

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ВЫСШИЙ КОЛЛЕДЖ ИНФОРМАТИКИ

Иваньчева Т. А.

Методическое пособие по языку SQL
(Диалект СУБД MySQL)
Часть 1

Новосибирск
2009

Составитель: Иваньчева Т.А.

МЕТОДИЧЕСКОЕ ПОСОБИЕ ПО ЯЗЫКУ SQL (Диалект СУБД MySQL)

Методическое пособие по языку SQL содержит материал по диалекту языка SQL сервера баз данных `mysql`. С использованием практических примеров и задач рассмотрен язык определения данных, позволяющий создавать объекты баз данных, такие как таблицы, индексы, представления, хранимые процедуры, функции, триггеры. В пособии также рассмотрен язык манипулирования данными, позволяющий выполнять обработку данных. Большое внимание уделено рассмотрению нетривиальных, вложенных запросов.

Данное методическое пособие предназначено для поддержки лекционных и практических занятий по курсу «Базы данных» для студентов 3 курса средне-технического факультета ВКИ НГУ. Пособие может быть также полезно для проведения практических занятий по курсу «Разработка удаленных баз данных», а также может оказать помощь для студентов, выполняющих дипломный проект и использующих в качестве основной СУБД сервер баз данных `MySQL`.

Пособие является результатом обобщения многолетнего опыта подготовки специалистов по технологиям баз данных.

Данный материал опробован на лекционных и практических занятиях по курсу «Базы данных». Данное пособие содержит не только теоретический материал по диалекту языка SQL сервера `MySQL`, но и снабжено многочисленными задачами и практическими заданиями для закрепления этого материала.

Рецензент: Н.А. Аульченко

Ответственный за выпуск: А.А. Юшкова

Оглавление

Глава 1. Введение	5
Информационные системы. Базы данных. СУБД	5
Обзор СУБД	6
Понятие сервера и клиента. Архитектура клиент-сервер	8
Сервер баз данных MySQL	9
Возможности и особенности сервера MySQL	11
Клиентские программы для работы с сервером MySQL	12
Контрольные вопросы	15
Глава 2. Объекты базы данных MySQL.	
Типы данных MySQL	16
Объекты базы данных	16
Идентификация объектов в базе данных	16
Типы таблиц в MySQL	17
Типы данных MySQL	19
Контрольные вопросы и упражнения	24
Глава 3. Базы данных и таблицы	25
Работа с базой данных	25
Работа с таблицами	27
Команда CREATE TABLE	28
Команда ALTER TABLE	36
Команда DROP TABLE	38
Просмотр структуры таблиц	39
Команда DESCRIBE	39
Команда SHOW	39
Контрольные вопросы	40

Глава 4. Работа с данными	41
Команда INSERT	41
Команда UPDATE	43
Команда DELETE	44
Команда SELECT	45
Использование функций в запросах	55
Контрольные вопросы и упражнения	60
Глава 5. Индексы и представления	61
Индексы(Ключи)	61
Создание индекса	62
Просмотр существующих индексов	63
Удаление индекса	64
Восстановление индекса	64
Для каких полей следует создавать индексы	65
Представления (просмотры)	65
Создание представления	66
Удаление представления	69
Просмотр структуры представления	69
Контрольные вопросы и упражнения	70
Литература	71

Глава 1

Введение

Информационные системы.

Базы данных. СУБД

Первоначально компьютер предназначался для решения вычислительных задач. Проблема автоматизации расчетов была очень актуальна. Само название «компьютер» происходит от английского глагола – to compute, что означает – вычислять. С развитием внешних запоминающих устройств стало возможным хранить и обрабатывать с помощью компьютеров большие объемы информации. Компьютерами заинтересовались бизнесмены, банковские служащие, экономисты, медицинские работники, работники других сфер, профессиональная деятельность которых тесно связана с обработкой информации. В настоящее время благодаря огромным возможностям компьютеров, связанным с хранением и обработкой больших массивов информации, компьютер применяется буквально во всех сферах человеческой деятельности: для решения широкого круга задач в экономике, промышленности, менеджменте, медицине, банковской сфере, управлении и т.д.

Задачи, связанные с хранением и обработкой информации принято называть информационными. Таких задач очень много. Это системы резервирования билетов, банковские системы, бухгалтерские и учетные системы, системы планирования и управления, обработка наблюдений и т.д. Результатом решения таких задач является создание информационных систем. **Информационные системы** имеют дело с базами данных. Данные, которые необходимо хранить и обрабатывать, принято называть

базой данных. Можно дать такое определение **база данных** – это набор взаимосвязанных данных о некоторой предметной области, которые имеют определенную структуру и постоянно хранятся в памяти компьютера.

Для управления такими данными было разработано специальное программное обеспечение, которое получило название **СУБД – система управления базами данных**. Само появление СУБД явилось шагом революционным. По значимости его можно сравнить с появлением первых компиляторов. Не сразу стало понятно, что для решения самых разнообразных задач из разных предметных областей необходим некоторый программный слой, который отвечает только за хранение данных на внешних устройствах и позволяет этими данными манипулировать. Но современные СУБД предоставляют пользователям значительно большие возможности.

Система управления базами данных (СУБД) – это комплекс программных и языковых средств, необходимых для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации.

Первые СУБД, поддерживающие организацию и ведение баз данных, появились в конце 60-х годов. Использование СУБД обеспечивает лучшее управление данными, более совершенную организацию файлов и более простое обращение к ним по сравнению с обычными способами хранения информации.

Обзор СУБД

Мир современных СУБД весьма обширен. СУБД можно различать по компаниям-разработчикам, которые их произвели, по моделям данных, которые они используют, по производительности и мощности, по комплексу услуг, которые они предоставляют пользователям.

Рассмотрим классификацию СУБД по возможностям работы

в сети. Соответственно, можно различить так называемые настольные СУБД, которые не рассчитаны для работы в сети и многопользовательские СУБД или серверы баз данных(рис. 1.1.).

