной записи в каждой строке (значения столбцов должны быть разделены символами табуляции и даны в том порядке, который был определен командой `CREATE TABLE`). Незаполненным полям (например, неизвестный пол или даты смерти живых на сегодняшний день животных), можно присвоить значение `NULL`. В текстовом файле это значение представляется символами `\N`. Например, запись для птицы `Whistler` должна выглядеть

примерно так (между значениями должны располагаться одиночные символы табуляции):

name	owner	species	sex	birth	death
Whistler	Gwen	bird	\N	1997-12-09	\N

Загрузить файл `pet.txt' в таблицу можно с помощью следующей команды:

```
mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
```

Маркер конца строки и символ, разделяющий значения столбцов, можно специально задать в команде LOAD DATA, но по умолчанию используются символы табуляции и перевода строки. Воспринимая их, команда сможет корректно прочитать файл `pet.txt'.

При добавлении одиночных записей используется команда INSERT. В самом простом варианте ее применения необходимо задать значения каждого столбца, в том порядке, в каком они были перечислены в команде CREATE TABLE. Предположим, Диана (Diane) купила хомячка по имени Puffball. Соответствующую запись в таблицу с можно внести с помощью команды INSERT примерно так:

```
mysql> INSERT INTO pet
    -> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);
```

Обратите внимание на то, что здесь строковые выражения и даты представлены в виде ограниченных кавычками строк. Кроме того, в команде INSERT отсутствующие данные можно прямо заменять на NULL. Пользоваться эвфемизмом \N, как в команде LOAD DATA, нужды нет.

Этот пример наглядно показывает, что если бы с самого начала все данные вносились в базу при помощи нескольких команд INSERT, а не одной команды LOAD DATA, то набирать пришлось бы гораздо больше текста.

Выборка информации из таблицы

Информация извлекается из таблиц при помощи команды SELECT.

Вызывается она так:

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy
```

где what_to_select обозначает нужные данные. Это может быть список столбцов или символ * (``все столбцы''). which_table указывает таблицу, из которой должны быть извлечены данные. Условие WHERE использовать необязательно, но если оно все же присутствует в вызове команды, то параметр conditions_to_satisfy задает условия, которым должны соответствовать нужные строки.

Выборка всех данных

В самом простом варианте вызова SELECT из таблицы извлекаются сразу все данные:

```
mysql> SELECT * FROM pet;
+----------+--------+---------+------+------------+------------+
| name     | owner  | species | sex  | birth      | death      |
+----------+--------+---------+------+------------+------------+
| Fluffy   | Harold | cat     | f    | 1993-02-04 | NULL       |
| Claws    | Gwen   | cat     | m    | 1994-03-17 | NULL       |
| Buffy    | Harold | dog     | f    | 1989-05-13 | NULL       |
| Fang     | Benny  | dog     | m    | 1990-08-27 | NULL       |
| Bowser   | Diane  | dog     | m    | 1998-08-31 | 1995-07-29 |
| Chirpy   | Gwen   | bird    | f    | 1998-09-11 | NULL       |
| Whistler | Gwen   | bird    | NULL | 1997-12-09 | NULL       |
| Slim     | Benny  | snake   | m    | 1996-04-29 | NULL       |
| Puffball | Diane  | hamster | f    | 1999-03-30 | NULL       |
+----------+--------+---------+------+------------+------------+
```

Использовать SELECT таким образом удобно, когда нужно просмотреть всю таблицу, например, после того, как в нее была загружена первая порция данных. Как часто случается, в выведенных на экран данных сразу можно увидеть ошибку в таблице: Bowser, оказывается, успел умереть еще до того, как родился! Заглянув в его родословную обнаруживаем, что пес родился в 1989, а не в 1998 году.

Исправить ошибку можно как минимум двумя способами:

Отредактировать файл pet.txt, затем очистить таблицу и снова заполнить ее командами DELETE и LOAD DATA:

- `mysql> SET AUTOCOMMIT=1;  # Used for quick re-create of the table`

```
mysql> DELETE FROM pet;
mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
```

Однако в таком случае придется снова ввести запись о Puffball.

- Исправить только ошибочно введенные данные при помощи команды `UPDATE`:

```
mysql> UPDATE pet SET birth = "1989-08-31" WHERE name = "Bowser";
```

Как видно из приведенного выше примера, загрузить всю таблицу сразу очень просто. Но на практике обычно этого не требуется, особенно когда таблица достигает значительных размеров. Чаще всего нужно просто ответить на какой-нибудь вопрос, для чего необходимо ввести ограничения, указывающие, какая же информация вам нужна. Давайте рассмотрим несколько запросов с точки зрения вопросов, на которые они отвечают.

Выборка определенных строк

Из таблицы можно выбрать и только нужные строки. Например, если вы хотите проверить правильность внесенных в дату рождения собаки Bowser изменений, соответствующую запись можно получить следующим способом:

```
mysql> SELECT * FROM pet WHERE name = "Bowser";
+--------+-------+---------+------+------------+------------+
| name   | owner | species | sex  | birth      | death      |
+--------+-------+---------+------+------------+------------+
| Bowser | Diane | dog     | m    | 1989-08-31 | 1995-07-29 |
+--------+-------+---------+------+------------+------------+
```

Теперь видно, что год рождения теперь записан правильно -1989, а не 1998.

В операции сравнения строк обычно не учитывается регистр символов, так что имя можно записать как `"bowser"`, `"BOWSER"` и т.п. Результаты запросов будут идентичными.

В условиях может указываться любой из столбцов, а не только `name`. Если, например, вам нужно узнать, какие их животных родились после 1998 года, в условие вводится значение столбца `birth`:

```
mysql> SELECT * FROM pet WHERE birth >= "1998-1-1";
+----------+-------+---------+------+------------+-------+
| name     | owner | species | sex  | birth      | death |
+----------+-------+---------+------+------------+-------+
| Chirpy   | Gwen  | bird    | f    | 1998-09-11 | NULL  |
| Puffball | Diane | hamster | f    | 1999-03-30 | NULL  |
+----------+-------+---------+------+------------+-------+
```

Условия можно и комбинировать, например для того, чтобы выделить всех собак женского пола:

```
mysql> SELECT * FROM pet WHERE species = "dog" AND sex = "f";
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

В предыдущем запросе использован оператор `AND`. Существует еще и оператор `OR`:

```
mysql> SELECT * FROM pet WHERE species = "snake" OR species = "bird";
+----------+-------+---------+------+------------+-------+
| name     | owner | species | sex  | birth      | death |
+----------+-------+---------+------+------------+-------+
| Chirpy   | Gwen  | bird    | f    | 1998-09-11 | NULL  |
| Whistler | Gwen  | bird    | NULL | 1997-12-09 | NULL  |
| Slim     | Benny | snake   | m    | 1996-04-29 | NULL  |
+----------+-------+---------+------+------------+-------+
```

Операторы `AND` и `OR` можно использовать совместно. В таком случае с помощью скобок можно указать порядок группировки условий:

```
mysql> SELECT * FROM pet WHERE (species = "cat" AND sex = "m")
    -> OR (species = "dog" AND sex = "f");
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen   | cat     | m    | 1994-03-17 | NULL  |
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
```

Выборка определенных столбцов

Если строки целиком вам не нужны, достаточно просто перечислить имена нужных столбцов, разделив их запятыми. Например, если вы хотите узнать, когда родились животные, выберите столбцы с именами и датами рождения:

```
mysql> SELECT name, birth FROM pet;
+----------+------------+
| name     | birth      |
+----------+------------+
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Buffy    | 1989-05-13 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Получить имена владельцев животных можно с помощью следующего запроса:

```
mysql> SELECT owner FROM pet;
+--------+
| owner  |
+--------+
| Harold |
| Gwen   |
| Harold |
| Benny  |
| Diane  |
| Gwen   |
| Gwen   |
| Benny  |
| Diane  |
+--------+
```

Однако этот запрос просто загружает поля с именем владельца из каждой записи, а некоторые имена встречаются более одного раза. Сократить количество выводимых строк можно, воспользовавшись ключевым словом `DISTINCT` - тогда будут выводиться только уникальные записи:

```
mysql> SELECT DISTINCT owner FROM pet;
+--------+
| owner  |
+--------+
| Benny  |
| Diane  |
| Gwen   |
| Harold |
+--------+
```

Сортировка строк

Вы уже, наверное, заметили, что результаты, выдававшиеся запросами из предыдущих примеров, выводились без какой-либо сортировки. Но ведь

часто разобраться в результатах легче, если они отсортированы. Для этого используется выражение `ORDER BY`.

Так выглядят даты рождения животных в отсортированном виде:

```
mysql> SELECT name, birth FROM pet ORDER BY birth;
+----------+------------+
| name     | birth      |
+----------+------------+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
| Fang     | 1990-08-27 |
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Slim     | 1996-04-29 |
| Whistler | 1997-12-09 |
| Chirpy   | 1998-09-11 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Над столбцами с символьными значениями операция сортировки - как и все другие операции сравнения - обычно проводится без учета регистра символов. Это значит, что порядок расположения столбцов, совпадающих во всем, кроме регистра символов, относительно друг друга будет не определен. Провести сортировку с учетом регистра символов можно при помощи команды `BINARY`: `ORDER BY BINARY(поле)`.

Для сортировки в обратном порядке к имени столбца следует добавить ключевое слово `DESC` (по убыванию):

```
mysql> SELECT name, birth FROM pet ORDER BY birth DESC;
+----------+------------+
| name     | birth      |
+----------+------------+
| Puffball | 1999-03-30 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Claws    | 1994-03-17 |
| Fluffy   | 1993-02-04 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+----------+------------+
```

Подсчет строк

Базы данных часто используются для получения ответа на вопросы типа: ``как часто данные определенного типа встречаются в таблице?" Вам, например, может понадобиться узнать общее количество животных, или то,

сколько животных имеется у каждого из владельцев, или провести статистические исследования на базе хранящейся информации.

Процедура подсчета количества животных в сущности идентична подсчету количества строк в таблице, так как на каждое животное приходится по одной записи. Функция `COUNT()` подсчитает количество непустых результатов, и с ее помощью можно составить следующий запрос для определения числа животных:

```
mysql> SELECT COUNT(*) FROM pet;
+----------+
| COUNT(*) |
+----------+
|        9 |
+----------+
```

Количество животных каждого пола:

```
mysql> SELECT sex, COUNT(*) FROM pet GROUP BY sex;
+------+----------+
| sex  | COUNT(*) |
+------+----------+
| NULL |        1 |
| f    |        4 |
| m    |        4 |
+------+----------+
```

(в этой таблице результатов `NULL` обозначает, что пол животного неизвестен).

Получение информации о базах данных и таблицах

Как быть, если вы забыли имя базы или таблицы, или структуру какой-либо из таблиц (например, имена столбцов)? В MySQL эта проблема решается при помощи нескольких команд, выводящих информацию о базе данных и содержащихся в ней таблицах.

Вы уже познакомились с командой `SHOW DATABASES`, выводящей список управляемых сервером баз данных. Определить, какая из них выбрана в данный момент, можно с помощью функции `DATABASE()`:

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| menagerie  |
+------------+
```

Если ни одна из баз не выбрана, результат будет пуст.

Выяснить, какие таблицы содержит текущая база данных (что необходимо, если, например, никак не получается вспомнить имя нужной таблицы), можно при помощи следующей команды:

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| event               |
| pet                 |
+---------------------+
```

Узнать структуру таблицы можно при помощи команды `DESCRIBE`, которая выводит информацию о каждом из столбцов таблицы:

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | YES  |     | NULL    |       |
| owner   | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex     | char(1)     | YES  |     | NULL    |       |
| birth   | date        | YES  |     | NULL    |       |
| death   | date        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

`Field` - имя столбца, `Type` - тип данных, к которому относится этот столбец, `NULL` указывает, может ли данный столбец содержать значения `NULL`, Key - является ли этот столбец индексным, и, наконец, `Default` указывает значение данного столбца по умолчанию.

Если для таблицы созданы индексы, информацию о них можно получить с помощью команды `SHOW INDEX FROM tbl_name`.

Примеры стандартных запросов

Здесь представлены примеры решения некоторых стандартных задач средствами MySQL.

В некоторых из примеров используется таблица `shop` (магазин), в которой содержатся цены по каждому изделию (`item number`)для определенных продавцов (`dealer`). Предположим, что каждый продавец имеет одну фиксированную цену для каждого изделия; тогда пара изделие-

продавец (`article, dealer`) является первичным ключом для записей таблицы.

Запустите клиента `mysql`:

```
mysql имя-вашей-базы-данных
```

(Для большинства инсталляций MySQL можно использовать базу данных '`test`'.)

