Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Институт повышения квалификации
и переподготовки кадров

Кафедра «Информатика»

Т. В. Тихоненко

СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

КУРС ЛЕКЦИЙ
по одноименной дисциплине для слушателей специальности 1-40 01 74 «Web-дизайн и компьютерная графика» заочной формы обучения

Гомель 2014

УДК 004.65(075.8)
ББК 32.973-018.2я73
Т46

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 4 от 25.11.2013 г.*)

Рецензент: доц. каф. «Детали машин» ГГТУ им. П. О. Сухого
канд. техн. наук *В. В. Комраков*

Тихоненко, Т. В.

Т46 Системы управления базами данных : курс лекций по одноим. дисциплине для слушателей специальности 1-40 01 74 «Web-дизайн и компьютерная графка» / Т. В. Тихоненко. – Гомель : ГГТУ им. П. О. Сухого, 2014. – 74 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://library.gstu.by. – Загл. с титул. экрана.

Данное издание представляет собой пособие по теоретическому и практическому освоению технологий создания баз данных, на примере системы управления базами данных – MySQL. Содержит множество примеров, позволяющих в кратчайшие сроки овладеть основными принципами работы с базами данных в MySQL.

Для слушателей специальности 1-40 01 74 «Web-дизайн и компьютерная графика» заочной формы обучения ИПК и ПК

УДК 004.65(075.8)
ББК 32.973-018я73

Содержание

1 Краткие теоретические сведения о базах данных и системах управления базами данных

1.1 Понятие базы данных, классификация баз данных

База данных – именованная совокупность структурированных данных, относящихся к некоторой предметной области, и хранящаяся в файлах.

База данных представляет собой компьютерный аналог организованной информации. Обычно элементы информации объединяет общая тема или назначение.

Основные отличительные признаки:

1. база данных хранится и обрабатывается в вычислительной системе;
2. данные в базе данных логически структурированы (систематизированы);
3. база данных включает метаданные, описывающие логическую структуру базы данных в формальном виде (в соответствии с некоторой метамоделью).

Из перечисленных признаков только первый является строгим, а другие допускает различные трактовки и различные степени оценки.

При размещении данных в базе данных определяется *физическая* и *логическая* организация данных. Физическая организация данных связана с размещением данных на реальных машинных носителях информации и в современных базах данных обеспечивается автоматически. Логическая организация данных определяется типом структур данных и видом модели.

Существует огромное количество разновидностей баз данных, отличающихся по различным критериям (например, в [1] определяются свыше 50 видов баз данных).

Укажем только основные классификации.

1. *Классификация по модели данных. Модель данных*– это совокупность структур данных и операций их обработки. Выделяют следующие модели данных: иерархические; сетевые; реляционные; объектные; объектно-ориентированные; объектно-реляционные.
2. *Классификация по технологии хранения.* Выделяют базы данных во вторичной памяти (традиционные); базы данных в оперативной памяти; базы данных в третичной памяти.

3. *Классификация по содержимому.* Выделяют географические; исторические; научные; мультимедийные и т.д.

1.2 Понятие системы управления базами данных, классификация

Для управления работой баз данных используется *система управления базами данных (СУБД)*.

СУБД – комплекс программ, необходимых для создания баз данных, внесения в них изменений и организации поиска необходимой информации.

Примеры СУБД: Paradox, MS Access, FoxPro, MSSQLServer, Oracle, MySQLи др.

Основные функции СУБД:

1. управление данными во внешней памяти (на дисках);
2. управление данными в оперативной памяти с использованием дискового кэша;
3. журнализация изменений, резервное копирование и базы данных после сбоев;
4. поддержка языков базы данных (*DDL* – язык определения данных, *DML* – язык манипулирования данными).

Обычно современная СУБД содержит следующие компоненты: ядро, процессор, подсистему поддержки времени исполнения, сервисные программы (внешние утилиты).

Классификация СУБД по способу доступа к базе данных: файл-серверные; клиент-серверные; встраиваемые.

1.3 Основные понятия реляционной модели данных

Реляционные базы данных в настоящее время наиболее распространены и фактически являются промышленным стандартом. В реляционных базах данных данные хранятся в двумерных таблицах. Строки реляционной таблицы соответствуют записям, а столбцы – полям.

Поле – элементарная единица логической организации данных, соответствующая логически неделимой единице информации (реквизиту).

Поле характеризуется: именем; типом; длиной (место, занимаемое в оперативной памяти); точностью числовых данных (количество знаков после запятой).

Запись – это совокупность логически связанных полей.

Таблица – совокупность записей одной структуры.

Ключ – уникальный идентификатор, состоящий из одного поля (простой ключ) или нескольких полей (составной ключ), однозначно определяющий запись.

Наличие ключа позволяет устранить избыточность и дублирование данных. Если среди реальных реквизитов такого нет, то можно добавить в данные дополнительный идентификатор. Рациональный способ организации таблиц позволяет свести дублирование данных к минимуму.

Требования к реляционным таблицам:

1. каждый элемент таблицы – один элемент данных;
2. все столбцы в таблице однородные, т. е. имеют одинаковый тип данных;
3. каждый столбец имеет уникальное имя;
4. одинаковые строки в таблице отсутствуют;
5. порядок следования строк и столбцов произвольный.

Таким образом, реляционная база данных является совокупностью двумерных таблиц, состоящих из записей и полей. Между таблицами устанавливаются логические связи, реализуемые за счет наличия одинаковых полей (ключей) в связываемых таблицах. В одной из таблиц ключ должен быть первичным (уникальным), а во второй внешним (повторяющимся).

1.4 Проектирование баз данных, нормализация

В результате проектирования базы данных должна быть разработана *информационно-логическая модель* данных. Компонентами такой модели являются *информационные объекты* и структурные *связи* между ними.

Информационный объект – это информационное отображение некоторого реального объекта, явления, процесса, информация о котором должна быть представлена в базе данных, в виде совокупности логически связанных реквизитов (информационных элементов, атрибутов).

Например, информационный объект СТУДЕНТ имеет реквизиты: НОМЕР ЗАЧЕТКИ (ключевое поле), ФАМИЛИЯ, ИМЯ, ДАТА РОЖДЕНИЯ и т.д. Информационный объект может иметь один первичный и несколько внешних ключей, которые используются для связи с другими информационными объектами.

Различают три типа связей между информационными объектами, они представлены в таблице 1.

Таблица 1 – Типы связей

Тип связи (обозначение)	Описание	Пример
один к одному (1:1)	Предполагает, что одному экземпляру первого информационного объекта соответствует только один экземпляр второго информационного объекта и наоборот. Такие информационные объекты можно объединить в один, содержащий атрибуты двух объектов.	Связь между объектами СТУДЕНТ и СЕССИЯ, когда каждый студент имеет определённый набор экзаменационных оценок в сессию.
один ко многим (1:∞)	Означает, что одному экземпляру первого информационного объекта соответствует 0, 1 или более экземпляров второго информационного объекта, но каждому экземпляру второго информационного объекта обязательно соответствует один экземпляр первого информационного объекта.	Связь между объектами ГРУППА и СТУДЕНТ, когда название группы может повторяться многократно для различных студентов.
многие ко многим (∞:∞)	Предполагает, что одному экземпляру первого информационного объекта соответствует 0, 1 или более экземпляров второго информационного объекта и наоборот.	Связь между объектами СТУДЕНТ и ПРЕПОДАВАТЕЛЬ, когда один студент обучается у многих преподавателей, а один преподаватель обучает многих студентов.

Информационные объекты должны быть представлены в виде реляционных таблиц. Связи между реляционными таблицами устанавливаются при помощи совпадающих значений полей. Для связей 1:1 и 1:∞ каждый информационный объект представляется соответствующей таблицей с теми же видами связей. Для связи ∞:∞ все реквизиты двух объектов представляются тремя таблицами с двумя связями

вида 1:∞.Для каждой связи определяется главная таблица (на стороне отношения 1) и подчиненная (на стороне отношения ∞). В главной таблице связующее поле является первичным ключом (уникальным), а в подчиненной внешним (повторяющимся). Для каждого значения внешнего ключа обязательно должно быть такое же значение первичного ключа.

Создадим базу данных «Библиотека», содержащую следующие данные: информацию о книгах, об авторах, об издательствах, о читателях, о выдачах книг читателям и о местах хранения книг в библиотеке.

Ниже на рисунке 1.1 приведена схема данных базы данных «Библиотека». Схема данных выполнена в MSAccess.

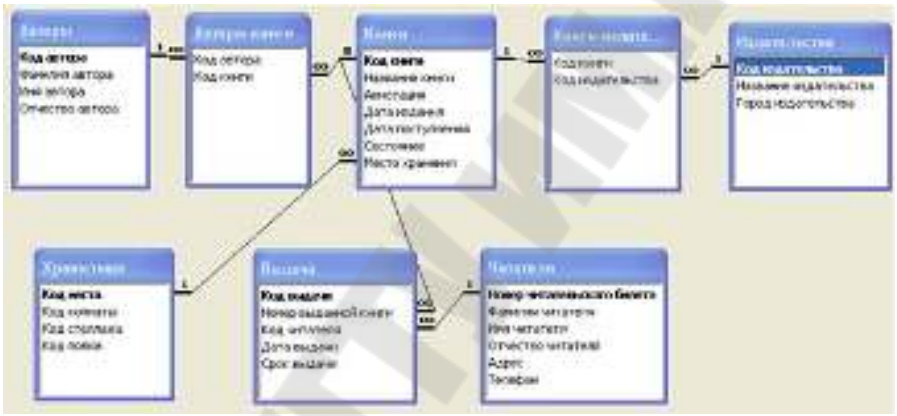


Рисунок 1.1– Схема данных базы данных «Библиотека»

Основная цель проектирования баз данных – это сокращение избыточности хранимых данных, а следовательно, экономия объема используемой памяти, уменьшение затрат на многократные операции обновления избыточных копий и устранение возможности возникновения противоречий из-за хранения в разных местах сведений об одном и том же объекте.

Так называемый, «чистый» проект базы данных («Каждый факт в одном месте») можно создать, используя методологию нормализации отношений. Нормализация должна использоваться на завершающей проверочной стадии проектирования базы данных.

Плохая проработка структуры базы почти всегда приводит к бесполезным затратам времени на ее переработку в дальнейшем. Опытные разработчики уделяют проектированию базы данных не меньше времени, чем их созданию.

Существует 7 основных этапов разработки базы данных:

1. определение назначения базы данных;
2. принятие решения о том, какие исходные данные база данных должна содержать;
3. определение исходных таблиц базы данных;
4. определение полей, которые будут входить в таблицы, и выбор полей, содержащих уникальные значения;
5. назначение связей между таблицами и окончательный просмотр получившейся структуры;
6. создание таблиц, связывание их между собой и экспериментальное наполнение базы пробными данными;
7. создание форм, отчетов и запросов для операций с введенными данными.

Разработка каждой базы данных начинается с изучения проблемы, которую она должна разрешить, или потребности, которую она должна удовлетворить. База данных «Библиотека» предназначена для хранения данных о приобретенных библиотекой книг, информации о местонахождении отдельных экземпляров каждого издания и сведений о читателях.

Для ведения библиотечных каталогов, организации поиска требуемых книг и библиотечной статистики в базе должны храниться сведения, большая часть которых размещаются в аннотированных каталожных карточках. Анализ запросов на литературу показывает, что для поиска подходящих книг (по тематике, автору, издательству и т.п.) и отбора нужного (например, по аннотации) следует выделить следующие атрибуты.

Атрибуты каталожной карточки: Автор (фамилия и имя каждого автора книги); Название книги; Место издания (город); Издательство (название издательства); Год выпуска; Аннотация.

Атрибуты места хранения отдельных экземпляров книг: Номер комнаты (помещения для хранения книг); Номер стеллажа в комнате; Номер полки на стеллаже; Номер (инвентарный номер книги); Дата приобретения; Дата размещения конкретной книги на конкретном месте; Дата изъятия книги с установленного места.

Атрибуты читателей: Номер читательского билета (формуляра); Фамилия читателя; Имя читателя; Отчество читателя; Адрес читателя; Телефон читателя; Дата выдачи читателю конкретной книги; Срок, на который конкретная книга выдана читателю; Дата возврата книги.

Анализ определенных выше объектов и атрибутов позволяет определить для проектируемой базы данных следующие таблицы:

1. Авторы – таблица предназначена для хранения сведений об авторах;

2. Книги – таблица предназначена для хранения сведений о книгах;

3. Издательства – таблица предназначена для хранения сведений об издательствах;

4. Хранилище – таблица предназначена для описания места хранения книг;

5. Выдача – таблица предназначена для хранения сведений о выданных книгах;

6. Читатели – таблица предназначена для хранения сведений о читателях библиотеки.

Определив набор таблиц, входящих в базу, надо продумать, какая информация о каждом объекте будет входить в каждую из таблиц. Каждое поле должно принадлежать одной отдельной таблице. В то же время информация в каждом поле должна быть структурно-элементарной, то есть она должна храниться в полях в виде наименьших логических компонентов.

Исходя из вышесказанного, определяем поля в выбранных таблицах и тип хранимых данных.

АВТОРЫ:

a. *код автора*. Поле счетчик, предназначено для однозначного определения каждого конкретного автора в базе данных;

b. *фамилия автора*. Символьное поле, не более 50 символов;

c. *имя автора*. Символьное поле, не более 25 символов;

d. *отчество автора*. Символьное поле, не более 25 символов.

КНИГИ:

a. *код книги*. Поле счетчик, предназначено для однозначного определения каждой конкретной книги в базе данных;

b. *название книги*. Символьное поле, не более 256 символов;

c. *аннотация*. Текстовое поле;

d. *дата издания.* Поле с типом дата;

e. *дата поступления в библиотеку.* Поле с типом дата;

f. *место хранения;* числовое поле.

ИЗДАТЕЛЬСТВА:

a. *код издательства*. Поле счетчик, предназначено для однозначного определения каждого конкретного издательства в базе данных;

b. *название издательства*. Символьное поле, не более 256 символов;

c. *город*, где расположено издательство. Символьное поле, не более 25 символов.

ХРАНИЛИЩЕ:

a. *код места*. Поле счетчик, предназначено для однозначного определения каждой конкретной полки в базе данных;

b. *номер комнаты*. Числовое поле;

c. *номер стеллажа*. Числовое поле;

d. *номер полки*. Числовое поле.

ВЫДАЧА:

a. *код выдачи*. Поле счетчик, предназначено для однозначного определения каждой конкретной выдачи в базе данных;

b. *номер выданной книги*. Числовое поле;

c. *код читателя*. Числовое поле;

d. *дата выдачи.* Поле с типом дата;

e. *срок выдачи* (количество дней). Числовое поле;

f. *дата возврата.* Поле с типом дата.

ЧИТАТЕЛИ:

a. *номер читательского билета.* Поле счетчик, предназначено для однозначного определения каждого конкретного читателя в базе данных;

b. *фамилия*. Символьное поле, не более 50 символов;

c. *имя*. Символьное поле, не более 50 символов;

d. *отчество*. Символьное поле, не более 50 символов;

e. *адрес*. Символьное поле, не более 256 символов;

f. *телефон*. Символьное поле, не более 20 символов.