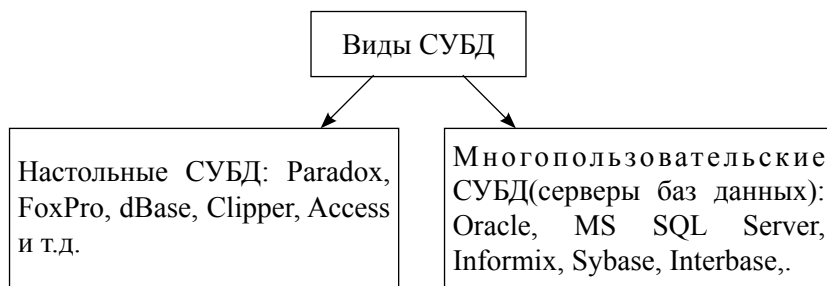


Рис. 1.1.

Настольные СУБД изначально предназначались для работы одного пользователя за одним компьютером. Они могут работать в сети, но при увеличении количества пользователей их производительность резко падает. Это СУБД хорошего качества, предназначенные для работы с небольшими централизованными базами данных. К ним относятся такие СУБД, как Access, ForPro, Paradox и т.д. Средства СУБД Access и FoxPro могут быть использованы для создания клиентской части приложений для серверов баз данных.

МногопользовательскиеСУБДилисерверыбазданныхэтоочень мощные профессиональные СУБД, изначально предназначенные для работы большого количества пользователей в сети с одними и теми же данными. При этом сами данные могут быть распределены по разным узлам сети. К серверам баз данных относятся: Oracle, MS SQL Server, Informix, Interbase, MySQL и т.д. Так, например, СУБД Oracle способна поддерживать одновременно работу до 5000 пользователей, а объем базы данных, обслуживаемой этой СУБД, может достигать нескольких терабайт. Для сравнения максимальный объем базы данных Interbase всего лишь около 20 Гигабайт.

Понятие сервера и клиента. Архитектура клиент-сервер

Под сервером понимается целый компьютер или процесс(т.е. программа), который обслуживает другие компьютеры или процессы, называемые клиентами. В зависимости от вида предоставляемых услуг различают разные виды серверов: Web-сервер, файловый сервер, сервер баз данных, почтовый сервер, вычислительный сервер и т.д.

Таким образом, сервер баз данных это программа, которая предоставляет услуги по работе с базами данных. Сервер баз данных это ядро СУБД, которое постоянно находится в оперативной памяти и выполняет основные функции СУБД по обработке данных. Архитектура сервера баз данных приведена на рисунке 1.2. Клиентские программы с нескольких рабочих станций обращаются к программе сервера за услугами по обработке данных. Языком запросов к серверу является язык SQL.

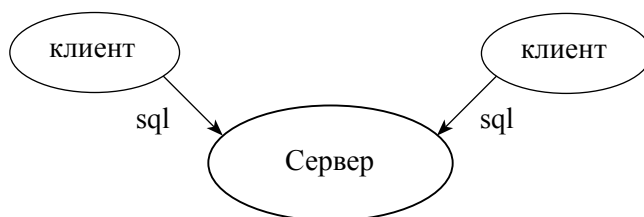


Рис 1.2.

SQL - это язык структурированных запросов (Structured Query Language), международный стандарт языка для доступа к базам данных. Первоначальное название языка SEQUEL – Structured English Query Language. Язык разрабатывался в лаборатории компании IBM с 1975 по 1979 годы и актуален до сих пор.

Особенности языка SQL:

- Язык декларативного типа, т.е. не процедурный язык, позволяющий декларировать, что вы хотите получить.

- В языке нет циклов.
- В языке нет условных операторов.
- Существует множество диалектов языка от разных производителей.

Все множество команд языка SQL делится на две группы: язык определения данных (Data Definition Language - DDL) и язык манипулирования данными (Data Manipulated Language - DML). С помощью команд языка определения данных можно создавать, изменять структуру и удалять объекты базы данных. Поэтому основными командами языка являются команды:

CREATE (создать),
ALTER (изменить),
DROP (удалить).

Команды языка манипулирования данными позволяют работать с данными базы данных, т.е. добавлять, изменять и удалять данные. Основными командами языка являются следующие команды:

INSERT (добавить),
UPDATE (изменить),
DELETE (удалить),
SELECT (выбрать).

Основной задачей данного пособия является рассмотрение возможностей языка запросов сервера MySQL.

Сервер баз данных MySQL

СУБД MySQL была разработана Михаэлем Видениусом (Michael Widenius, monty@analytikerna.se). MySQL является относительно небольшой и быстрой реляционной СУБД основанной на традициях Hughes Technologies Mini SQL (mSQL).

Сайт с документацией по MySQL доступен по ссылкам www.mysql.com и www.mysql.ru.

MySQL - это торговая марка компании MySQL AB, которая

выполняет разработку и сопровождение MySQL, самой популярной SQL-базы данных с открытым кодом.

MySQL AB - коммерческая компания, основанная разработчиками MySQL, строящая свой бизнес, предоставляя различные сервисы для СУБД MySQL.

Программное обеспечение MySQL - это программное обеспечение с открытым кодом. Программное обеспечение (ПО) с открытым кодом означает, что применять и модифицировать его может любой желающий. Такое ПО можно получать по Internet и использовать бесплатно. При этом каждый пользователь может изучить исходный код и изменить его в соответствии со своими потребностями.

Программное обеспечение MySQL имеет двойное лицензирование. Это означает, что пользователи могут выбирать, использовать ли программное обеспечение MySQL бесплатно по общедоступной лицензии GNU General Public License (GPL) или приобрести одну из стандартных коммерческих лицензий MySQL AB. (<http://www.gnu.org/licenses/>).

Использование программного обеспечения MySQL регламентируется лицензией GPL (GNU General Public License), <http://www.gnu.org/licenses/>, в которой указано, что можно и чего нельзя делать с этим программным обеспечением в различных ситуациях. Если работа в рамках GPL вас не устраивает или планируется встраивание MySQL-кода в коммерческое приложение, есть возможность купить коммерческую лицензированную версию у компании MySQL AB.