Таблицу примера можно создать таким образом:

```
CREATE TABLE shop (
 article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
 dealer  CHAR(20)                 DEFAULT ''     NOT NULL,
 price   DOUBLE(16,2)             DEFAULT '0.00' NOT NULL,
 PRIMARY KEY(article, dealer));

INSERT INTO shop VALUES
(1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),(3,'C',1.69),
(3,'D',1.25),(4,'D',19.95);
```

Ну и, скажем, данные для примера будут такими:

```
mysql> SELECT * FROM shop;

+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0001 | A      |  3.45 |
|    0001 | B      |  3.99 |
|    0002 | A      | 10.99 |
|    0003 | B      |  1.45 |
|    0003 | C      |  1.69 |
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

Максимальное значение столбца

"Как определить наибольшее значение в столбце?"

```
SELECT MAX(article) AS article FROM shop
+---------+
| article |
+---------+
|       4 |
+---------+
```

Строка, содержащая максимальное значение некоторого столбца

В ANSI SQL это легко делается при помощи вложенного запроса:

```
SELECT article, dealer, price
FROM   shop
WHERE  price=(SELECT MAX(price) FROM shop)
```

В MySQL (в котором вложенные операторы SELECT еще не реализованы) такая задача выполняется в два этапа:

1. Следует получить максимальное значение цены из таблицы при помощи оператора SELECT.
2. Используя это значение, необходимо составить следующий запрос:

```
SELECT article, dealer, price
FROM   shop
WHERE  price=19.95
```

Существует еще одно решение: отсортировать все строки по убыванию цен и после этого получить первую строку, используя специальный оператор LIMIT:

```
SELECT article, dealer, price
FROM   shop
ORDER BY price DESC
LIMIT 1
```

Примечание: если существует несколько самых дорогих изделий (например, каждое из них стоит 19,95), запрос, использующий `LIMIT`, возвращает лишь одно из них!

Более подробную информацию по реализации запросов можно получить по ссылке: http://www.mysql.ru/docs/man/example-Maximum-row.html

Подробное руководство по работе с MySQL можно получить по ссылке http://www.mysql.ru/docs/man/

Использование PhpMyAdmin

После установки open server для работы с базой данных (создание базы данных, создание, добавление таблиц, запросы) можно также выполнять с помощью PhpMyAdmin. Ниже приведено одно из возможных описаний работы с MySQL с помощью PhpMyAdmin.

http://webmastertema.ru/sajtostroenie/sozdanie-sajta-na-lokalnom-servere-openserver.html

Задание к индивидуальной работе и требования к отчету

1) Скачать и установить Open Server под свою операционную систему. Привести скриншот и краткое описание.
2) Настроить Open Server (минимальные настройки для работы с MySQL). Привести скриншот и краткое описание.
3) Создать базу данных (вариант тот же, что и для лаб. работ). Для создания базы данных использовать как режим командной строки (консоль), так и возможности PhpMyAdmin. Привести скриншот.
4) Создать таблицы базы данных. В рамках индивидуальной работы можно создать не все таблицы, но кол-во таблиц должно быть не менее 4-х. Привести скриншоты процесса создания таблиц.
5) Добавить любым удобным способом 3-4 записи в каждую из созданных таблиц. Привести скриншоты процесса добавления записей в таблицу.
6) Выполнить 2 запроса на выборку (симметричные запросы с условием, без условия, с сортировкой в прямом и в обратном порядке). Привести в отчете формулировку запроса, его реализацию на языке SQL, результаты работы запросов на выборку.
7) Выполнить 1 запрос итоговый, привести в отчете формулировку запроса, его реализацию на языке SQL, результаты работы итогового запроса.

   Отчет по индивидуальной работе должен быть не более 12 страниц.

Контрольные вопросы:

1.Что такое СУБД?

2.Какие типы СУБД Вы знаете?

3.Какие сложности возникли при настройке Open Server?

4.Как создать базу данных?

5.Как добавить таблицу в базу данных?

6.Как добавить записи в таблицу?

7.Как удалить записи из таблицы?

МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ
к индивидуальной работе по дисциплине

«Организация баз данных и знаний»

(для студентов направлений подготовки 09.03.02 "Информационные системы и технологии" (специальностей "Информационные технологии в медиаиндустрии и дизайне", "Системы автоматизированного проектирования"), 02.03.01 "Математика и компьютерные науки" (специальность "Компьютерное моделирование и дизайн")

Составители: Алла Викторовна Чернышова