Связь между таблицами устанавливается с помощью уникальных полей (первичных ключей). Для нашей база данных первичными ключами являются следующие поля: АВТОРЫ - код автора; КНИГИ - код книги; ИЗДАТЕЛЬСТВА - код издательства; ХРАНИЛИЩЕ - код места; ВЫДАЧА - код выдачи; ЧИТАТЕЛИ - номер билета.

Для того чтобы связать одну таблицу с другой, надо ввести во вторую таблицу поле первичного ключа из первой таблицы, т.е. ввести во вторую таблицу внешний ключ.

Связь двух таблиц выполняется подключением первичного ключа главной таблицы (находящейся на стороне отношения «один») к такому же полю внешнего ключа связанной таблицы (находящейся на стороне отношения «многие»).

Поле внешнего ключа в связанной таблице должно иметь тот же тип данных, что и первичный ключ в родительской таблице, но с одним исключением. Если первичный ключ главной таблицы имеет тип данных «Счетчик», то поле внешнего ключа в связанной таблице должно иметь тип данных «Числовой».

В нашей базе данных установим следующие типы связей между таблицами:

АВТОРЫ – КНИГИ. Здесь связь многие-ко-многим, у любого автора может быть более одной книги, и любая книга может быть написана несколькими авторами. Поэтому вводим вспомогательную таблицу «АВТОРЫ-КНИГИ» со следующими полями:

a. код автора. Числовой тип данных;
b. код книги. Числовой тип данных.

КНИГИ - ИЗДАТЕЛЬСТВА. Здесь связь многие-ко-многим, любая книга может быть издана несколькими издательствами и любое издательство издает не одну книгу. Поэтому вводим еще одну вспомогательную таблицу «КНИГИ-ИЗДАТЕЛЬСТВА» со следующими полями:

a. код книги. Числовой тип данных;
b. код издательства. Числовой тип данных.

ХРАНИЛИЩЕ - КНИГИ. Здесь связь один-ко-многим, на одной полке можно расставить множество книг, но любая книга может быть только на одной полке в хранилище. Поэтому поле «Место хранения» в таблице «Книги» определяем как внешний ключ, и связываем таблицы «Хранилище» и «Книги» первичным ключом «Код места» и внешним ключом «Место хранения».

КНИГИ - ВЫДАЧА. Здесь связь один-ко-многим, т.е. одна и та же книга может быть выдана несколько раз в разные даты на разные. Поэтому поле «Номер выданной книги» в таблице «Выдача» определяем как внешний ключ, и связываем таблицы «Книги» и «Выдача» первичным ключом «Код книги» и внешним ключом «Номер выданной книги».

ЧИТАТЕЛИ - ВЫДАЧА. Здесь связь один-ко-многим, т.е. одна и та же книга может быть выдана несколько раз разным читателям в разные сроки. Поэтому поле «Код читателя» в таблице «Выдача» оп-

ределяем как внешний ключ, и связываем таблицы «Читатели» и «Выдача» первичным ключом «Номер читательского билета» и внешним ключом «Код читателя».

Закончив проектирование таблиц и выявив связи, существующие между ними, необходимо тщательно перепроверить полученную структуру, прежде чем приступать к созданию таблиц и вводу информации. Нормализация отношений позволяет существенно сократить объем хранимой информации и устранить аномалии в организации хранения данных.

Правило 1: каждое поле таблицы должно представлять уникальный тип информации. В спроектированной нами базе данных нет полей в разных таблицах, содержащих одну и ту же информацию (за исключением внешних ключей).

Правило 2: каждая таблица должна иметь уникальный идентификатор, или первичный ключ, который может состоять из одного или нескольких полей. В спроектированной нами базе данных все таблицы (за исключением вспомогательных «АВТОРЫ-КНИГИ» и «ИЗДАТЕЛЬСТВА-КНИГИ») содержат первичный ключ.

Правило 3: для каждого значения первичного ключа значения в столбцах данных должны относиться к объекту таблицы и полностью его описывать. Это правило используется двояко. Во-первых, в таблице не должно быть данных, не относящихся к объекту, определяемому первичным ключом. Например, хотя для каждой книги требуется информация о ее авторе, но автор является самостоятельным объектом, и данные о нем должны находиться в соответствующей таблице. Во-вторых, данные в таблице должны полностью описывать объект.

Правило 4: должна быть возможность изменять значения любого поля (не входящего в первичный ключ) без воздействия на данные других полей. Последнее правило позволяет проверить, не возникнут ли проблемы при изменении данных в таблицах. Поскольку в спроектированной нами базе данные, содержащиеся в разных полях таблиц, нигде не повторяются, мы имеем возможность корректировать значения любых полей (за исключением первичных ключей).

Чтобы определить, насколько структура база данных соответствует поставленной задаче и насколько удобно с этой базой работать, необходимо ввести несколько простейших записей.

2 Введение в MySQL

2.1 Основные сведения о MySQL

MySQL - это быстрая, надежная, открыто распространяемая *СУБД*.

СУБД MySQL функционирует по модели «клиент/сервер». Под этим подразумевается сетевая архитектура, в которой компьютеры играют роли клиентов либо серверов. На рисунке 1.2 изображена схема передачи информации между компьютером клиента и жестким диском сервера.

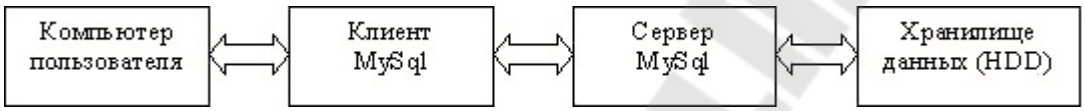


Рисунок 1.2 – Схема передачи данных в архитектуре «клиент/сервер»

Клиентская программа MySQL представляет собой утилиту командной строки. Работать с MySQL можно тремя основными способами: используя командную строку, используя веб-интерфейс наподобие phpMyAdmin и используя такой язык программирования, как PHP. Первые два способа будут рассмотрены в этом пособии.

MySQL взаимодействует с базой данных на языке, называемом *SQL* (Structured Query Language – язык структурированных запросов).SQL предназначен для манипуляции данными, которые хранятся в системах управления реляционными базами данных (СУРБД). SQL имеет команды, с помощью которых данные можно извлекать, сортировать, обновлять, удалять и добавлять. Язык структурированных запросов можно использовать с такими СУРБД как MySQL, mSQL, PostgreSQL, Oracle, Microsoft SQL Server, Access, Sybase, Ingres. Эти системы СУРБД поддерживают все важные и общепринятые операторы SQL, однако каждая из них имеет множество своих собственных патентованных операторов и расширений [2].

MySQL имеет API для языков *Delphi, C, C++, Эйфель, Java, Лисп, Perl, PHP, PureBasic, Python, Ruby, Smalltalk, Компонентный Паскаль и Tcl* библиотеки для языков платформы .NET, а также обеспечивает поддержку для ODBC посредством ODBC-драйвера MyODBC.

Пользователю базы данных необязательно знать, как инсталлировать MySQL. В крупных организациях есть системные администраторы, которые этим занимаются. Что касается разработчиков, то им нужно понимать особенности данного процесса. Именно здесь у них появляется доступ к различным конфигурационным установкам, с помощью которых можно настроить производительность программы. Естественно, необходимо обладать правами администратора на том компьютере, где MySQL устанавливается в виде сервиса, запускаемого автоматически. Скачать дистрибутив можно с сайта разработчиков http://www.mysql.com/.

Установка и настройка сервера MySQL для различных операционных систем есть в книгах Л. Аткинсона [3], Р. Никсона [4].

2.2 Основы работы с MySQL из командной строки

Нет никакой разницы, из какой именно операционной системы вы получаете доступ к MySQL, поскольку все используемые команды абсолютно одинаковы.

Для завершения команды или отделения их друг от друга используется символ точка с запятой «;». Если забыть поставить этот символ, MySQL выдаст приглашение и будет ожидать от вас ввода команды. Запрашиваемая тоска с запятой стала частью синтаксиса, позволяющего вводить длинные команды, разбивая их на несколько строк. Она также позволяет вводить сразу несколько команд, после каждой из которых стоит точка с запятой. После нажатия клавиши Enter интерпретатор получает все эти команды в едином пакете и выполнит их в порядке следования.

На экране могут появляться шесть различных приглашений MySQL (таблица 2.1), позволяющих определить, на каком именно этапе многострочного ввода вы находитесь.

Таблица 2.1 – Виды приглашений MySQL

Приглашение MySQL	**Значение**
mysql>	MySQL готова к работе и ждет ввода команды
->	Ожидание следующей строки команды
‘>	Ожидание следующей строки строкового значения, которое началось с одинарной кавычки
“>	Ожидание следующей строки строкового значения, которое началось с двойной кавычки

Таблица 2.1 (продолжение)

`>	Ожидание следующей строки строкового значения, которое началось с символа засечки (`)
/*>	Ожидание следующей строки комментария, который начался с символов /*

Чтобы удалить набранную часть программы, следует ввести команду \c и нажать клавишу Enter. Комбинация \c после точки с запятой работать не будет.

В таблице 2.2 приведена подборка наиболее востребованных команд MySQL, а также кратких форм некоторых команд.

Таблица 2.2 – Некоторые команды MySQL

Команда	**Параметр (-ы)**	**Назначение**
ALTER	База данных, таблица	Внесение изменений в базу данных или таблицу
BACKUP	Таблица	Создание резервной копии таблицы
\c		Отмена ввода
CREATE	База данных, таблица	Создание базы данных или таблицы
DELETE	(Выражение с участием таблицы и строки)	Удаление строки из таблицы
DESCRIBE	Таблица	Описание столбцов таблицы
DROP	База данных, таблица	Удаление база данных или таблицы
EXIT(CTRL-C)		Выход
GRANT	(Пользователь подробности)	Изменение привилегий пользователя
HELP (\h, \?)	Элемент	Отображение подсказки по элементу
INSERT	(Выражение с данными)	Вставка данных
LOCK	Таблица (таблицы)	Блокировка таблицы (таблиц)
QUIT (\q)		Тоже что и EXIT
RENAME	Таблица	Переименование таблицы
SHOW	(Множество элементов для вывода в виде списка)	Список сведений об элементах
SOURCE	Имя_файла	Выполнение команд из файла имя_файла

Таблица 2.2 (продолжение)

STATUS (\s)		Отображение текущего состояния
TRUNCATE	Таблица	Опустошение таблицы
UNLOCK	Таблица (таблицы)	Снятие блокировки таблицы (таблиц)
UPDATE	(Выражение с данными)	Обновление существующей записи
USE	База_данных	Использование базы_данных

Многие из этих команд будут рассмотрены по мере изучения. Далее приведем два положения, касающихся команд MySQL:

1. Команды и ключевые слова SQL нечувствительны к регистру. Но чтобы было понятнее, для команд рекомендуется использовать буквы верхнего регистра;
2. Имена таблиц чувствительны к регистру в Linux и MacOSX, но нечувствительны в Windows. Для имен таблиц рекомендуется использовать буквы нижнего регистра.

3 Создание учебной базы данных

В этом разделе создадим ранее рассматриваемую базу данных библиотеки художественной литературы «*Библиотека*». База данных предназначена для хранения данных о приобретенных библиотекой книг, информации о местонахождении отдельных экземпляров каждого издания и сведений о читателях.

3.1 Создание базы данных

Для создания базы данных в MySQL используется команда ***create database***.

Синтаксис команды:

```
create database [if not exists]имя_базы_данных
```

Данная команда создает базу данных с указанным именем. Если база данных с таким именем существует, генерируется ошибка. Чтобы иметь возможность пользоваться этой командой необходимо иметь привилегию create для базы данных. О привилегиях мы поговорим позже.

Шаги создания базы данных в Windows.

1. Запустите сервер MySQL, выполняя следующую команду в командной строке:

```
start mysql;
```

2. Затем запустите программу клиента mysql, введите в строке приглашения пароль (если вы его задавали при установке).

3. Появится приглашение: mysql>. Далее введите команду:

```
create database biblioteca;
```

Примечание: Обратите внимание, что все команды набираются в одну строку и заканчивается символом точка с запятой.

4. Сервер MySQL должен ответить примерно так:

```
Query OK, 1 row affected (0.05 sec)
```

Эта запись означает, что «запрос обработан, изменилась одна строка (0.00 сек)». Таким образом, база данных создана.

5. Чтобы просмотреть имеющиеся базы данных в системе нужно выполнить следующую команду:

```
show databases;
```

Сервер MySQL покажет список имеющихся баз данных, рисунок 3.1.

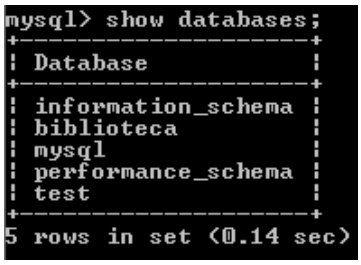


Рисунок 3.1 – Просмотр баз данных

Как видно на рис. 3.1 показаны пять баз данных: четыре базы данных сервер MySQL создал во время установки, а пятая база данных *biblioteca* была создана нами.

6. Чтобы закончить работу с сервером MySQL нужно ввести команду quit; или exit; в приглашении mysql. Затем в командной строке прописать команду

```
stop mysql;
```

Таким образом, работа сервера MySQL будет остановлена.

Базы данных в MySQL реализованы в виде каталогов, которые содержат файлы, соответствующие таблицам базы данных. Поскольку изначально в базе нет никаких таблиц, оператор ***create database*** только создает подкаталог в каталоге данных MySQL.

База данных *biblioteca* уже создана. Для работы с ней, необходимо её «активировать» или «выбрать». В приглашении mysql выполните команду:

```
select database();
```

На экране увидим ответ системы, как показано на рисунке 3.2.

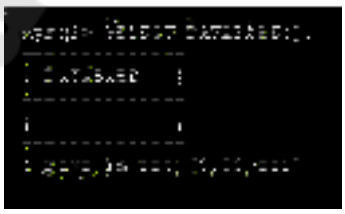


Рисунок 3.2 – Выбор базы данных

Это говорит о том, что ни одна база данных не была выбрана. На самом деле всякий раз при работе с клиентом mysql необходимо определять, какая база данных будет использоваться.

Определить текущую базу данных можно несколькими способами (в Windows):

– определение имени базы данных при запуске. Для этого введите в приглашении системы следующее:

```
mysql biblioteca
```

– определение базы данных с помощью оператора USE в приглашении mysql:

```
mysql>USE biblioteca;
```

– определение базы данных с помощью \u в приглашении mysql:

```
mysql>\u biblioteca;
```

При работе необходимо определять базу данных, которая будет использоваться, иначе MySQL будет порождать ошибку.

3.2 Создание первой таблицы

Теперь рассмотрим команды MySQL для создания таблиц базы данных.

При создании таблиц, будем придерживаться структуры, описанной в разделе 1. В таблице 3.2 приведены данные некоторых книг.