Самую свежую информацию о программном обеспечении MySQL можно получить на следующих веб-сайтах MySQL:

www.mysql.com и www.mysql.ru/.

Программное обеспечение MySQL (TM) представляет собой очень быстрый многопоточный, многопользовательский надежный SQL-сервер баз данных. Сервер MySQL предназначен как для

критических по задачам производственных систем с большой нагрузкой, так и для встраивания в программное обеспечение массового распространения.

Возможности и особенности сервера MySQL

- Работа на нескольких платформах:

- Обеспечивается транзакционное и не транзакционное ядро.
- Используются быстрые таблицы (MyISAM, InnoDB).
- Быстрые соединения.

- Типы данных:

Большой набор типов данных: знаковые и беззнаковые целые 1, 2, 3, 4, и 8 байт, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM, записи фиксированной и переменной длины.

- Большой набор встроенных функций:

Использование операторов и функций в предложениях SELECT и WHERE.

Например:

```
mysql> SELECT CONCAT (first_name, ' ', last_name)
```

```
-> FROM citizen
```

```
-> WHERE income/dependents > 10000 AND age > 30;
```

- Поддержка GROUP BY и ORDER BY в операторе SELECT.

Поддержка агрегатных функций: (COUNT(), COUNT(DISTINCT ...), AVG(), STD(), SUM(), MAX(), MIN(), и GROUP_CONCAT()).

- Поддержка левых и правых внешних соединений:

LEFT OUTER JOIN и RIGHT OUTER JOIN

- Поддержка алиасов таблиц и столбцов, как требует стандарт SQL.

- операторы DELETE, INSERT, REPLACE и UPDATE возвращают число измененных строк.

- Имена функций не конфликтуют с именами таблиц и столбцов.
Например, ABS может быть не только именем функции, но и правильным именем столбца. Можно ссылаться на таблицы из разных баз данных в одном предложении.
- Масштабируемость и ограничения:
Управление большими базами данных. MySQL Server может использоваться с базами, которые содержат более 50 миллионов записей. Известны пользователи, которые использовали базы данных, содержащие 60000 таблиц и около 5,000,000,000 записей.
- Для каждой таблицы разрешено использовать до 64 индексов.
Каждый индекс может содержать от 1 до 16 столбцов. Максимальный размер индекса 1000 байт (767 для таблиц InnoDB). Индекс может использовать префиксы для полей типа CHAR, VARCHAR, BLOB, или TEXT.
- Возможность создания внешних ключей для таблиц InnoDB.
- Использование просмотров (представлений).
- Использование хранимых процедур и функций.
- Использование триггеров.

Последние версии сервера MySQL(v. 5.1 и v. 6.0) по своим функциональным возможностям мало чем отличаются от современных мощных серверов, таких как Microsoft SQL Server, Oracle, Interbase и т.д.

Клиентские программы для работы с сервером MySQL

В качестве клиентских приложений для работы с сервером MySQL используются следующие программы:

phpmyAdmin – Web-приложение, обеспечивающее удобный пользовательский интерфейс для работы с данными.

mysql – консольное приложение, которое в режиме командной строки позволяет вводить команды SQL для передачи их серверу.

phpmyAdmin - некоммерческое приложение, написанное на языке PHP, реализующее удобный и функциональный Web-интерфейс к базе данных MySQL. Данный продукт является Open-Source (продуктом с открытым кодом) и распространяется в соответствии со Стандартной общественной лицензией GNU. Скачать программу можно с сайта: <http://php-myadmin.ru>.

В данный момент phpMyAdmin позволяет:

- создавать и удалять базы данных;
- создавать, копировать, удалять, переименовывать и изменять таблицы;
- осуществлять сопровождение таблиц;
- удалять, править и добавлять поля;
- выполнять SQL-запросы, в том числе пакетные SQL-запросы;
- управлять ключами;
- загружать текстовые файлы в таблицы;
- создавать и просматривать дампы таблиц;
- экспортировать данные в форматах CSV, XML, PDF, ISO/IEC 26300 - OpenDocument Text and Spreadsheet, Word, Excel и LATEX;
- выполнять администрирование нескольких серверов;
- управлять пользователями MySQL и привилегиями;
- проверять целостность ссылочных данных в таблицах MyISAM;
- использовать запрос по образцу (Query-by-example - QBE), создавать комплексные запросы, автоматически соединяясь с указанными таблицами;
- создавать графическую схему базы данных в формате PDF;
- осуществлять поиск в базе данных или в её разделах;
- модифицировать хранимые данные в различные форматы, используемые в предустановленных функциях, например, отображение BLOB-данных как изображений или как загружаемые ссылки и т.д.;

- поддерживает InnoDB таблицы и внешние ключи;
- поддерживать mysql, улучшенное расширение MySQL ;
- переведен более чем на 50 языков.

Окно программы phpMyAdmin приведено на рисунке 1.3.:

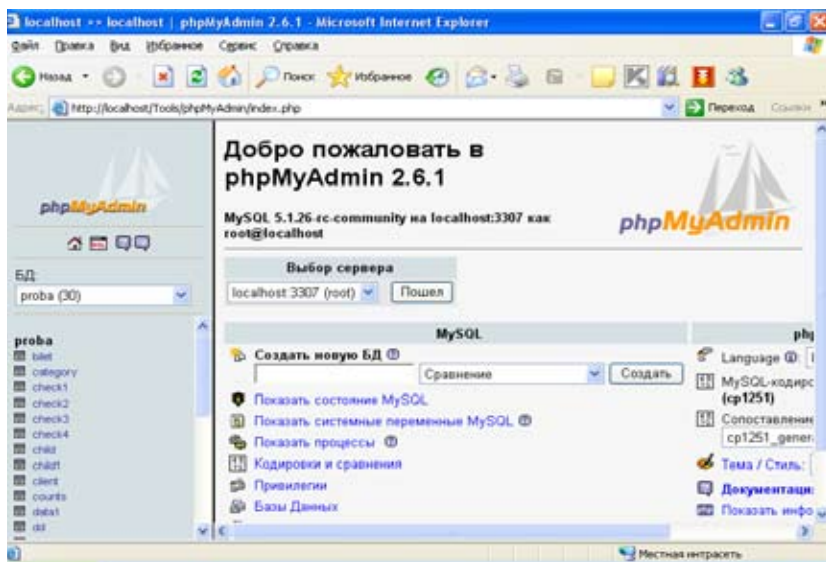


Рис. 1.3.