Таблица 3.2 – Данные о книгах

Код книги	Название книги	Аннотация	Дата поступления	Дата издания	Состояние	Вес, гр.	Страницы, число	Место хранения
1	Волшебник изумрудного города	Эта книга о…	28.04.2001	21.01.2000	Хорошее	100	55	5
2	Сонная сказка	Эта книга о…	24.09.2008	25.03.2006	Хорошее	100	583	4
3	43 медвежки	Эта книга о…	03.04.1999	09.03.1999	Хорошее	240	739	6
4	Ежик	Эта книга о…	09.12.2003	26.05.2000	Хорошее	159	875	7
5	Страна мудрецов	Эта книга о…	09.05.2002	05.01.2000	Плохое	320	2095	5
…	…	…	…	…	…	…	…	…

Приведенная выше таблица содержит девять столбцов. Каждая строка содержит данные об одной книге. Чтобы найти дату поступления и место хранения книги «Ежик», сначала надо выбрать Название_книги во втором столбце, а затем посмотреть содержимое четвертого и девятого столбцов этой же строки.

База данных может содержать множество таблиц. Давайте создадим первую таблицу в MySQL.После выбора базы данных biblioteca (команда use), выполните в приглашении mysql команду ***create table***.Синтаксис этой команды выглядит следующим образом:

```
create table books_data
(
book_idint unsigned not null auto_increment primary key,
book_title varchar(50),
annotation varchar(120),
r_date date,
p_date date,
state ENUM('perfect', 'good', 'bad'),
weight int,
pages int,
storage int unsigned not null
);
```

Нажатие клавиши Enter после ввода первой строки изменяет приглашение mysql на ->. Это означает, что mysql понимает, что команда не завершена и приглашает ввести дополнительные операторы. Помните, что каждая команда mysql заканчивается точкой с запятой, а каждое объявление столбца отделяется запятой. Можно также при желании ввести всю команду на одной строке.

Теперь разберем подробнее эту команду. За ключевыми словами ***create table*** следует имя создаваемой таблицы books_data. Каждая строка внутри скобок представляет один столбец. Эти столбцы хранят для каждой книги идентификационный номер (book_id), название книги (book_title), аннотацию (annotation), дату поступления (r_data), дату публикации (p_data), состояние(status), вес (weight), страницы (pages)и место хранения (storage).

За именем каждого столбца следует тип столбца. Типы столбцов определяют тип данных, которые будет содержать столбец. В данном примере столбцы book_title, annotaciya будут содержать текстовые строки, поэтому тип столбца задан как varchar, что означает переменное количество символов. Максимальное число символов для столбцов varchar определяется числом, заключенным в скобки, которое следует сразу за именем столбца. Первый столбец (book_id) содержит идентификационный номер (id) книги. Рассмотрим тип этого столбца по частям.

Таблица 3.2 –Тип столбца book_id

Тип	Описание
int	определяет тип столбца как целое число
unsigned	означает, что число будет без знака (положительное целое)
notnull	определяет, что значение не может быть null (пустым)
auto_increment	генерируется новое значение, которое на единицу больше, чем наибольшее значение в столбце. Не нужно задавать для этого столбца значения. Каждое значение в этом столбце будет уникальным
primarykey	помогает при индексировании столбца, что ускоряет поиск значений. Каждое значение должно быть уникально. Ключевой столбец необходим для того, чтобы исключить возможность совпадения данных. Например, две книги могут иметь одно и то же название, и тогда встанет проблема – как различать эти книги, если не задать им уникальные идентификационные номера. Если имеется столбец с уникальными значениями, то можно легко различить две записи. Лучше всего поручить присваивание уникальных значений самой системе MySQL

Если вы набрали все правильно, то вывод на экране должен соответствовать рисунку 3.3.

Рисунок 3.3 – Создание таблицы

Общий синтаксис команды ***create table*** следующий:

```
create [temporary] table [if not exists] имя
[(спецификация, ...)]
[опция, ...]
[[ignore | replace] запрос]
```

Флаг *temporary* задает создание временной таблицы, существующей в течение текущего сеанса. По завершении сеанса таблица удаляется. Временным таблицам можно присваивать имена других

таблиц, делая последние временно недоступными. Спецификатор *if not exists* подавляет вывод сообщений об ошибках в случае, если таблица с указанным именем уже существует. Имени таблицы может предшествовать имя базы данных, отделенное точкой. Если это не сделано, таблица будет создана в базе данных, которая установлена по умолчанию.

Чтобы задать имя таблицы с пробелами, необходимо заключить его в обратные кавычки, например 'courseslist'. То же самое нужно будет делать во всех ссылках на таблицу, поскольку пробелы используются для разделения идентификаторов.

Разрешается создавать таблицы без столбцов, однако в большинстве случаев спецификация хотя бы одного столбца все же присутствует. Спецификации столбцов и индексов приводятся в круглых скобках и разделяются запятыми.

Формат спецификации следующий:

```
имя тип
[not null | null]
[default значение]
[auto_increment]
[key]
[ссылка]
```

Типы столбцов более подробно будут рассмотрены в следующем разделе.

Спецификация типа включает название типа и его размерность. По умолчанию столбцы принимают значения *null*. Спецификатор *not null* запрещает подобное поведение.

У любого столбца есть значение по умолчанию. Если оно не указано, программа MySQL выберет его самостоятельно. Для столбцов, принимающих значения *null*, значением по умолчанию будет *null*, для строковых столбцов – пустая строка, для численных столбцов – нуль. Изменить эту установку позволяет предложение *default*.

Поля-счетчики, создаваемые с помощью флага auto_increment, игнорируют значения по умолчанию, так как в них записываются порядковые номера. Тип счетчика должен быть беззнаковым целым. В таблице может присутствовать лишь одно поле-счетчик. Им не обязательно является первичный ключ.

3.3 Некоторые команды при работе с таблицами

Перед тем как изменить атрибуты таблиц, нужно не забыть сделать нужную базу данных текущей, используя оператор USE.

Для того, чтобы удалить таблицу, убедимся сперва, что она существует. Это можно проверить с помощью команды ***show tables;***, как показано на рисунке 3.4.

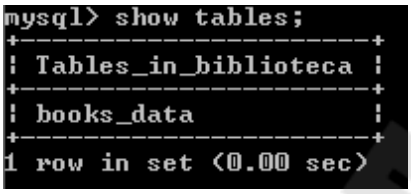


Рисунок 3.4 – Просмотр таблиц в базе

Для удаления таблицы используется команда ***drop table;***, как показано на рисунке 3.5.

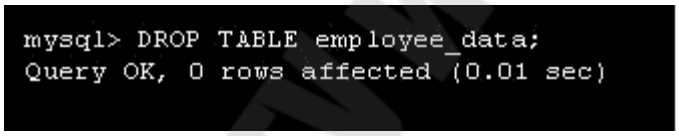


Рисунок 3.5 – Удаление таблицы

Теперь команда ***show tables;*** этой таблицы больше не покажет.

Синтаксис команды ***drop table***:

drop table `[if exists] таблица [restrict|cascade]`

Спецификация *if exists* подавляет вывод сообщения об ошибке, выдаваемого в случае, если заданная таблица не существует. Можно указывать несколько имен таблиц, разделяя их запятыми.

Флаги *restrict* и *cascade* предназначены для выполнения сценариев, созданных в других СУБД.

В таблице 3.3 приведен перечень некоторых команд, также полезных при работе с таблицами, столбцами и ячейками таблиц.

Таблица 3.3 – Дополнительные команды

Синтаксис	Описание
ALTER TABLE название таблицы RENAME новое название;	переименовать имя таблицы
ALTER TABLE название таблицы CHANGE старое название ячейки новое название ячейки VARCHAR(50);	переименовать ячейку таблицы

Таблица 3.3 (продолжение)

ALTER TABLE название таблицы ADD название столбца TIMESTAMP;	добавить столбец
ALTER TABLE название таблицы MODIFY название колонки VARCHAR(20);	изменить значение
ALTER TABLE название таблицы DROP COLUMN название столбца;	удалить столбец

4 Типы данных

MySQL поддерживает три основных типа данных столбцов: числовые типы, типы даты и времени, символьные (строковые) типы.

4.1 Числовые типы данных

Числовые типы данных столбцов используются для хранения чисел. Типы INTEGER (сокр. INT) (целое число) и FLOAT(число с плавающей точкой) являются представителями точных числовых типов и приближенных числовых типов соответственно. Числовые типы могут характеризоваться максимальной длинной, а типы с плавающей точкой – числом десятичных разрядов. Эти значения указываются сразу после объявления типа, например: summa decimal(10,2). Здесь указаны длина 10 и два знака после десятичного разделителя.

Объявления числовых типов можно также завершать ключевым словом UNSIGNED, это означает, что столбец содержит только положительные числа или нули.

Типы данных DECIMAL (DEC) иNUMERIC идентичны. Эти типы используются для хранения точных значений с плавающей запятой (например, для хранения в столбцах денежных значений). Они имеют тот же диапазон, что и числа с плавающей запятой двойной точности (типDOUBLE).

Тип INTEGERили сокращенно INTзанимает 4 байта, с диапазоном из 2^{32} возможных значений. У типа INTсуществует несколько вариаций:

- TINYINT–занимает 1 байт (2^8 возможных значений). Синонимами являются типы BIT и BOOT.
- SMALLINT – занимает 2 байта (2^{16} возможных значений).
- MEDIUMINT– занимает 3 байта (2^{24} возможных значений).
- BIGINT– занимает 8 байта (2^{64} возможных значений).

Тип FLOAT – это числа с плавающей запятой с обычной точностью. Они могут представлять положительные числа в диапазоне от $1{,}18 \times 10^{-38}$ до $3{,}40 \times 10^{38}$и аналогичный диапазон отрицательных чисел.

Тип DOUBLE– это числа с плавающей запятой с двойной точностью. Синонимами типа DOUBLE являются REAL и DOUBLE PRECISION. Они могут представлять положительные числа в диапазоне от $2{,}23 \times 10^{-308}$ до $1{,}80 \times 10^{308}$ и аналогичный диапазон отрицательных чисел.

4.2 Типы данных даты и времени

Существуют следующие типы данных даты и времени: DATETIME, DATE, TIMESTAMP, TIME и YEAR. Каждый из них имеет интервал допустимых значений, а также значение «ноль», которое используется, когда пользователь вводит действительно недопустимое значение.

MySQL пытается интерпретировать значения даты и времени в нескольких форматах. Однако, во всех случаях ожидается, что крайним слева будет раздел значения даты, содержащий год. Даты должны задаваться в порядке год-месяц-день (например, '98-09-04'), а не в порядке месяц-день-год или день-месяц-год, т.е. не так, как мы их обычно записываем.

MySQL автоматически преобразует значение, имеющее тип даты или времени, в число, если данная величина используется в числовом контексте, и наоборот.

Тип DATE предназначен для хранения дат. Ожидается хранение даты в виде ГГГГ-ММ-ДД.

Тип TIME предназначен для хранения значений времени, отображаемых в виде ЧЧ:ММ:СС.

Тип DATETIME представляет собой комбинацию двух предыдущих типов в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.

TIMESTAMP– очень полезный тип столбца. Если в соответствующем столбце строки вы укажете конкретное значение или NULL, там будет записано время, когда соответствующая строка была создана или в последний раз изменена. Извлекаемое значение TIMESTAMP будет отображаться в формате DATETIME.

Тип YEAR предназначен для хранения значений года. При объявлении столбца этого типа можно объявить его как YEAR(2) или YEAR(4), чтобы узнать число знаков. MySQL извлекает и выводит величины YEAR в формате ГГГГ. Диапазон возможных значений - от 1901 до 2155. Недопустимые величины YEAR преобразуются в 0000.

4.3 Символьные типы данных

Существуют следующие символьные типы данных: CHAR, VARCHAR, BLOB, TEXT, ENUM и SET. Рассмотрим каждый из этих типов по очереди.

Типы данных CHAR и VARCHARочень схожи между собой, но различаются по способам их хранения и извлечения.

Тип данных CHAR.

Длина поля постоянна и задается при создании таблицы. Эта длина может принимать любое значение между 1 и 255. Величины типа CHAR при хранении дополняются справа пробелами до заданной длины. Эти концевые пробелы удаляются при извлечении хранимых величин.

Тип данных VARCHAR.

Величины в столбцах данного типа представляют собой строки переменной длины. Так же как и для столбцов CHAR, можно задать столбец VARCHAR любой длины между 1 и 255. Однако, в противоположность CHAR, при хранении величин типа VARCHAR используется только то количество символов, которое необходимо, плюс один байт для записи длины. Хранимые величины пробелами не дополняются, наоборот, концевые пробелы при хранении удаляются.

Перед CHARи VARCHARможно указать ключевое слово NATIONAL, чтобы ограничить содержимое столбца соответствующим стандартным набором символов. В MySQLоно используется по умолчанию, так что вам это ключевое слово может потребоваться для кроссплатформенной совместимости.

После CHARи VARCHARможно добавить ключевое слово BINARY, означающее необходимость учета регистра символов при сравнении строк. По умолчанию при сравнении строк регистр символов игнорируется.

Тип данных BLOB

Представляет собой двоичный объект большого размера, который может содержать переменное количество данных. Существуют 4 модификации этого типа - TINYBLOB, BLOB, MEDIUMBLOB и LONGBLOB, отличающиеся только максимальной длиной хранимых величин.

Тип данных TEXT

Имеет 4 модификации - TINYTEXT, TEXT, MEDIUMTEXT и LONGTEXT, соответствующие упомянутым четырем типам BLOB и имеющие те же максимальную длину и требования к объему памяти. Единственное различие между типами BLOB и TEXT состоит в том, что сортировка и сравнение данных выполняются с учетом регистра для величин BLOB и без учета регистра для величин TEXT.

Тип перечисления ENUM

Этот тип может принимать значение из списка допустимых значений, явно перечисленных в спецификации столбца в момент создания таблицы.

Перечисление может иметь максимум 65535 элементов.

Регистр не играет роли, когда вы делаете вставку в столбец ENUM. Однако регистр значений, получаемых из этого столбца, совпадает с регистром в написании соответствующего значения, заданного во время создания таблицы.

Тип множества SET

Тип SET– строковый, может принимать ноль или более значений, каждое из которых должно быть выбрано из списка допустимых значений, определенных при создании таблицы. Элементы множества SET разделяются запятыми. Как следствие, сами элементы множества не могут содержать запятых. Множество SET может иметь максимум 64 различных элементов. Если вы вставляете в столбец SET некорректную величину, это значение будет проигнорировано.

5 Работа с таблицами

5.1 Запись данных в таблицы

Оператор INSERT заполняет таблицу данными. Синтаксис:

```
INSERT into table_name (column1, column2, ...)
values (value1, value2...);
```

где table_name является именем таблицы, в которую надо внести данные; column1, column2 и т.д. являются именами столбцов, а value1, value2 и т.д. являются значениями для соответствующих столбцов.

Следующий оператор вносит первую *запись в таблицу* books_data, которую мы рассматривали в разделе 3.

```
INSERT into books_data
(book_title, annotation, r_date, p_date,
state, weight, pages, storage)
values
('Волшебник изумрудного города', 'Эта книга
о…', 28.04.2001, 21.01.2000,'good', 100, 55, 5);
```

Как и другие операторы MySQL, эту команду можно вводить на одной строке или разместить ее на нескольких строках.

Значениями для столбцов book_title, annotation, state являются текстовые строки, и они записываются в кавычках. Значением для r_date, p_date являются тип дата, и оно не имеет кавычек. Можно видеть, что данные заданы для всех столбцов кроме book_id. Значение для этого столбца задает система MySQL, которая находит в столбце наибольшее значение, увеличивает его на единицу, и вставляет новое значение.

Если приведенная выше команда правильно введена в приглашении клиента mysql, то программа выведет сообщение об успешном выполнении, как показано на рисунке5.1.

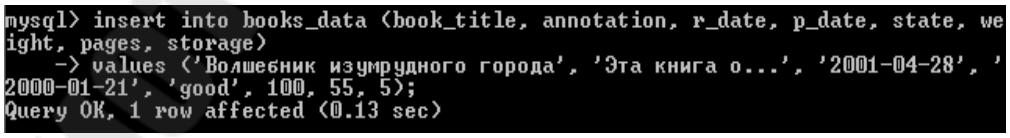


Рисунок 5.1 – Ввод данных в таблицу

Создание дополнительных записей требует использования отдельных операторов INSERT. Чтобы облегчить эту работу можно поместить все операторы INSERT в файл. Это должен быть обычный текстовый файл с оператором INSERT в каждой строке.

Заполнение таблицы books_data данными с помощью файла biblioteca.dat

В системе Windows

1) Поместите файл в каталог c:\mysql\bin.

2) Проверьте, что MySQL работает.

3) Выполните команду

mysqlbooks_data < biblioteca.dat

Пусть таблица содержит теперь 10 запись (9 из файла biblioteca.dat и одну, вставленную оператором INSERT в начале раздела).

5.2 Запрос данных из таблицы

Таблица books_data содержит теперь достаточно данных, чтобы можно было начать с ней работать. *Запрос данных* выполняется с помощью команды MySQL SELECT.

Оператор SELECT имеет следующий формат:

```
SELECT имена_столбцов from имя_таблицы
[WHERE ...условия];
```

Часть оператора с условиями является необязательной (мы рассмотрим ее позже). По сути, требуется знать имена столбцов и имя таблицы, из которой извлекаются данные.

Например, чтобы извлечь названия всех книг и их состояние, выполните следующую команду.

```
SELECT book_title, state from books_data;
```

Оператор приказывает MySQL вывести все данные из столбцов book_title и state. Результат работы оператора представлен на рисунке 5.2.

```
mysql> select book_title, state from books_data;
+-----------------------------+---------+
| book_title                  | state   |
+-----------------------------+---------+
| Волшебник изумрудного города | good    |
| Сонная сказка               | good    |
| 43 медвежки                 | good    |
| Ежик                        | good    |
| Страна мудрецов             | bad     |
| 12 месяцев                  | good    |
| Тень                        | bad     |
| Тень                        | perfect |
| Солнечный мальчик           | good    |
| Три толстяка                | perfect |
+-----------------------------+---------+
10 rows in set (0.02 sec)
```

Рисунок 5.2 – Вывод данных из таблицы

При ближайшем рассмотрении можно заметить, что данные представлены в том порядке, в котором они были введены. Более того, последняя строка указывает число строк в таблице - 10.

Чтобы вывести всю таблицу, можно либо ввести имена всех столбцов, либо воспользоваться упрощенной формой оператора SELECT.

```
SELECT * from books_data;
```

Символ * в этом выражении означает «все столбцы». Поэтому этот оператор выводит все строки всех столбцов.

5.3 Выборка данных с помощью условий

Теперь более подробно рассмотрим формат оператора SELECT. Его полный формат имеет вид:

```
SELECT имена_столбцов from имя_таблицы
[WHERE ...условия];
```

В операторе SELECT условия являются необязательными.

Оператор SELECT без условий выводит все данные из указанных столбцов.

Теперь перейдём к рассмотрению примеров с применением операторов сравнения, таких как =, !=, >, <, >=, <=.

Ниже приведен пример выборки названий и дат поступления в библиотеку тех книг, у которых отличное состояние.

```
SELECT book_title, r_date from books_data
Where state = 'perfect';
```

Результат запроса приведен на рисунке 5.3.

```
mysql> select book_title, r_date from books_data where state='perfect';
+--------------+------------+
| book_title   | r_date     |
+--------------+------------+
| Тень         | 2012-02-24 |
| Три толстяка | 2011-10-04 |
+--------------+------------+
2 rows in set (0.06 sec)
```

Рисунок 5.3 – Выборка столбцов с условием для поля «состояние»

Отметим, что *perfect*в условии заключено в одиночные кавычки. Можно использовать также двойные кавычки. Кавычки являются обязательными, так как MySQL будет порождать ошибку при их отсутствии. Кроме того сравнения MySQL не различают регистр символов,

что означает, что с равным успехом можно использовать "Perfect", "perfect" и даже "PeRfEct".

Рассмотрим другой пример:

```
SELECT book_title,weight from books_data
Where storage="1";
```

Результат запроса приведен на рисунке 5.4.

```
mysql> select book_title, weight from books_data where storage='1';
+--------------+--------+
| book_title   | weight |
+--------------+--------+
| 43 медвежки  |     15 |
| Ежик         |     15 |
| Три толстяка |    100 |
+--------------+--------+
3 rows in set (0.00 sec)
```

Рисунок 5.4 – Выборка столбцов с условием для поля "storage"

Данный запрос выбирает названия и вес всех книг, которые хранятся в заданном месте.

Оператор!= означает «не равно» и является противоположным оператору равенства.

Давайте получим названия книг, их состояния и место хранения, число страниц у которых меньше 20.

```
SELECT book_title, state, storage
from books_data where pages<20;
```

Результат запроса приведен на рисунке 5.5.

```
mysql> select book_title, state, storage from books_data where pages < 20;
+------------------+-------+---------+
| book_title       | state | storage |
+------------------+-------+---------+
| Сонная сказка    | good  |       2 |
| Солнечный мальчик | good  |       4 |
+------------------+-------+---------+
2 rows in set (0.00 sec)
```

Рисунок 5.5 – Выборка столбцов с условием «меньше» для поля «pages»

Используемые в основном с целочисленными данными операторы меньше или равно (<=) и больше или равно (>=) обеспечивают дополнительные возможности.

5.4 Поиск текстовых данных по шаблону

В данной части мы рассмотрим поиск текстовых данных по шаблону с помощью предложения where и оператора LIKE.

Допустим необходимо вывести данные о книгах, названия которых начинается с буквы С. Язык SQL позволяет выполнить поиск строковых данных по шаблону. Для этого в предложении where используется оператор LIKE следующим образом.

```
Select book_title, annotation from books_data
Where book_title LIKE "C%";
```

Результат запроса приведен на рисунке 5.6.

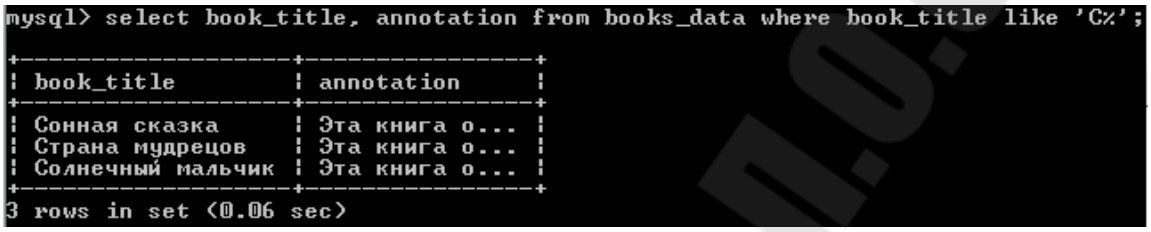


Рисунок 5.6 – Результат поиска книг на букву С

Можно видеть, что здесь в условии вместо знака равенства используется LIKE и знак процента в шаблоне.

Знак % действует как символ-заместитель. Он заменяет собой любую последовательность символов. Таким образом "С%" обозначает все строки, которые начинаются с буквы С. Аналогично "%С" выбирает строки, которые заканчиваются символом С, а "%В%" строки, которые содержат букву С.

Давайте выведем, например, все книги, которые имеют в названии строку «ма».

```
Select book_title, state from books_data
Where book_title like '%ма%';
```

Результат запроса приведен на рисунке 5.7.

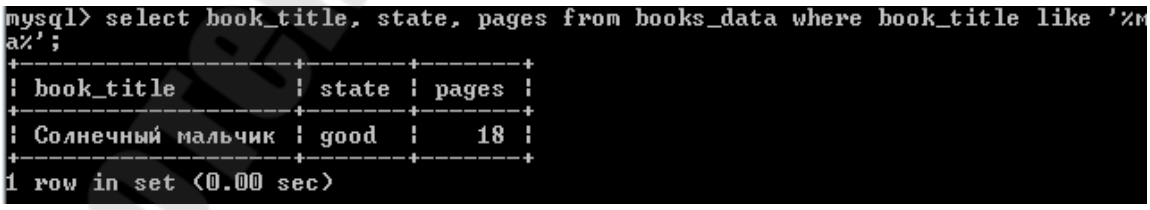


Рисунок 5.7 – Результат поиска книг, содержащих строку «ма»

5.5 Группировка и предложение HAVING

Чтобы вывести средний вес всех книг в различных подразделениях (место хранения), используется предложение GROUP BY, например:

```
select storage, AVG(weight)from books_data
```

```
GROUP BY storage;
```

Результат запроса приведен на рисунке 5.8.

```
mysql> select storage, avg(weight) from books_data group by storage;
+---------+-------------+
| storage | avg(weight) |
+---------+-------------+
|       1 |     43.3333 |
|       2 |     38.3333 |
|       4 |     45.0000 |
|       5 |    100.0000 |
|       6 |    100.0000 |
+---------+-------------+
5 rows in set (0.00 sec)
```

Рисунок 5.8 – Вывод среднего веса книг по месту хранения

Предположим теперь, что требуется вывести только те места хранения, где средний вес книг не превышает 45 грамм. Это можно сделать с помощью предложения HAVING.

```
Select storage, AVG(weight)
From books_data
GROUP BY storage
HAVING AVG(weight) <= 45;
```

Результат запроса приведен на рисунке 5.9.

```
mysql> select storage, avg(weight)
    -> from books_data
    -> group by storage
    -> having avg(weight) <= 45;
+---------+-------------+
| storage | avg(weight) |
+---------+-------------+
|       1 |     43.3333 |
|       2 |     38.3333 |
|       4 |     45.0000 |
+---------+-------------+
3 rows in set (0.00 sec)
```

Рисунок 5.9 – Вывод среднего веса определённого диапазона по месту хранения

5.6 Удаление записей из таблицы

Для удаления записей из таблицы можно использовать оператор DELETE.

Оператор удаления DELETE требует задания имени таблицы и необязательных условий. Его синтаксис следующий:

```
DELETE from имя_таблицы
[WHERE условия];
```

Если никакие условия не будут заданы, то удаляются *все данные* в таблице!

Предположим, нужно удалить книгу из таблицы, поступившую 1.02.2004.

```
DELETE from books_data
WHERE r_date='2004-02-01';
```

Результат запроса приведен на рисунке 5.10.

Рисунок 5.10 – Результат удаления записи из таблицы

Задания для самоконтроля

1. Напишите оператор SQL для создания новой базы данных с именем Avtory.

2. Какой оператор используется для получения информации о таблице? Как используется этот оператор?

3. Как получить список всех баз данных, доступных в системе?

4. Приведите две формы оператора SELECT, которые будут выводить все данные из таблицы books_data.

5. Как извлечь данные столбцов book_title, r_date из таблицы books_data?

6. Как узнать число строк в таблице с помощью оператора SELECT?

7. Напишите оператор SELECT для извлечения идентификационного номера книг, число страниц которых больше или равно 120.

8. Напишите оператор SELECT для извлечения информации о книге «Сонная сказка»

9. Что выведет следующий оператор SELECT:

```
SELECT * from books_data where weight <= 200;
```

10. Как вывести название книг, у которых состояние «bad».

11. Перечислить все книги, названия которых заканчивается на «ий».

12. Что выведет следующий оператор

```
SELECT book_title, r_date, p_date
from books_data
where book_title like '%ок%';
```

13. Вывести названия книг, их состояния и среднее количество страниц, где средний вес больше 30.

6 Логические операторы

6.1 Операторы AND, OR, NOT

В этом разделе рассмотрим, как выбрать данные на основе условий SQL, представленных с помощью *булевых (логических) операторов* – AND, OR, NOT.

Ниже показан оператор SELECT, который выводит названия книг, которые весят более 45 грамм, но менее150 грамм.

```
SELECT book_title from books_data
where weight>45 AND weight<150;
```

На рисунке 6.1 приведен результат этого запроса.

```
mysql> select book_title from books_data
    -> where weight > 45 and weight < 150;
+---------------------------+
| book_title                |
+---------------------------+
| Волшебник изумрудного города |
| 12 месяцев                |
| Тень                      |
| Солнечный мальчик         |
| Три толстяка              |
+---------------------------+
5 rows in set (0.13 sec)
```

Рисунок 6.1 – Названия книг, вес которых более 450, но меньше 150

Давайте выведем список книг, названия которых начинаются с буквы С или Т.

```
SELECT book_title from books_data
where book_title like 'C%' OR book_title like
'T%';
```

На рисунке 6.2 приведен результат этого запроса.

```
mysql> select book_title, storage from books_data
    -> where book_title like 'C%' or book_title like 'T%';
+-------------------+---------+
| book_title        | storage |
+-------------------+---------+
| Сонная сказка     |       2 |
| Страна мудрецов   |       4 |
| Тень              |       2 |
| Тень              |       2 |
| Солнечный мальчик |       4 |
| Три толстяка      |       1 |
+-------------------+---------+
6 rows in set (0.00 sec)
```

Рисунок 6.2 – Книги, начинающиеся с буквы С или Т

Другой пример, вывести названия книг, их вес и место хранения, названия которых начинаются с буквы «С» или «Т», и которые легче 30 грамм.

```
SELECT book_title, storage, weight
from books_data
where (book_title like 'C%' OR l_name like
'T%') AND weight< 30;
```

На рисунке 6.3 приведен результат этого запроса.

```
mysql> select book_title, storage, weight from books_data
    -> where (book_title like 'C%' or book_title like 'T%')
    -> and weight <30;
+-----------------+---------+--------+
| book_title      | storage | weight |
+-----------------+---------+--------+
| Сонная сказка   |       2 |     20 |
| Страна мудрецов |       4 |     15 |
+-----------------+---------+--------+
2 rows in set (0.00 sec)
```

Рисунок 6.3 – Запрос со сложным условием

Обратите внимание на использование скобок в представленном выше операторе. Скобки предназначены для выделения различных логических условий и удаления двусмысленностей.