Утилиту **mysql** часто называют «терминальным монитором» или просто «монитором».

Запуск консольного приложения выполняется из командной строки Windows следующим образом:

```
> mysql -u <User> -h <Host> -P <Port> -p<Password>
```

Где ключи обозначают следующее:

u – имя пользователя

h – имя хоста

P – номер порта

p – пароль.

Например:

```
>mysql -u root -h ntv.dom201.local -P 3307 -p12345
```

Окно консольного приложения mysql приведено на рисунке 1.4.:

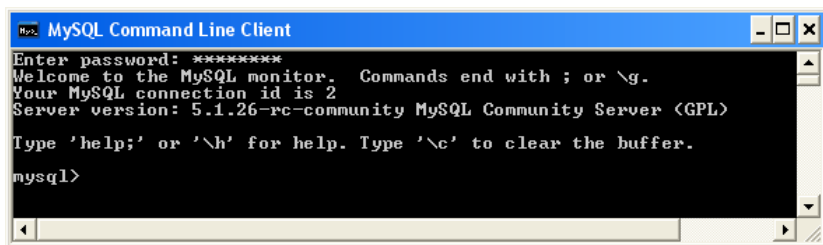


Рис. 1.4.

Контрольные вопросы

1. Что такое информационная система?
2. Что такое база данных?
3. Для чего используется СУБД?
4. Что такое сервер? Что такое клиент? Может ли одна и та же программа выступать как в роли клиента, так и в роли сервера?
5. Какие возможности предоставляет пользователю настольная СУБД?
6. Какие возможности предоставляют серверы базы данных?
7. Особенности языка SQL.
8. Возможности языка определения данных. Основные команды.
9. Возможности языка манипулирования данными. Основные команды.
10. Основные возможности сервера MySQL.
11. Какие клиентские программы для доступа к серверу MySQL Вам известны? Перечислите их основные возможности. Как ими пользоваться?

Глава 2

Объекты базы данных MySQL.

Типы данных MySQL

Объекты базы данных

Каждая база данных MySQL создается в отдельной папке, причем имя папки совпадает с именем базы данных. В базе данных MySQL могут быть созданы следующие основные объекты:

Таблицы (Table), именно в таблицах хранятся данные.

Представления (просмотры) (View) – объекты базы данных, которые позволяют ограничить доступ пользователя подмножеством столбцов и строк одной или нескольких таблиц.

Индексы (Index) – дополнительные структуры, благодаря которым обеспечивается быстрый индексный поиск данных.

Хранимые процедуры (Procedure) и **хранимые функции** (Function) – это программный код, которые хранятся в базе данных вместе с данными.

Триггеры (Trigger) – это хранимые процедуры, которые срабатывают автоматически при наступлении определенных событий при работе с данными: добавление, удаление или изменение.

Идентификация объектов в базе данных

В СУБД MySQL каждая база данных хранится в отдельной папке, имя папки определяет имя базы. Каждая таблица хранится в отдельном файле. Имя может содержать только латинские буквы и цифры, состоящие из символов кодировки, заданной при создании

базы данных, а также символов «_» и «\$». Имя может начинаться с любого допустимого символа, включая цифры. Однако имя столбца не должно состоять из одних цифр: это сделает его неотличимым от чисел.

Идентификаторы можно заключать в обратные кавычки (`), что позволяет использовать в именах любые символы, включая ключевые слова и пробелы.

Для имен баз данных и таблиц нельзя использовать точку и разделители, применяемые в именах пути доступа операционных систем (“\”, “/”), т.к. каждая база данных MySQL хранится в отдельной папке, а каждая таблица в файле.

Псевдонимы таблиц и полей могут быть произвольными. Они берутся в обратные кавычки, если совпадают с ключевым словом.

Длина идентификатора меньше 64 символов. Идентификаторы хранятся в кодировке utf8(Unicode), символы представляются от одного до 3-х байт. Кодировку можно задавать на уровне базы данных, таблицы, столбца.

Имена функций, ключевые слова и операторы не чувствительны к регистру букв. Имена баз данных и таблиц подчиняются правилам именования каталогов операционной системы.

Типы таблиц в MySQL

СУБД MySQL поддерживает несколько типов таблиц: ISAM, MyISAM, MERGE, HEAP, InnoDB.

Таблицы ISAM являются устаревшими. Сейчас используются таблицы **MyISAM(они используются по умолчанию)**. Они позволяют:

- Поддерживать большой объем таблиц.
- Содержимое таблицы хранится в формате, не зависящем от платформы.

- Более эффективно работать с индексами и атрибутом `auto-increment`.
- Более эффективно организована поддержка целостности таблицы.
- Поддерживается полнотекстовый поиск с использованием индекса `FULLTEXT`.

Таблицы **MERGE** – предназначены для объединения нескольких таблиц в одну. Причем с помощью одного запроса можно обращаться ко всем таблицам, входящим в состав единой таблицы.

Таблицы **HEAP** – это временные таблицы, предназначенные для хранения в оперативной памяти. Для повышения эффективности в них применяют только строки фиксированной длины.

Таблицы **BDB** – поддерживаются дескриптором Berkeley DB, разработанный компанией Sleerucat. Дескриптор обеспечивает:

- Обработку таблиц с использованием транзакций.
- Автоматическое восстановление после сбоев.
- Блокирование на уровне страниц, обеспечивающее хорошую производительность при обработке параллельных запросов.