Оператор NOT поможет при поиске всех книг, в названии которых нет слова «сказка».

```
SELECT book_title, storage from books_data
where book_title NOT LIKE "%сказка%";
```

На рисунке 6.4 приведен результат этого запроса.

```
mysql> select book_title, storage from books_data
    -> where book_title not like '%сказка%';
+----------------------------+---------+
| book_title                 | storage |
+----------------------------+---------+
| Волшебник изумрудного города |       5 |
| 43 медвежки                |       1 |
| Ежик                       |       1 |
| Страна мудрецов            |       4 |
| 12 месяцев                 |       6 |
| Тень                       |       2 |
| Тень                       |       2 |
| Солнечный мальчик          |       4 |
| Три толстяка               |       1 |
+----------------------------+---------+
9 rows in set (0.00 sec)
```

Рисунок 6.4 – Книги, не содержащие в своем названии слово «сказка»

Показать все книги, число страниц в которых больше 30, а вес меньше 100.

```
Select book_title from books_data
where pages>30 AND weight<100;
```

На рисунке 6.5 приведен результат этого запроса.

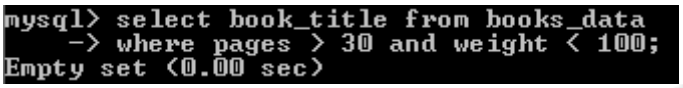


Рисунок 6.5 – Книги со страницами из заданного диапазона

Как видно из рисунка 6.5 в базе данных нет книг с числом страниц, лежащим в диапазоне от 30 и до 100.

6.2 Операторы IN и BETWEEN

Чтобы найти книги, которые называются «12 месяцев» и «Ежик», можно использовать оператор SELECT следующего вида:

```
SELECT book_title, storage from books_data
where book_title = '12 месяцев'
OR book_title = 'Ежик';
```

На рисунке 6.6 приведен результат этого запроса.

Рисунок 6.6 – Поиск книг по заданному названию

В SQL имеется более простой способ сделать это с помощью оператора IN (в множестве). Например,

```
SELECT book_title, storage
From books_data
where book_title IN ('12 месяцев', 'Ежик');
```

Результат будет аналогичен рисунку 6.6.

Использование NOT перед IN позволяет вывести данные, которые не входят во множество, определяемое условием IN.

Оператор BETWEEN используется для определения целочисленных границ. Поэтому вместо weight >= 50 AND weight <= 150 можно использовать weight BETWEEN 50 AND 150. Например,

```
select book_title, weight
from books_data
where weight BETWEEN 50 AND 150;
```

На рисунке 6.7 приведен результат этого запроса.

```
mysql> select book_title, weight from books_data
    -> where weight between 50 and 150;
+---------------------------+--------+
| book_title                | weight |
+---------------------------+--------+
| Волшебник изумрудного города |    100 |
| 12 месяцев                |    100 |
| Тень                      |     50 |
| Солнечный мальчик         |     75 |
| Три толстяка              |    100 |
+---------------------------+--------+
5 rows in set (0.06 sec)
```

Рисунок 6.7 – Поиск книг, вес которых лежит в указанном промежутке

NOT также можно использовать вместе с BETWEEN, как в следующем операторе, который выводит книги, вес которых меньше 10 или больше 30 грамм.

```
Select book_title, weight
from books_data
where weight NOT BETWEEN 10 AND 30;
```

6.3 Упорядочивание данных

Рассмотрим вопрос о том, как можно изменить порядок вывода данных, извлеченных из таблиц MySQL, используя предложение ORDER BY оператора SELECT.

Извлекаемые до сих пор данные всегда выводились в том порядке, в котором они были сохранены в таблице. В действительности SQL позволяет сортировать извлеченные данные с помощью предложения ORDER BY. Это предложение требует имя столбца, на основе которого будут сортироваться данные. Давайте посмотрим, как можно вывести список книг с упорядоченными по алфавиту названиями (в возрастающем порядке).

```
SELECT book_title from books_data
ORDER BY book_title;
```

А вот так книги можно отсортировать по весу.

```
SELECT book_title, weight
from books_data
ORDER BY weight;
```

Предложение ORDER BY может сортировать в возрастающем порядке (ASCENDING или ASC) или в убывающем порядке (DESCENDING или DESC) в зависимости от указанного аргумента.

Чтобы вывести список книг в убывающем порядке, можно использовать следующий оператор.

```
SELECT book_title, weight
from books_data
ORDER by weight DESC;
```

```
mysql> select book_title, weight from books_data
    -> order by weight desc;
+-----------------------------+--------+
| book_title                  | weight |
+-----------------------------+--------+
| Волшебник изумрудного города |    100 |
| 12 месяцев                  |    100 |
| Три толстяка                |    100 |
| Солнечный мальчик           |     75 |
| Тень                        |     50 |
| Тень                        |     45 |
| Сонная сказка               |     20 |
| 43 медвежки                 |     15 |
| Ежик                        |     15 |
| Страна мудрецов             |     15 |
+-----------------------------+--------+
10 rows in set (0.06 sec)
```

Рисунок 6.8 – Упорядочение веса книг в убывающем порядке

Возрастающий порядок (ASC) используется по умолчанию.

6.4 Ограничение количества извлекаемых данных

Далее рассмотрим, как ограничить число записей, выводимых оператором SELECT.

По мере увеличения таблиц возникает необходимость вывода только подмножества данных. Этого можно добиться с помощью предложения LIMIT.

Например, чтобы вывести из таблицы названия только первых пяти книг, используется оператор LIMIT с аргументом равным 5.

```
SELECT book_title from
books_data LIMIT 5;
```

На рисунке 6.9 приведен результат этого запроса.

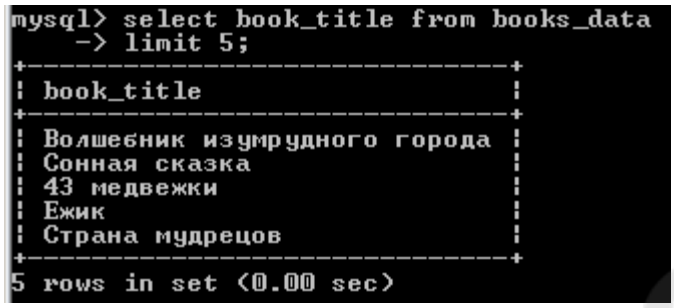


Рисунок 6.9 – Названия первых пяти книг

Это первые пять записей таблицы.

Можно соединить оператор LIMIT с оператором ORDER BY. Таким образом, следующий оператор выведет четыре самые легкие книги.

```
SELECT book_title, pages
From books_data
ORDER BY pages LIMIT 3;
```

На рисунке 6.10 приведен результат этого запроса.

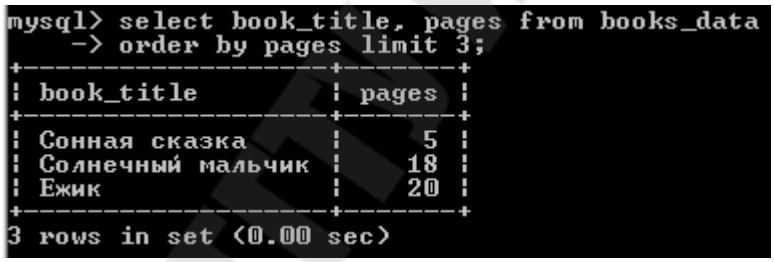


Рисунок 6.10 – Четыре самых легкие книги

LIMIT можно использовать также для извлечения подмножества данных, используя дополнительные аргументы.

Общая форма оператора LIMIT имеет следующий вид:

```
SELECT (something) from table_name
LIMIT начальная строка, извлекаемое число записей;
```

Рассмотрим пример:

```
SELECT book_title from books_data
LIMIT 6,3;
```

На рисунке 6.11 приведен результат этого запроса. Будут извлечены три строки, начиная с шестой.

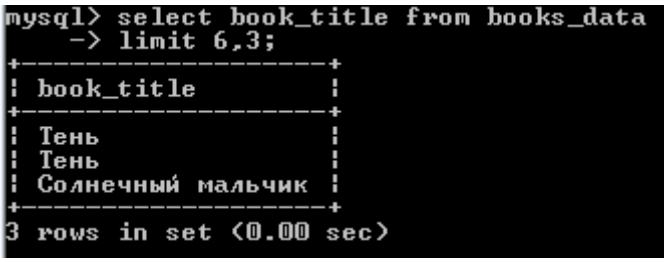


Рисунок 6.11 – Извлечение трёх строк начиная с 6-й

6.5Ключевое слово DISTINCT

Рассмотрим теперь, как выбрать и вывести записи таблиц MySQL с помощью ключевого слова DISTINCT (различный), использование которого исключает появление повторяющихся данных.

Чтобы вывести список всех книг базы данных, можно выполнить следующий оператор:

```
Select book_title from books_data;
```

На рисунке 6.12 приведен результат этого запроса.

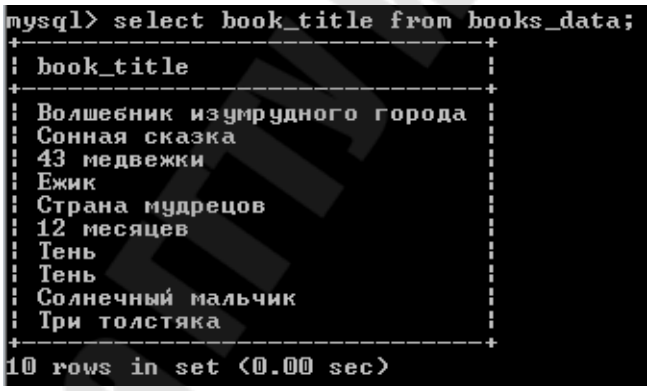


Рисунок 6.12 – Все книги базы данных «Библиотека»

Можно видеть, что список содержит повторяющиеся данные. В нашем примере, их только две, но в реальности их может быть гораздо больше. Предложение SQL DISTINCT выводит только уникальные данные. Вот как оно используется.

```
Select DISTINCT book_title from books_data;
```

На рисунке 6.13 приведен результат этого запроса.

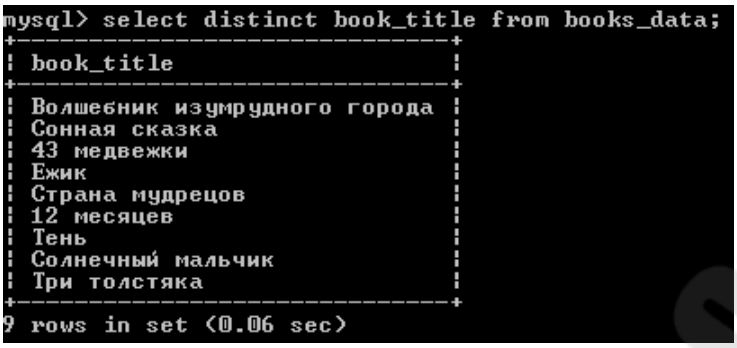


Рисунок 6.13 – Все книги базы данных библиотеки без повторов

Из этого можно видеть, что в библиотеке имеется 9 уникальных книг.

Уникальные записи можно также отсортировать с помощью ORDER BY.

```
select DISTINCT weight from books_data
ORDER BY weight;
```

На рисунке 6.14 приведен результат этого запроса.

Рисунок 6.14 – Значения весов книг без повторов

DISTINCT часто используется вместе с функцией COUNT, которая будет рассмотрена далее.

6.6 Изменение записей

Команда UPDATE выполняет изменение данных в таблицах. Она имеет очень простой формат.

```
UPDATE имя_таблицы SET
имя_столбца_1 = значение_1,
имя_столбца_2 = значение_2,
имя_столбца_3 = значение_3, ...
[WHERE условия];
```

Как и все другие команды SQL можно вводить ее на одной строке или на нескольких строках.

Рассмотрим несколько примеров.

Предположим, для книги «Сонная сказка» неверно было задано число страниц, реальное число страниц на 2 больше. Предыдущее число страниц знать не требуется. Можно воспользоваться арифметическими операторами.

```
UPDATEbooks_dataSET
pages = pages + 2
WHERE book_title='Сонная сказка';
```

В качестве другого примера можно попробовать изменить состояние книги «Ежик» с «good» на «bad».

```
Update books_data SET
State = 'bad'
WHERE book_title = 'Ежик';
```

На рисунке 6.15 приведен результат этого запроса.

Рисунок 6.15 – Данные обновлены

Важно также перед выполнением изменений внимательно изучить часть оператора с условием, так как легко можно изменить не те данные. Оператор UPDATE без условий изменит все данные столбца во всех строках. Надо быть очень осторожным при внесении изменений.

Задания для самоконтроля

1. Что делает следующий оператор?

```
SELECT book_title, state from books_data
Where book_title NOT LIKE '%город%' AND pages
< 30;
```

2. Вывести все идентификационные номера и названия книг весом между 100 и 300 граммами.

3. Выберите названия книг с датой поступления до 2009 года и в отличном состоянии.

4. Найдите все книги, которые имеют состояние «good» и «bad».

5. Выведите список книг, число страниц которых от 70 до 900.

6. Что делает следующий оператор?

```
SELECT book_name, r_date, storage
from books_data
where state NOT IN ('perfect', 'good');
```

7. Вот более сложный оператор, который объединяет BETWEEN и IN. Что он делает?

```
SELECT book_title, state, storage
from books_data
where storage NOT IN('good', 'bad')
AND pages NOT BETWEEN 30 and 100;
```

8. Вывести список книг в порядке, определяемом весом, который они имеют.

9. Выведите список книг в убывающем порядке их числа страниц.

10. Что делает следующий оператор?

```
SELECT book_title, state, pages
from books_data
ORDER BY book_title DESC, weight ASC;
```

11. Найдите 5 самых легких книг.

12. Извлеките 2 записей, начиная с 6 строки.

13. Выведите две книги, с самым большим объемом (число страниц).

14. Что делает следующий оператор?

```
SELECT book_title, state, weight
from books_data
ORDER BY weight DESC LIMIT 3;
```

15. Сколько уникальных вариантов страниц книг имеется в библиотеке? Представьте их в убывающем порядке.

16. Сколько различных статусов имеется в базе данных?

17. Измените состояние всех книг с «perfect» на «good».

7 Команды обработки данных

7.1 Поиск минимального и максимального значений

В MySQL имеются встроенные функции для вычисления минимального и максимального значений.

SQL имеет 5 агрегатных функций.

1. MIN(): минимальное значение;
2. MAX(): максимальное значение;
3. SUM(): сумма значений;
4. AVG(): среднее значений;
5. COUNT(): подсчитывает число записей.

В этом параграфе мы рассмотрим поиск минимального и максимального значений столбца.

```
select MIN(weight) from books_data;
```

На рисунок 7.1 приведен результат запроса.

```
mysql> select min(weight) from books_data;
+-------------+
| min(weight) |
+-------------+
|          15 |
+-------------+
1 row in set (0.08 sec)
```

Рисунок 7.1 – Поиск минимального веса книги

```
select MAX(weight)from books_data;
```

На рисунке 7.2 приведен результат запроса.

```
mysql> select max(weight) from books_data;
+-------------+
| max(weight) |
+-------------+
|         100 |
+-------------+
1 row in set (0.00 sec)
```

Рисунок 7.2 – Поиск максимального веса книги

7.2 Поиск среднего значения и суммы

Агрегатная функция SUM()вычисляет общую сумму значений в столбце. Для этого необходимо задать имя столбца, которое должно быть помещено внутри скобок.

Давайте посмотрим, сколько весят все книги в библиотеке.

```
select SUM(weight) from books_data;
```

На рисунок 7.3 приведен результат этого запроса.

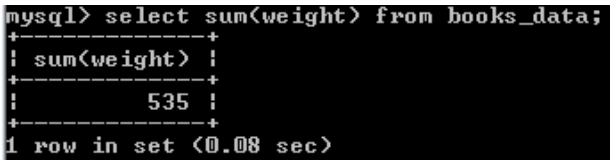


Рисунок 7.3 – Сумма всех весов книг

Имеются также дополнительные возможности команды SELECT. Значения можно складывать, вычитать, умножать или делить. Например, `select SUM(width)+SUM(height) from squares;`

В действительности можно записывать полноценные арифметические выражения.

Агрегатная функция AVG() используется для вычисления среднего значения данных в столбце.

```
select avg(pages) from books_data;
```

На рисунке 7.4 приведен результат запроса.

Рисунок 7.4 – Средний число страниц всех книг

Следующий пример вычисляет средний вес всех книг библиотеки.

```
Select avg(weight) from books_data;
```

7.3 Именование названий столбцов

MySQL позволяет задавать имена для выводимых столбцов. Поэтому вместо book_title и т.д. можно использовать более понятные и наглядные термины. Это делается с помощью оператора AS.

```
select avg(pages) AS
'Среднее число страниц' from
books_data;
```

На рисунке 7.5 приведен результат этого запроса.

```
mysql> select avg(pages) as
    -> 'Среднее число страниц' from books_data;
+------------------------+
| Среднее число страниц |
+------------------------+
|               29.4000 |
+------------------------+
1 row in set (0.01 sec)
```

Рисунок 7.5 – Вывод средней цены с использованием оператора AS

Такие псевдо-имена могут сделать вывод более понятным для пользователей. Важно только помнить, что при задании псевдо-имен с пробелами необходимо заключать такие имена в кавычки.

7.4 Подсчет числа записей

Агрегатная функция COUNT() подсчитывает и выводит общее число записей. Например, чтобы подсчитать общее число записей в таблице, выполните следующую команду.

```
select COUNT(*) from books_data;
```

На рисунок 7.6 приведен результат запроса. Знак * выводи «все данные».

```
mysql> select count(*) from books_data;
+----------+
| count(*) |
+----------+
|       10 |
+----------+
1 row in set (0.03 sec)
```

Рисунок 7.6 – Общее количество записей

Теперь давайте подсчитаем общее число всех книг в плохом состоянии.

```
select COUNT(*) from books_data
where state = 'bad';
```

На рисунке 7.7 приведен результат этого запроса.

```
mysql> select count(*) from books_data
    -> where state='bad';
+----------+
| count(*) |
+----------+
|        3 |
+----------+
1 row in set (0.06 sec)
```

Рисунок 7.7 – Общее количество книг в плохом состоянии

7.5 Группировка данных

Предложение GROUP BY позволяет группировать аналогичные данные. Поэтому, чтобы вывести все уникальные названия книг в таблице, можно выполнить команду

```
Select book_title from books_data
GROUP BY book_title;
```

На рисунке 7.8 приведен результат этого запроса.

```
mysql> select book_title from books_data
    -> group by book_title;
+-----------------------------+
| book_title                  |
+-----------------------------+
| 12 месяцев                  |
| 43 медвежки                 |
| Волшебник изумрудного города |
| Ежик                        |
| Страна мудрецов             |
| Солнечный мальчик           |
| Сонная сказка               |
| Три толстяка                |
| Тень                        |
+-----------------------------+
9 rows in set (0.00 sec)
```

Рисунок 7.8 – Все уникальные названия книг

Можно видеть, что это аналогично использованию DISTINCT.
Подсчитаем число книг имеющих определенное название.

```
Select book_title, count(*)
From books_data GROUP BY book_title;
```

На рисунке 7.9 приведен результат запроса.

```
+-----------------------------+----------+
| book_title                  | count(*) |
+-----------------------------+----------+
| 12 месяцев                  |        1 |
| 43 медвежки                 |        1 |
| Волшебник изумрудного города |        1 |
| Ежик                        |        1 |
| Страна мудрецов             |        1 |
| Солнечный мальчик           |        1 |
| Сонная сказка               |        1 |
| Три толстяка                |        1 |
| Тень                        |        2 |
+-----------------------------+----------+
9 rows in set (0.00 sec)
```

Рисунок 7.9 – Количество книг по названиям

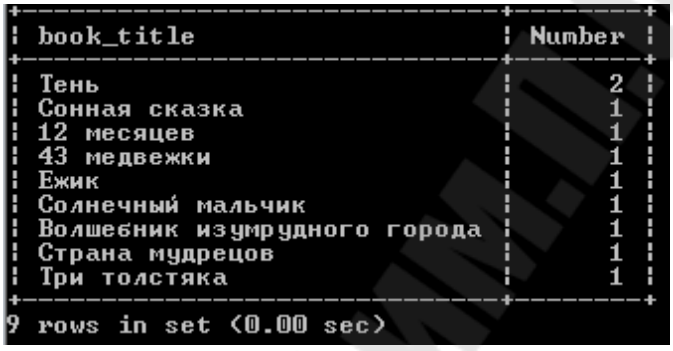
В предыдущей команде MySQL сначала создает группы различных названий, а затем выполняет подсчет в каждой группе.

7.6 Сортировка данных

Теперь давайте найдем и выведем число книг, имеющих различные названия, и отсортируем их с помощью ORDER BY.

```
select book_title, count(*) AS Number
from books_data
GROUP BY book_title
ORDER BY Number desc;
```

На рисунке 7.10 приведен результат этого запроса.



book_title	Number
Тень	2
Сонная сказка	1
12 месяцев	1
43 медвежки	1
Ежик	1
Солнечный мальчик	1
Волшебник изумрудного города	1
Страна мудрецов	1
Три толстяка	1

9 rows in set (0.00 sec)

Рисунок 7.10 – Количество книг по названиям с сортировкой

7.7 Работа с датой и временем

До сих пор мы имели дело с текстом (varchar) и числами (int). Теперь пришло время ознакомиться с типом данных date (дата).

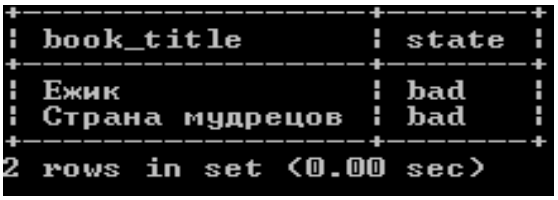
Даты в MySQL всегда представлены с годом, за которым следует месяц и затем день месяца. Даты часто записывают в виде YYYY-MM-DD, где YYYY – 4 цифры года, MM – 2 цифры месяца и DD – 2 цифры дня месяца.

Тип столбца даты позволяет выполнять несколько операций, таких как сортировка, проверка условий с помощью операторов сравнения и т.д.

Рассмотрим некоторые примеры использования операторов, при работе с датой.

```
select book_title, state from books_data
where r_date = '2001-02-10';
```

Результат запроса представлен на рисунке 7.11.



book_title	state
Ежик	bad
Страна мудрецов	bad

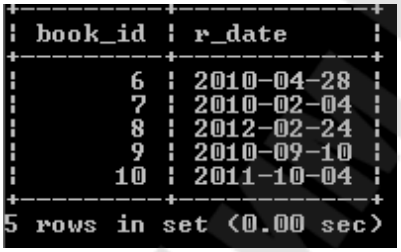
2 rows in set (0.00 sec)

Рисунок 7.11 – Поиск по дате поступления

Оператор != означает неравенство. При работе с датами в MySQL требуется, чтобы даты были заключены в кавычки.

```
select book_id, r_date from books_data
where r_date>= '2000-01-01';
```

Результат запроса представлен на рисунке 7.12.



book_id	r_date
6	2010-04-28
7	2010-02-04
8	2012-02-24
9	2010-09-10
10	2011-10-04

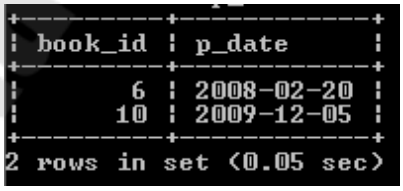
5 rows in set (0.00 sec)

Рисунок 7.12 – Поиск с использованием оператора >=

Далее рассмотрен пример использования диапазонов.

```
select book_id, p_date from books_data
where p_date BETWEEN
'2004-01-01' AND '2011-01-01';
```

Результат запроса представлен на рисунке 7.13.



book_id	p_date
6	2008-02-20
10	2009-12-05

2 rows in set (0.05 sec)

Рисунок 7.13 – Поиск по дате публикации в определенном диапазоне

Тот же запрос можно представить без конструкции BETWEEN:

```
select book_id, p_date from books_data
where p_date>= '2004-01-01'
AND p_date<= '2011-01-01';
```

Результат запроса будет аналогичен рисунку 7.13.

Использование Date для сортировки данных, рассмотрено ниже:

```
select book_id, p_date from books_data
ORDER BY p_date;
```

Результат запроса представлен на рисунке 7.14.

```
+---------+------------+
| book_id | p_date     |
+---------+------------+
|       5 | 1997-02-15 |
|       4 | 1998-03-15 |
|       3 | 1998-05-05 |
|       1 | 2000-01-21 |
|       9 | 2000-02-05 |
|       7 | 2000-02-25 |
|       2 | 2000-06-25 |
|       6 | 2008-02-20 |
|      10 | 2009-12-05 |
|       8 | 2011-03-15 |
+---------+------------+
10 rows in set (0.00 sec)
```

Рисунок 7.14–Сортировка по дате публикации

Вот как можно выбрать книги, которые были опубликованы в феврале.

```
select book_id, p_date from books_data
where MONTH(p_date) = 2;
```

Результат запроса представлен на рисунке 7.15.

```
+---------+------------+
| book_id | p_date     |
+---------+------------+
|       5 | 1997-02-15 |
|       6 | 2008-02-20 |
|       7 | 2000-02-25 |
|       9 | 2000-02-05 |
+---------+------------+
4 rows in set (0.09 sec)
```

Рисунок 7.15 – Поиск по месяцу

Можно также использовать вместо чисел названия месяцев.

```
select book_id, p_date from books_data
where MONTHNAME(p_date) = 'January';
```

Результат запроса представлен на рисунке 7.16.

```
+---------+------------+
| book_id | p_date     |
+---------+------------+
|       1 | 2000-01-21 |
+---------+------------+
1 row in set (0.08 sec)
```

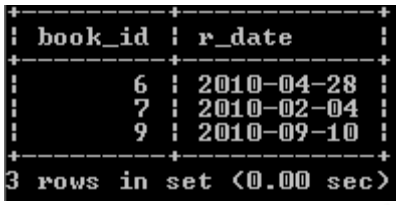
Рисунок 7.16 – Использование названия месяца в поиске

При использовании названий месяцев важен регистр. Поэтому January будет работать, а JANUARY не будет!

Аналогично можно выбрать книги, поступившие в определенный год или определенный день.

```
select book_id, r_date from books_data
where year(r_date) = 2010;
```

Результат запроса представлен на рисунок 7.17.

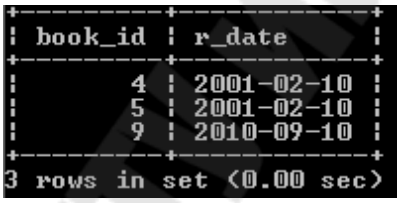


```
+---------+------------+
| book_id | r_date     |
+---------+------------+
|       6 | 2010-04-28 |
|       7 | 2010-02-04 |
|       9 | 2010-09-10 |
+---------+------------+
3 rows in set (0.00 sec)
```

Рисунок 7.17 – Поиск по году поступления

```
select book_id, r_date from books_data
where DAYOFMONTH(r_date) = 10;
```

Результат запроса представлен на рисунке 7.18.



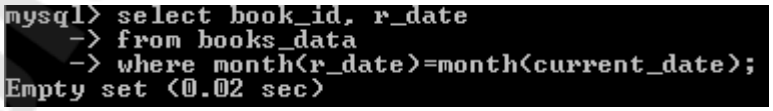
```
+---------+------------+
| book_id | r_date     |
+---------+------------+
|       4 | 2001-02-10 |
|       5 | 2001-02-10 |
|       9 | 2010-09-10 |
+---------+------------+
3 rows in set (0.00 sec)
```

Рисунок 7.18 – Поиск по дню поступления

Текущую дату, месяц и год можно вывести с помощью аргумента CURRENT_DATE предложений DAYOFMONTH(), MONTH() и YEAR(), соответственно. То же самое можно использовать для выборки данных из таблиц.

```
select book_id, r_data from books_data
where MONTH(r_date) = MONTH(CURRENT_DATE);
```

Результат запроса представлен на рисунке 7.19.



```
mysql> select book_id, r_date
    -> from books_data
    -> where month(r_date)=month(current_date);
Empty set (0.02 sec)
```

Рисунок 7.19 – Поиск по текущему месяцу

7.8 Математические функции

Описанные ниже функции выполняют различные математические операции. В качестве аргументов большинство из них принимает

числа с плавающей запятой и возвращает результат аналогичного типа.

Для того, чтобы воспользоваться функцией необходимо в приглашении MySQLпрописать зарезервированное слово SELECT, а затем саму функцию. Например, `mysql>select abs(-3).`

Ниже приведена таблица 7.1, в которой можно найти форматы записи и описание некоторых математических функций.

Таблица 7.1 – Некоторые математические функции

Формат записи функции	**Описание функции**
ABS(число)	Возвращает модуль числа
CEILING(число) CEIL(число)	Округляет число до ближайшего большего целого числа.
DEGREES(число)	Возвращает аргумент, преобразованный из радианов в градусы.
FLOOR(число)	Округляет число до ближайшего меньшего целого числа.
GREATEST(a,b,c,…,d)	Эта функция возвращает наибольшее значение из списка. Она может работать как с числами, так и со строками.
LEAST(...)	Возвращает наименьшее значение из списка.
MOD(число1,число2) число1%число2 число1 MOD число2	Возвращает остаток от деления первого числа на второе подобно оператору %.
POW(число1,число2) POWER(число1,число2)	Возвращает значение число1, возведенное в степень число2.
RAND([число])	Возвращает случайное число двойной точности в диапазоне от 0 до 1. Если указан целочисленный аргумент, он служит начальным числом для генератора случайных чисел (генерируя повторяющуюся последовательность). Если аргумент отсутствует, используется значение системных часов.
ROUND(число[,точность])	Округляет число с плавающей запятой до целого числа или, если указан второй аргумент, до заданного количества цифр после запятой. Если точность отрицательная, обнуляется целая часть числа.
SIGN(число)	Возвращает знак аргумента как -1, 0 или 1, в зависимости от того, число отрицательное, нуль или положительное.

Таблица 7.1 (продолжение)

SQRT(число)	Возвращает квадратный корень числа.
TRUNCATE(число1,число2)	Возвращает число1 с дробной частью, усеченной до число2 десятичных разрядов. Если число2 равно 0, результат не имеет точки и дробной части. Если число2 отрицательное, целая часть числа длиной число2 обнуляется.