Таблицы **InnoDB** – самые новые таблицы недавно введенные в MySQL. Этот тип поддерживается дескриптором InnoDB, разработанным компанией Innobase Oy. Обеспечивает:

- Обработку таблиц с использованием транзакций.
- Автоматическое восстановление после сбоев.
- Поддержка ключей, в том числе и каскадного удаления.
- Блокирование на уровне страниц, обеспечивающее хорошую производительность при обработке параллельных запросов.

Таблицы могут быть распределены по нескольким файлам.

Типы данных MySQL

Тип данных задает множество правильных значений переменной и множество операций, которые над ней можно выполнять. Например, над числами разрешены все арифметические операции, над строками эти операции не имеют смысла. Каждый столбец таблицы определяется на своем типе, исходя из тех данных, которые предполагается хранить в этом столбце.

СУБД MySQL обеспечивает следующие типы данных.

Целые числа

Тип	Описание
TINYINT[(length)] [UNSIGNED] [ZEROFILL]	1 байт
SMALLINT[(length)] [UNSIGNED] [ZEROFILL]	2 байта
MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]	3 байта
INT[(length)] [UNSIGNED] [ZEROFILL]	4 байта
INTEGER [(length)] [UNSIGNED] [ZEROFILL]	4 байта
BIGINT[(length)] [UNSIGNED] [ZEROFILL]	8 байт целое

Длина поля length определяет, сколько всего цифр может иметь число.

ZEROFILL означает, что значение дополняется слева нулями до максимальной длины поля.

UNSIGNED – означает беззнаковое число.

Атрибут AUTO_INCREMENT при определении столбца можно использовать только в одном поле таблицы. Обратите внимание также, что это поле должно быть объявлено как первичный ключ, и должно быть числовым.

ПРИМЕР:

INT(5) ZEROFILL; значение 5 превращается в «00005»

Вещественные числа

Вещественные или дробные числа в общем виде записываются так:

ИмяТипа[(length, dec)] [UNSIGNED]

Где length – количество знакомест(ширина поля), в которых будет размещено дробное число, dec – количество знаков после точки.

Тип	Описание
FLOAT [(precision)]	Число с плавающей запятой. FLOAT(4) и FLOAT одиночная точность. FLOAT(8) обеспечивает двойную точность.
FLOAT [(length,dec)]	Число одиночной точности с максимальной длиной и фиксированным числом десятичных чисел (4 байта).
DOUBLE	Числосплавающейзапятой,обеспечивает двойную точность.
REAL	Синоним для DOUBLE
DECIMAL	Дробное число, хранящееся в виде строки

Например: FLOAT(9, 2) – вещественное число, состоящее из 9 цифр и имеющее 2 цифры после точки.

Строки

Тип	Описание
CHAR(NUM)	Строки фиксированной длины 65 535 символов
VARCHAR(NUM)	Строка переменной длины 65 535 символов
TINYTEXT	Может хранить максимум 255 символов
TEXT	Может хранить максимум 65 535 символов
MEDIUMTEXT	Может хранить максимум 16 777 215
LONGTEXT	Может хранить максимум 4 294 967 295 символов

Строковый тип данных позволяет хранить текстовую информацию. Где NUM при определении типа означает длину строки. Чаще всего применяется тип TEXT. Обычно при поиске по текстовым полям по запросу SELECT не берется в рассмотрение регистр символов, т.е. строки «ФЕДЯ» и «Федя» считаются одинаковыми. Однако если указан флаг BINARY, то при запросе SELECT строка будет сравниваться с учетом регистра. Например:

VARCHAR(50) BINARY.

Строка может заключаться либо в одинарные кавычки (') либо в двойные кавычки («). Примеры правильных строк:

- 'hello'
- «hello»
- '»»hello»»'
- «'ello»
- «'e»l»lo»
- 'hello'
- «This\nIs\nFour\nlines» .

Апострофы, если они используются внутри строки, удваиваются. Например, см. рис 2.1:

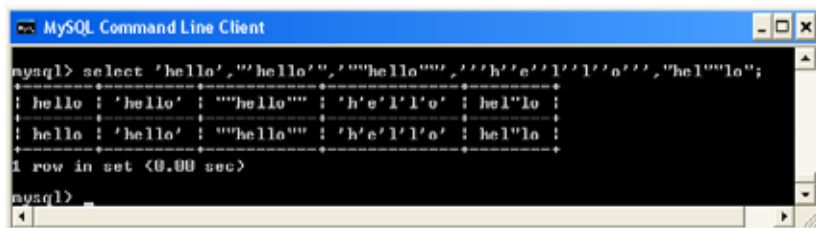


Рис. 2.1.

Бинарные данные (BLOB)

Тип	Описание
TINYBLOB	Двоичный объект с максимальной длиной 255 байт.
BLOB	Двоичный объект (максимальная длина 65535 байт)
MEDIUMBLOB	Двоичный объект с максимальной длиной 16777216 байт.
LOB	Двоичный объект с максимальной длиной 2^{32} байт.

Тип данных BLOB (Binary Large Object) используется для хранения рисунков, фотографий, графиков и других двоичных объектов в базе данных.

Дата и время

Тип	Объем памяти в байтах	Описание
DATE	3	Дата в формате ГГГГ-ММ-ДД
TIME	3	Время в формате ЧЧ:ММ:СС
DATETIME	8	Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС
TIMESTAMP	4	Дата и время в формате Unix timestamp. Однако при получении значения поля оно отображается не в формате timestamp, а в виде ГГГГММДДЧЧММСС, что сильно умаляет преимущества его использования в РНР.
YEAR[(M)]	1	От 1901 до 2155 для YEAR(4) От 1970 до 2069 для YEAR(2)

Этот тип данных используется для хранения данных о дате и времени. Столбцы YEAR предназначены для хранения только года, указание параметра M позволяет задать формат года. Так, для YEAR(2) год представляется в виде двух цифр, например 05 или 97, с диапазоном данных от 1970 до 2069. А тип YEAR(4) позволяет задать тип в виде 4 цифр YYYY, например 2005 или 1997, с диапазоном от 1901 до 2155 года. Если параметр не указан, то по умолчанию он считается равным 4.