Задания для самоконтроля

1. Вывести сумму весов тех книг, которые имеют плохое состояние.
2. Найдите книги с минимальным и максимальным количеством страниц.
3. Вывести идентификаторы, даты поступления и место хранения книг, опубликованных не ранее 2004 г.
4. Вывести идентификаторы, названия и даты публикаций книг, опубликованных в 2011 г., и отсортируйте записи на основе названий книг в алфавитном порядке.
5. Выведите идентификаторы книг, поступивших в текущем месяце.
6. Вывести список уникальных годов поступления и число книг, поступивших в каждом таком году.
7. Сколько книг поступило в каждом месяце? Выдача должна содержать названия месяцев (не номера), и записи должны быть упорядочены по убыванию по месяцам, начиная от наибольшего номера.

8 Создание связей между таблицами

8.1 Запросы данных из нескольких таблиц, теоретические сведения

Нередко web-разработчикам приходится сталкиваться с необходимостью выбора данных из нескольких таблиц одновременно. Необходимость использования таких запросов очевидна. Такие запросы уменьшают число обращений к базе данных, тем самым уменьшая время выполнения скрипта и нагрузку на сервер.

Далее рассмотрим, как пользоваться многотабличными запросами. Допустим, у нас есть две таблицы:

Таблица table1:

id	name
1	А
2	Б
3	В

Таблица table2:

id	letter
2	Г
3	Д
4	Е

Произведём простейший запрос к двум таблицам.

```
select * from table1,table2
where table1.id = table2.id
```

В результате этого запроса получится следующая таблица:

id	name	id	letter
2	Б	2	Г
3	В	3	Д

То есть, данный запрос строго выбирает из двух таблиц только те строки, столбцы id которых совпадают. Аналогичного результата можно добиться с помощью следующего запроса:

```
select * from table1 JOIN table2
wheretable1.id=table2.id
```

Для формирования условия в запросах, использующих объединение JOIN, вместо ключевого слова WHERE предпочтительно использовать ключевое слово ON, как это продемонстрировано в следующем примере.

```
select * from table1 JOIN table2
ONtable1.id=table2.id
```

Выше было продемонстрировано перекрестное объединение таблиц table1 и table2. При этом результирующая таблица содержит комбинации строк обеих таблиц, удовлетворяющих условию table1.id = table2.id.

Левое объединение (LEFT JOIN) позволяет включить в результирующую таблицу строки «левой» таблицы table1, которой не нашлось соответствие в «правой» таблице table2.

```
select * from table1 LEFT JOIN table2
ON table1.id=table2.id
```

Результатом будет следующая таблица:

id	name	id	letter
1	А	NULL	NULL
2	Б	2	Г
3	В	3	Д

Как видно из примера, записи в таблице table1 со значением id = 1 не нашлось соответствия в таблице table2, т.к. поле id в ней принимает значения 2,3,4. Тем не менее в результирующую таблицу запись включена, при этом значения полей из таблицы table2 принимают значение NULL. Следует заметить, что для задания условия вместо ключевого слова WHERE при левом и правом объединениях используется ключевое слово ON.

В следующем примере демонстрируется «правое» объединение при помощи конструкции RIGHT JOIN.

```
select * from table1 RIGHT JOIN table2
ON table1.id=table2.id
```

Результирующая таблица:

id	name	id	letter
2	Б	2	Г
3	В	3	Д
NULL	NULL	4	Е

Пример показывает, что при правом объединении возвращаются строки, удовлетворяющие условию table1.id = table2.id, и строки «правой» таблицы table2, которым не нашлось соответствия в левой таблице table1.

8.2 Выборка данных из нескольких таблиц

В этом подразделе создадим некоторые таблицы, которые были выделены нами при проектировании базы данных «Библиотека» в главе 1, п. 1.4. Научимся создавать различные связи между таблицами и получать данные из нескольких таблиц на конкретном примере.

Создадим в текстовом редакторе файл issuance.dat, который содержит оператор создания таблицы CREATE следующего вида:

```
CREATE TABLE authors (
author_idint unsigned not null auto_increment primary key,
l_name varchar(20),
f_name varchar(10),
patronymic varchar(20),
a_email varchar(60)
);
```

и последовательность операторов INSERT, например, такого вида (количество записей может быть произвольно):

```
INSERT INTO authors_data (l_name, f_phone, patronymic, a_email)
values ('Волков', 'Александр', 'Мелентьевич', 'avolkov@rambler.ru');
```

Затем загрузим этот файл, как мы делали раньше, в базу данных.

Поскольку тип связи между таблицами «Авторы» и «Книги» многие-ко-многим (∞:∞), то необходимо создать еще одну связующую их таблицу «Авторы_Книги».

На рисунке 8.1 представлены команды создания таблиц «Авторы_Книги» и «Авторы». Как видно из рисунка 8.1 таблица «Авторы_Книги» содержит только два поля: идентификационный номер книги и идентификационный номер автора.

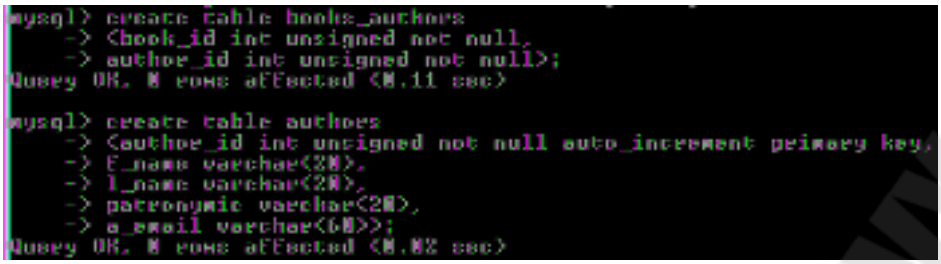


Рисунок 8.1 – Создание новой таблицы

Заполним таблицы данными. После это делаем запрос данных из таблицы «Авторы» и «Книги» с помощью оператора RIGHTJOINON, как показано на рисунке 8.2.

```
select books_data.book_title, authors.l_name
from authors join books_authors
right join books_data on
(authors.author_id = books_authors.author_id)
and
books_authors.book_id=books_data.book_id);
```

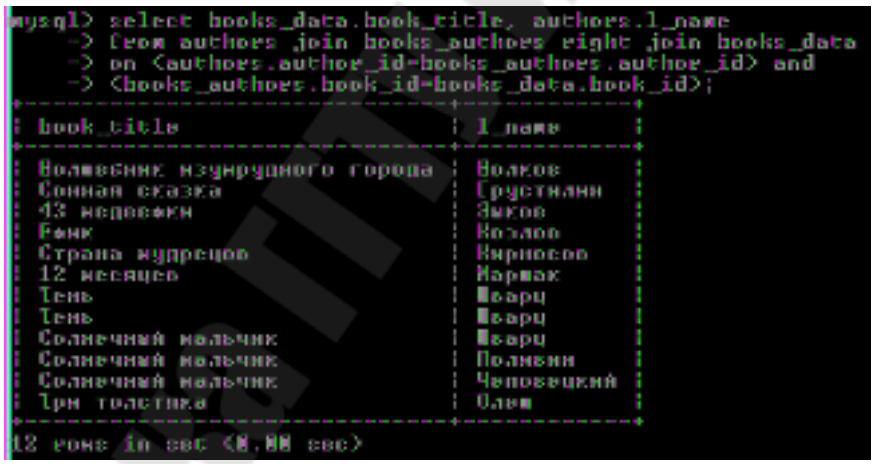


Рисунок 8.2 – Запрос данных их двух таблиц

Далее создадим таблицы «Выдачи» и «Читатели», используя операторы CREATEи INSERT.На рисунке 8.3 и 8.4 приведены команды создания таблиц «Выдачи» и «Читатели», а также выполнены запрос, показывающие все данные из созданных таблиц.

Таблица «Выдачи» содержит следующие поля: код выдачи, код книги, код читателя, дату выдачи, срок выдачи (дата возврата книги).

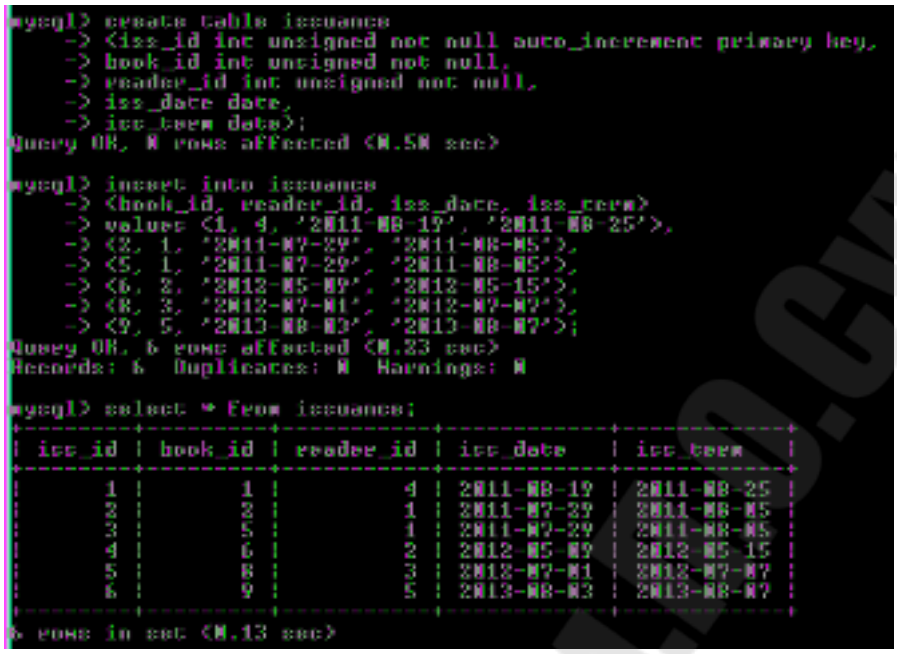


Рисунок 8.3 – Создание таблицы «Выдачи»

Рисунок 8.4 – Создание таблицы «Читатели»

На рисунке 8.5 приведен запрос, выводящий необходимые данные из трех таблиц «Книги», «Выдачи» и «Читатели». Этот запрос показывает сроки сдачи книг читателями. По полученной таблице можно узнать имя, фамилию и телефон читателя, какая книга у него «на руках» и срок сдачи этой книги в библиотеку.

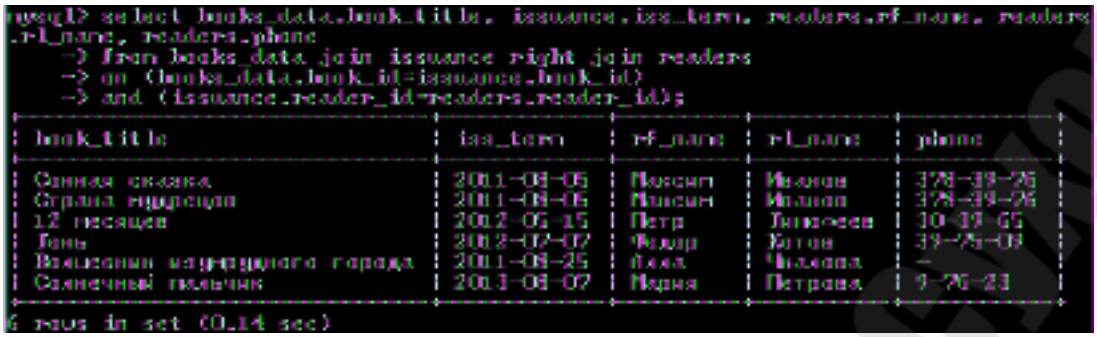


Рисунок 8.5 – Запрос из трех таблиц

При желании можно получить данные о тех читателях, срок сдачи книг у которых уже просрочен.

8.3 Управление пользователями и привилегиями

Рассмотрим операторы управления пользователями и привилегиями (таблица 8.1).

Существует три группы привилегий:

- данные
- структура
- администрирование

Первая группа привилегий – данные

SELECT - эта привилегия позволяет делать выборку (вытаскивание) записей из таблиц баз данных.

INSERT - привилегия, которая необходима для добавления новых записей в таблицу.

UPDATE - право, позволяющее обновлять записи в таблице.

DELETE - эта привилегия позволяет удалять записи из таблицы.

FILE - разрешает делать выборку записей и записывать данные в файл, а также считывать их оттуда.

Вторая группа привилегий (привилегии пользователей MySQL) – структура

CREATE - привилегия, позволяющая создавать новые базы данных, а также новые таблицы в базе данных.

ALTER - привилегия, позволяющая переименовывать таблицы, вставлять новые поля в таблицу, удалять поля из таблицы, а также модифицировать их.

INDEX - разрешает создавать индекс по определённому полю и удалять его.

DROP - право, которое позволяет удалять либо таблицы, либо целые базы данных.

CREATE TEMPORARY TABLES - возможность создавать временные таблицы, которые хранятся во время сессии, а после окончания сессии данная таблица автоматически удаляется.

Третья группа привилегий – администрирование

GRANT - привилегия, которая позволяет создавать новых пользователей, а также менять права у существующих. Тут есть очень важная деталь: нельзя изменять значения привилегий, которыми сам не обладаешь. То есть если человек обладает привилегией GRANT, но не обладает привилегией SELECT, то он не может новым пользователям дать привилегию SELECT.

SUPER - позволяет использовать команду "kill", то есть убить поток. Поток - это текущее подключение другого пользователя к базе данных.

PROCESS - привилегия, позволяющая выполнить команду "processlist", которая показывает список потоков.

RELOAD - позволяет открывать и закрывать файлы журналов, а также перечитывать таблицы привилегий пользователей.

SHUTDOWN - привилегия, позволяющая выполнить команду "shutdown", отключающая работу сервера.

SHOW DATABASES - разрешает просматривать все существующие базы данных.

REFERENCES - данная привилегия ещё не доступна, а только зарезервирована для использования в будущем.

LOCK TABLES - позволяет блокировать таблицы от указанных потоков.

EXECUTE - позволяет запускать хранимые процедуры.

REPLICATION CLIENT - даёт право получать местонахождение ведущего (master) и ведомых (slaves) серверов.

REPLICATION SLAVE - это привилегия, позволяющая читать ведомым журнала ведущего сервера.

Специальные привилегии

MAX QUERIES PER HOUR - максимальное количество запросов в час, которое может отправить пользователь.

MAX UPDATES PER HOUR - максимальное количество команд в час, которые каким-либо образом изменяют либо таблицу, либо базу данных.

MAX CONNECTIONS PER HOUR - максимальное количество подключений в час, которое может сделать пользователь.

При работе с операторами управления пользователями и привилегиями, не забывайте ставить точку между названием базы данных и названием таблицы.

Таблица 8.1 – Управление пользователями и привилегиями

Синтаксис	Описание
SHOWGRANTSFOR 'пользователь'@'localhost';	Просмотр привилегий и прав пользователя
GRANT операции ON название_БД.название_таблицы TO 'имя пользователя '@'localhost' IDENTIFIED BY 'пароль'; Пример: GRANT SELECT, INSERT, DELETE, UPDATE ON new.* TO 'user'@'localhost' IDENTIFIED BY 'pas';	Создание пользователя с правами. В примере создается пользователь user для таблицы new с правами
REVOKE DELETE ON название_БД.* FROM 'пользователь'@'localhost'; (где * означает все таблицы)	Удаление прав
DELETE FROM mysql.user WHERE USER='пользователь@localhst';	Удаление пользователя

Задания для самоконтроля

1. Каким образом происходит объединения двух таблиц?
2. Что такое конструкция JOIN ON ?
3. Чем отличается RIGHT JOIN от LEFT JOIN?
4. Выдать названия книг и читателей, которые взяли книги в марте текущего года.

9 Работа с базами данных в phpMyAdmin

9.1 Основные сведения о phpMyAdmin

phpMyAdmin — веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL. phpMyAdmin позволяет через браузер осуществлять администрирование сервера MySQL, запускать команды SQL и просматривать содержимое таблиц и баз данных. Приложение пользуется большой популярностью у веб-разработчиков, так как позволяет управлять СУБД MySQL без непосредственного ввода SQL команд, предоставляя дружественный интерфейс.

На сегодняшний день phpMyAdmin широко применяется на практике. Последнее связано с тем, что разработчики интенсивно развивают свой продукт, учитывая все нововведения СУБД MySQL. Подавляющее большинство российских провайдеров используют это приложение в качестве панели управления для того, чтобы предоставить своим клиентам возможность администрирования выделенных им баз данных.

Для работы с hphMyAdminнеобходимо установить локальный веб-сервер, например, DENWER (см. www.denwer.ru).

9.2 Управление пользователями в hphMyAdmin

В этом подразделе подробно рассмотрим вопросы о том, как создать нового пользователя в phpMyAdmin, как редактировать пользователя в phpMyAdmin и как удалить пользователя в phpMyAdmin.

Для того, чтобы *создать нового пользователя*, нужно зайти на главную страницу phpMyAdminи выбрать пункт «Привилегии». Далее перейти по ссылке «Добавить нового пользователя». Настроить параметры нового пользователя и нажать на кнопку «Пошёл».

Параметры нового пользователя:

- Имя пользователя. Если указать в выпадающем списке «Любой пользователь», то логин задавать не нужно. Иначе можно задавать обычный логин.
- Хост. Это тот адрес, с которого данный пользователь может подключаться. Как правило, пишут «localhost», чтобы подключаться можно было только с этого же хоста, однако, иногда требуется под-

ключение с других хостов. В таком случае нужно выбрать в выпадающем списке «Любой хост».

- Пароль. Ввести пароль. Если не хотите использовать пароль, то можете выбрать в выпадающем списке «Без пароля». Обратите внимание: «Без пароля» - это не то же самое, что «Любой пароль».
- Подтверждение. Если пароль указан, то повторите его ввод.
- Глобальные привилегии. Рассмотрены ранее.

Для того чтобы *редактировать пользователей* в phpMyAdmin, нужно зайти на главную страницу, выбрать пункт «Привилегии», далее выбрать пользователя, которого хотите отредактировать, изменить настройки пользователя и нажать на кнопку «Пошёл». При изменении настроек существующих пользователей принцип тот же, что и при настройке новых пользователей.

Для того, чтобы *удалить пользователя* в phpMyAdmin, нужно зайти на главную страницу, выбрать пункт «Привилегии», далее выбрать пользователя, которого хотите удалить и нажать на кнопку «Пошёл».

9.3 Управление базами данных и таблицами в phpMyAdmin

С базами данных в phpMyAdmin можно проводить следующие операции: создавать, редактировать и удалять.

Для того, чтобы *создать новую базу данных*, необходимо зайти на главную страницу, задать имя для новой базы данных, выбрать кодировку и нажать на кнопку «Создать». Если у Вас на сайте будет только русские и латинские буквы, то ставьте кириллицу (cp1251_general_ci). Если у Вас будет мультиязычный сайт, то ставьте unicode (utf8_general_ci).

Для того, чтобы *редактировать базу данных*, нужно зайти на главную страницу, выбрать из выпадающего списка имя базы данных, которую нужно отредактировать, далее перейти в пункт «Операции», отредактировать базу данных и нажать на соответствующую настройке кнопку «Пошёл».

Для того, чтобы *удалить базу данных*, нужно зайти на главную страницу, выбрать из выпадающего списка имя базы данных, которую нужно удалить, далее перейти в пункт «Уничтожить» и подтвердить удаление выбранной базы данных.

С таблицами в phpMyAdmin можно проводить следующие операции: создавать, редактировать и удалять.

Для того, чтобы создать таблицу, нужно зайти на главную страницу phpMyAdmin, выбрать из выпадающего списка имя базы данных, в которую будет добавлена таблица, ввести имя новой таблицы, указать количество полей (столбцов) и нажать на кнопку «Пошёл». Далее нужно указать соответствующие настройки для каждого поля, а потом для таблицы. Нажать кнопку «Пошёл».

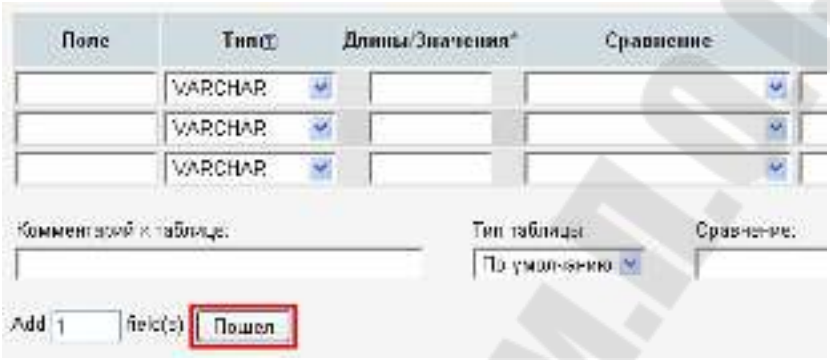


Рисунок 9.1 – Настройка полей таблицы

Кратко рассмотрим настройки для полей таблицы:

ИМЯ. Нельзя давать зарезервированное слово, например, "index".

ТИП. Тип данных столбцов.

ДЛИНА/ЗНАЧЕНИЯ. Здесь можно указать предельные значения(длины) переменных.

КОДИРОВКА. Разумеется, данная настройка актуальна только для строковых типов.

АТРИБУТЫ. Относится только к числам. Если число заведомо положительное, то включите опцию "UNSIGNED", что позволит расширить положительный диапазон значений выбранного типа. Также есть опция "UNSIGNED ZEROFILL", которая редко используется, но делает она следующее: заполняет нулями все неиспользованные старшие разряды.

НОЛЬ. Эта опция позволяет включить возможность значения "null". Запомните: "null" - это не 0, и не пустая строка. Это пустой объект, то есть ничего!

ПОУМОЛЧАНИЮ. Полезная опция, позволяющая задать значение поля по умолчанию.

ДОПОЛНИТЕЛЬНО. Позволяет задать "auto_increment". И чтобы не думать об уникальности этого, достаточно просто включить опцию "auto_increment".

Дальше идут РАДИОПЕРЕКЛЮЧАТЕЛИ:

– Первичный ключ. Как правило, этим ключом назначают поле ID. Означает этот ключ, что значение поля уникально для всей таблицы, более того, по этому полю создаётся индекс.

– Индекс. Собственно, создавать или нет для этого поля индекс.

– Уникальное. Если поставить эту опцию, то значение данного поля должно быть уникальным.

– И флажок "ПолнТекст" означает, создавать полнотекстовый индекс или нет.

КОММЕНТАРИИ. Это поле можете заполнять на своё усмотрение.

MIME-тип. Это поле нужно заполнять, если у Вас какой-нибудь особенное значение будет храниться в поле, например, изображение. Как правило, ставится "auto-detect".

ТРАНСФОРМАЦИИ БРАУЗЕРА. Используется крайне редко.

ОПЦИИ трансформации браузера. Тут можно задать параметры для трансформации, если Вы их используете.

Далее опишем настройки таблицы:

КОММЕНТАРИЙ к таблице. Можете заполнить, а можете ничего не писать.

ТИП к таблице:

– MyISAM. Самый популярный тип таблиц в MySQL, и он подходит для большинства задач.

– Heap. Особенность данной таблицы в том, что она хранится в памяти, в результате данный тип таблиц обрабатывается очень быстро. Идеальный вариант для временных таблиц. Разумеется, при сбое в работе все данные будут утеряны.

– Merge. Этот тип представляет собой совокупность обычный таблиц MyISAM.

КОДИРОВКА таблицы. Абсолютно те же правила, что и для задания кодировки для полей.

Для того, чтобы *изменить настройки таблицы*, нужно зайти на главную страницуphpMyAdmin, выбрать из выпадающего списка имя базы данных, где находится требуемая таблица, кликнуть по имени таблицы, которую нужно отредактировать, далее перейти в пункт «Операции», задать необходимые настройки и нажать на соответствующую настройке кнопку «Пошёл».

Для того, чтобы *редактировать поля в таблице*, необходимо зайти на главную страницу phpMyAdmin, выбрать из выпадающего списка имя базы данных, в которой находится искомая таблица, далее щелкнуть на значок карандаша напротив поля, которое нужно отредактировать, изменить необходимые настройки и нажать на кнопку «Сохранить».

Для того, чтобы *удалить таблицу*, необходимо зайти на главную страницу phpMyAdmin, выбрать из выпадающего списка имя базы данных, в которой находится искомая таблица, кликнуть по имени таблицы, которую хотите удалить, перейти по ссылке «Уничтожить», далее подтвердить удаление.

Для того, чтобы *удалить поле в таблице*, необходимо зайти на главную страницу phpMyAdmin, выбрать из выпадающего списка имя базы данных, где находится искомая таблица, кликнуть по имени таблицы, в которой нужно удалить поле, щелкнуть на значок «Крестик», напротив того поля, которое хотите удалить, далее подтвердить удаление.

С записями можно выполнять в PHPMyAdmin три действия:

- Добавлять записи в таблицу.
- Редактировать записи в таблицах.
- Удалять записи из таблицы.

Для того, чтобы *добавить запись в таблицу*, необходимо зайти на главную страницу phpMyAdmin, выбрать из выпадающего списка имя базы данных, где находится искомая таблица, кликнуть по имени таблицы, в которой нужно добавить запись, перейти в пункт «Вставить» и нажать на кнопку «Пошёл».

Для того, чтобы *редактировать запись в таблице*, необходимо зайти на главную страницу phpMyAdmin, выбрать из выпадающего списка имя базы данных, где находится искомая таблица, кликнуть по имени таблицы, в которой нужно отредактировать запись, перейти по ссылке «Обзор», щелкнуть на значок «Карандаша», напротив той записи, которую хотите отредактировать, внести изменения и нажать на кнопку «Пошёл».

Для того, чтобы *удалить записи из таблицы*, необходимо зайти на главную страницу phpMyAdmin, выбрать из выпадающего списка имя базы данных, где находится искомая таблица, кликнуть по имени таблицы, в которой нужно удалить запись, перейти по ссылке «Обзор», щелкнуть на значок «Крестик», напротив той записи, которую хотите удалить, далее подтвердить удаление.

Как сделать индекс в MySQL?

Для первичных ключей (PRIMARY KEY) индекс создаётся автоматически, а вот для других полей последовательность действий в phpMyAdmin следующая: Зайти на главную страницу phpMyAdmin, выбрать из выпадающего списка имя базы данных, где находится требуемая таблица, кликнуть по имени таблицы, в которой нужно создать индекс, щёлкнуть на значок "Молнии" напротив того поля, для которого нужно создать индекс.

Когда надо создавать индексы MySQL

- Если по полю очень часто идёт выборка, то его надо делать индексом.
- Если в таблицу очень часто добавляются записи, и при этом выборка происходит редко (такое иногда бывает), то индексы делать не надо.

Для того, чтобы узнать синтаксис SQL-запроса, первым делом необходимо зайти в phpMyAdmin, затем выбрать какую-нибудь таблицу и просто вставить туда новую запись. После вставки появится окно, как на рисунке 9.2.

Рисунок 9.2 – Окно SQL-запросов

9.3 Аналоги phpMyAdmin

Многие компании создают различные многофункциональные приложения для облегчения управления, разработки и администрирования баз данных.

Большинство реляционных баз данных, за исключением MSAccess, состоят из двух отдельных компонентов: «back-end», где хранятся данные и «front-end» – пользовательский интерфейс для взаимодействия с данными. Этот тип конструкции распараллеливает двухуровневую модель программирования, которая отделяет слой данных от пользовательского интерфейса.

В Интернете каждый может найти много продуктов для разработки и администрирования баз данных MySQL. Опишем 4 бесплатных самых популярных инструментов.

1. Workbench

Что делает Workbench популярным?

- возможность представить модель базы данных в графическом виде, а также редактирование данных в таблице;
- наличие простого и функционального механизма по созданию связей между полями таблиц, среди которых реализована связь «многие-ко-многим» с возможностью создания таблицы связей;
- функция ReverseEngineering позволяет восстанавливать структуру таблиц и связей из той, которая была реализована ранее и хранится на сервере БД;
- наличие редактора SQL-запросов, который дает возможность при отправке на сервер получать ответ в табличном виде и другие возможности.

2. PHPMyAdmin

Что делает PHPMyAdmin популярным?

- возможность управлять СУБД MySQL без непосредственного ввода SQL команд;
- как панель управления PHPMyAdmin предоставляет возможность администрирования выделенных баз данных;
- интенсивное развитие;
- возможность интегрировать PHPMyAdmin в собственные разработки благодаря лицензии GNU GeneralPublicLicense и другие возможности.

3. HeidiSQL

Что делает HeidiSQL популярным?

- возможность подключаться к серверу с помощью командной строки;
- возможность пакетной оптимизации и восстановления таблиц;
- возможность редактирования столбцов, индексов и внешних ключей таблиц, редактирование тела и параметров SQL процедур, триггеров и др.;
- простое форматирование неупорядоченных SQL;

- синхронизация таблицы между разными базами данных и другие возможности.

4. MyDBStudio

Что делает MyDBStudio популярным?

- возможность подключаться к неограниченному количеству баз данных;
- возможность подключения по SSH каналам;
- создание откатов и экспорт БД в различные форматы;
- возможность переноса, резервного копирования, также восстановления базы данных и другие возможности.

Список использованных источников

1. Когаловский, М. Р. Энциклопедии технологий баз данных / М.: Финансы и статистика, 2002. – 800 с.

2. Интуит. Национальный открытый университет [Электронный ресурс]/ Открытый образовательный портал. – Москва, 2008. – Режим доступа: http://www.intuit.ru. –Дата доступа: 20.09.2013.

3. Аткинсон, Л. MySQL. Библиотека профессионала / М.: Издательский дом «Вильяме», 2002. – 624 с.: ил.

4. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript / СПб.: Питер, 2011. – 496 с.: ил.

Тихоненко Татьяна Владимировна

СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Курс лекций по одноименной дисциплине для слушателей специальности 1-40 01 74 «Web-дизайн и компьютерная графика» заочной формы обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 30.03.14.
Рег. № 59Е.
http://www.gstu.by