Надо заметить, что в некоторых случаях в PHP будет проще самостоятельно генерировать дату и время при вставке данных в таблицу, а не задействовать встроенные в MySQL типы.

Перечисления(ENUM)

При использовании этого типа данных значением столбца может быть только одно значение из заданного списка. Примеры:

```
ENUM('Y','N')
```

```
ENUM('show','hide')
```

Определение столбца задано так:

```
'hide' enum('show','hide') NOT NULL default 'show';
```

- здесь столбец с именем hide (т.к. его имя совпадает с ключевым словом, поэтому имя взято в обратные кавычки) может иметь только одно значение из заданного списка 'show','hide'. Значением по умолчанию для этого столбца является значение 'show'.

Множества(SET)

Значением столбца может быть одно или несколько значений из заданного списка. Например, поле f1 определено следующим образом:

```
f1 SET('a','b','c','d') default ('a,b,c');
```

Значением этого поля может быть как 'a' и 'a','b', так и 'a','b','c' и 'a','b','c','d'.

Контрольные вопросы и упражнения

1. Какие объекты могут присутствовать в базе данных MySQL?
Охарактеризуйте каждый объект.
2. Какие виды таблиц поддерживает сервер MySQL?
Охарактеризуйте каждый вид?
3. Правила идентификации объектов MySQL. Какие имена считаются правильными?
4. Что задает тип данных? Какие типы данных MySQL Вы знаете?
5. Поле «Зарплата» хранит информацию о заработной плате сотрудников организации. Какой тип данных можно использовать для описания этого поля?
6. Поле «Телефон» хранит информацию о телефонах студентов и может содержать информацию следующего вида: 8-913-222-56-78. Какой тип данных можно использовать для описания этого поля?
7. Поле «Оценка» хранит информацию об оценках студентов. Оценка может принимать только целочисленные значения от 1 до 5. Какой тип данных можно использовать для описания этого поля?
8. Поле «Пол» хранит информацию о половой принадлежности студента, и может содержать информацию только следующего вида: «женский» или «мужской». Какой тип данных можно использовать для описания этого поля?

Глава 3

Базы данных и таблицы

Работа с базой данных

Все объекты базы данных: таблицы, индексы, представления, процедуры, триггеры, все данные хранятся в базе данных. Поэтому первое, что нужно сделать при работе с данными, это создать базу данных.

В СУБД MySQL создание базы данных сводится к созданию нового подкаталога в каталоге данных. Для операционной системы Windows это каталог – `usr\local\mysql5\data`, если считать от корневого каталога вашего Web-сервера.

Создание базы данных средствами SQL выполняется с помощью команды `CREATE DATABASE <ИмяБазыДанных>`.

После выполнения этого запроса заглянув в каталог `usr\local\mysql5\data` можно увидеть новый каталог с именем, указанным в операторе `CREATE DATABASE`.

При создании базы данных можно указать кодировку, которая будет назначаться таблицам и столбцам по умолчанию. Для этого после имени базы данных необходимо указать конструкцию `DEFAULT CHARACTER SET <НазваниеКодировки>`. Для русских букв следует назначать кодировку `cp1251`, которая обозначает русскую Windows-кодировку.

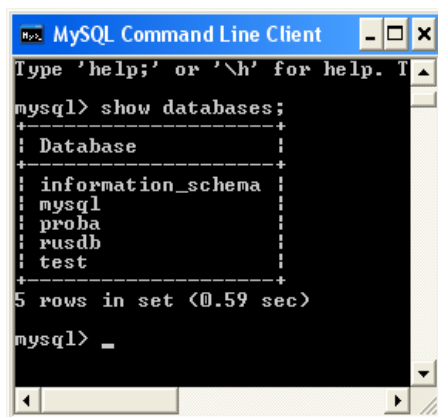
Для изменения параметров базы данных можно воспользоваться оператором `ALTER DATABASE`. Например:

```
ALTER DATABASE Proba CHARACTER SET cp1251;
```

Проконтролировать создание базы данных, а также узнать

имена существующих баз данных, можно при помощи оператора SHOW DATABASES. Пример использования приведен на рис. 3.1

Как видно из рисунка 3.1 оператор SHOW DATABASES вернул имена пяти баз данных. Базы данных information_schema и mysql являются служебными и необходимы для поддержания сервера MySQL в работоспособном состоянии, в них хранится информация об учетных записях, региональных настройках и т.п.

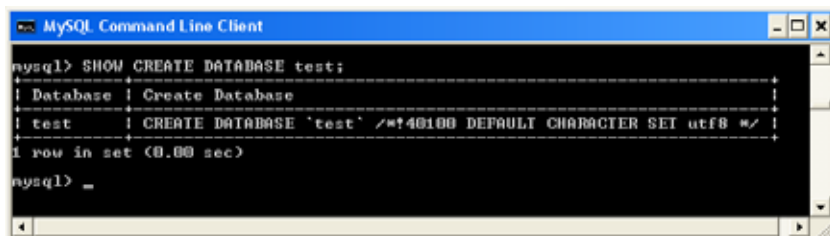


```
MySQL Command Line Client
Type 'help;' or '\h' for help. T
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| proba |
| rusdb |
| test |
+-----+
5 rows in set (0.59 sec)

mysql> _
```

Рис. 3.1.

Оператор SHOW CREATE DATABASE позволяет просмотреть структуру конкретной базы данных. В качестве параметра команде передается имя базы данных. Пример использования приведен на рисунке 3.2:



```
MySQL Command Line Client
mysql> SHOW CREATE DATABASE test;
+-----+-----+
| Database | Create Database |
+-----+-----+
| test | CREATE DATABASE `test` /*!40100 DEFAULT CHARACTER SET utf8 */ |
+-----+-----+
1 row in set (0.00 sec)

mysql> _
```

Рис 3.2.

Попробуйте создать в папке `usr\local\mysql5\data` новый каталог и выполнить после этого запрос `SHOW DATABASES` – созданная таким образом база данных будет отображена в списке баз. Удаление каталога приведет к исчезновению базы данных.

Удаление базы данных можно осуществить также с помощью оператора `DROP DATABASE`, за которым следует имя базы данных. Например:

`DROP DATABASE Proba;`

При попытке создать базу данных, которая уже существует, будет выдана ошибка. Для предотвращения такой ошибки оператор `CREATE DATABASE` необходимо снабдить конструкцией `IF NOT EXISTS`, при наличии которой база данных создается, если она еще не существует, если же существует, то никакие действия не производятся.

Для переименования базы данных используется оператор `RENAME DATABASE`, который имеет следующий синтаксис:

`RENAME DATABASE <СтароеИмя> TO <Новое Имя>;`

Приступая к работе с базами данных необходимо выбрать ту базу, с которой будет осуществляться обращение. Каждая клиентская программа решает эту задачу по-своему. Например, в консольном клиенте `mysql` выбрать базы данных можно с помощью оператора `USE <имя БД>`. Например:

`USE Proba;`

Если база данных не выбрана, то будет выдаваться сообщение об ошибке: «No database selected».

Работа с таблицами

При работе с таблицами на языке SQL используются следующие основные команды:

- `CREATE TABLE` – позволяет создать таблицу;

- ALTER TABLE – позволяет изменить структуру таблицы;
- DROP TABLE – позволяет удалить таблицу.

Рассмотрим синтаксис основных команд языка SQL. При описании синтаксиса квадратные скобки ([...]) означают необязательные опции. Фигурные скобки ({...}) – выбор из нескольких вариантов, при этом разные варианты отделяются друг от друга с помощью вертикальной черты. Например:

[IF NOT EXISTS] – здесь квадратные скобки означают, что использование IF NOT EXISTS не является обязательным.

{INDEX|KEY} – здесь фигурные скобки означают использование либо ключевого слова INDEX, либо ключевого слова KEY.

Команда CREATE TABLE

Команда CREATE TABLE позволяет создать таблицу в базе данных и имеет следующий синтаксис:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] *ИмяТаблицы*
 (*Определение*,...)
 [*ОпцииТаблицы*] ...

Определение:

ИмяАтрибута *ОпределениеАтрибута*
 | PRIMARY KEY (*ИмяАтрибута* ,...)
 | {INDEX|KEY} (*ИмяАтрибута* ,...)
 | [CONSTRAINT [*Символ*]] UNIQUE [INDEX|KEY]
 (*ИмяАтрибута* ,...)
 | {FULLTEXT|SPATIAL} [INDEX|KEY] [*ИмяИндекса*]
 (*ОпредСтолбцаВИндексе*,...)
 | [CONSTRAINT [*Символ*]] FOREIGN KEY
 [*ИмяИндекса*] (*Опред СтолбцаВИндексе*,...)
 ОпределениеВнешнегоКлюча

ОпределениеАтрибута:

ТипДанных [NOT NULL | NULL]

[DEFAULT *ЗначениеПоУмолчанию*]
[AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
[COMMENT 'Строка'] [*ОпределениеВнешнегоКлюча*]

ОпределениеВнешнего Ключа:

[CONSTRAINT [*Символ*]] FOREIGN KEY
[*index_name*] (*ОпредСтолбцаВИндеkse, ...*)
REFERENCES *Имя Таблицы* (*Опред СтолбцаВИндеkse,...*)
[ON DELETE *ОпцииСсылки*]
[ON UPDATE *ОпцииСсылки*]

ОпредСтолбцаВИндеkse:

ИмяСтолбца [(*Длина*)] [ASC | DESC]

ОпцииСсылки:

RESTRICT | CASCADE | SET NULL | NO ACTION

ОпцииТаблицы:

ENGINE [=] *Тип Таблицы*
| AUTO_INCREMENT [=] *Значение*

Описание синтаксиса:

Использование ключевого слова TEMPORARY позволяет создать временную таблицу. Временная таблица автоматически удаляется по завершении соединения с сервером, а ее имя действительно только в течение данного соединения. Это означает, что два разных клиента могут использовать временные таблицы с одинаковыми именами без конфликтов друг с другом.

Повторное создание таблицы без конструкции IF NOT EXISTS приводит к возникновению ошибки: "Table ... already exists" (таблица уже существует).

После ключевых слов CREATE TABLE необходимо указать имя таблицы, созданное по правилам идентификации, принятым в СУБД MySQL. Далее, в круглых скобках следует перечень

описаний определений. Определение может содержать:

- Описание атрибута, при этом указывает имя атрибута и его тип. В качестве типа можно использовать любой допустимый тип MySQL.
- Определение первичного ключа(PRIMARY KEY). При этом в скобках указывается список атрибутов, входящих в состав первичного ключа.
- Определение индекса (INDEX или KEY), при этом указывается список атрибутов, из которых состоит индекс.
- Определение ограничения (CONSTRAINT) уникального индекса UNIQUE [INDEX|KEY]. Ключевое слово CONSTRAINT позволяет задать необязательное имя ограничения. Если имя ограничения не задается, то ограничению присваивается системное имя.
- Определение полнотекстового индекса (FULLTEXT).
- Определение внешнего ключа (FOREIGN KEY).

Рассмотрим более подробно определение атрибута:

Определение Атрибута:

ИмяАтрибута ТипДанных [NOT NULL | NULL]
[DEFAULT *ЗначениеПоУмолчанию*]
[AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
[COMMENT '*Строка*'] [*ОпределениеВнешнегоКлюча*]

Каждый атрибут имеет имя, созданное по правилам идентификации, принятым в СУБД MySQL. Атрибут определяется на типе данных. В качестве типа можно использовать любой допустимый тип данных MySQL.

NOT NULL (не пусто) означает, что атрибут обязан иметь значение.

Ключевое слово DEFAULT задает значение по умолчанию для атрибута, это означает, если при добавлении новой строки к таблице не задается значение атрибута, ему будет присвоено

значение по-умолчанию.

Ключевое слово `AUTO_INCREMENT` задает поле типа «счетчик», которое при добавлении новой записи автоматически увеличивают свое значение на 1. `AUTO_INCREMENT` разрешается использовать только в одном поле таблицы. Это поле должно быть объявлено как первичный ключ, и должно быть числовым.

Ключевое слово `UNIQUE` задает уникальный индекс для данного атрибута, который не позволяет вводить в поле повторяющиеся значения.

Ключевые слова `PRIMARY KEY` определяет атрибут, как первичный ключ.

Далее, рассмотрим более подробно определение внешнего ключа:

Определение Внешнего ключа:

```
[CONSTRAINT [Символ]] FOREIGN KEY  
[ИмяИндекса] (ОпредСтолбцаВИндексе, ...)  
REFERENCES ИмяТаблицы (Опред СтолбцаВИндексе,...)  
[ON DELETE ОпцииСсылки]  
[ON UPDATE ОпцииСсылки]
```

ОпцииСсылки:

```
RESTRICT | CASCADE | SET NULL | NO ACTION
```

Ключевое слово `CONSTRAINT` задает имя ограничения. Используя это имя можно, например, удалить ограничение по его имени.

Ключевые слова `FOREIGN KEY` определяют атрибут как внешний ключ. В скобках указывается список имен столбцов через запятую, которые входят в состав внешнего ключа.

`[ИмяИндекса]` задает необязательный индекс, который может быть создан для внешнего ключа.

Ключевое слово `REFERENCES` задает имя таблицы, на которую

ссылается внешний ключ. В скобках указывается столбцы таблицы, на которые ссылается внешний ключ.

[ON DELETE *ОпцииСсылки*] задает, что будет происходить при удалении строки, на которую указывает ссылка.

[ON UPDATE *ОпцииСсылки*] задает, что будет происходить при обновлении строки, на которую указывает ссылка.

Опции ссылки:

CASCADE означает каскадное удаление или обновление, т.е. при попытке удалить или обновить запись таблицы, на которую указывает ссылка, автоматически удаляются (или обновляются) все записи из ссылающегося отношения, которые на нее указывают.

SET NULL означает, что при удалении или обновлении записи родительской таблицы (на которую ссылаются) значение внешнего ключа в дочерней таблице устанавливается равным NULL.

NO ACTION означает, что удаление или обновление записи в родительской таблице запрещено.

RESTRICT синоним NO ACTION.

Внешние ключи можно создавать только для таблиц типа InnoDB.

Рассмотрим отдельно опции таблицы:

ОпцииТаблицы:

ENGINE [=] *ТипТаблицы*

| AUTO_INCREMENT [=] *Значение*

Ключевое слово ENGINE задает тип таблицы.

Ключевое слово AUTO_INCREMENT задает начальное значение для счетчика.

В MySQL все поля имеют неявное значение по умолчанию, если объявлены, как не пустые (NOT NULL). Для создания таблицы необходимо иметь права доступа *create*.

Замечания:

- Столбцы, определенные как BLOB, не могут быть ключами. Нельзя группировать по BLOB. Однако, можно использовать строковые функции MySQL, чтобы группировать на подразделах BLOB.
- Теперь можно использовать BLOB-столбцы в предложении WHERE.
- При операциях INSERT/UPDATE все строки (CHAR и VARCHAR) приводятся к максимальной длине, заданной в операторе CREATE. Все хвостовые пробелы автоматически удаляются. Например, VARCHAR(10) задает, что столбец может содержать строки с длиной до 10 символов.

Таким образом, исходя из синтаксиса, можно сделать следующие выводы:

- Первичный ключ, состоящий из одного поля, может быть определен на уровне столбца, в то время как составной первичный ключ, т.е. состоящий из нескольких полей, можно создать только на уровне таблицы.
- Внешний ключ может быть создан только на уровне таблицы, т.е. после описания всех атрибутов.

Примеры

1. Использование первичных и внешних ключей. С помощью оператора CREATE TABLE создаются две таблицы Отделы (otdel) и Сотрудники (sotr). Поле Nom из таблицы sotr является внешним ключом и ссылается на первичный ключ таблицы Отделы (otdel). При определении внешнего ключа задается каскадное удаление:

```
CREATE TABLE otdel (
    nomer INT(11) NOT NULL AUTO_INCREMENT
    PRIMARY KEY,
    nazv CHAR(50) DEFAULT NULL,
) ENGINE=InnoDB;
```

```
CREATE TABLE sotr (TabNom INTEGER NOT NULL
    AUTO_INCREMENT PRIMARY KEY,
    FIO CHAR(40),
    Zarpl DECIMAL(9,2),
    Nom INTEGER,
    FOREIGN KEY(Nom)
    REFERENCES otdel(nomer) ON DELETE CASCADE)
ENGINE=InnoDB;
```

2. Использование индексированных столбцов. При создании таблицы Материалы (*materials*) с помощью оператора CREATE TABLE создаются два индекса: index1 создается по двум полям id и name, индекс index2 создается для поля name.

```
CREATE TABLE materials ( id INT NOT NULL,
    name CHAR(50) NOT NULL,
    resistance INT,
    melting_pt REAL,
    INDEX index1 (id, name),
    UNIQUE INDEX index2 (name))
```

Задание 1

Написать SQL-операторы, позволяющие создать следующие таблицы. Для каждого поля таблицы назначить правильные типы данных.

1. Tem(*N_t(номер темы), Nazv_t(название темы))
2. Books(*N_b(номер книги), Nazv(название), Avtor(автор),

Cena(цена), God_izd(год издания), N_tem(номер темы))

3. Readers(*N_r(номер читателя), Fio(ФИО), Adres(Адрес), Tel(телефон), pol(пол))

4. Registr(*N_b(номер книги), *N_r(номер читателя), *Data_v(Дата выдачи), Priznak(признак возврата)).

Где первичные ключи отмечены звездочкой(*).

В таблице Registr (Регистрация выдачи/возврата книг) составной первичный ключ, в состав которого входят три атрибута: (*N_b(номер книги), *N_r(номер читателя), *Data_v(Дата выдачи)).

В таблице Books атрибут N_tem(номер темы) является внешним ключом и ссылается на первичный ключ таблицы Tem(Темы).

В таблице Registr два внешних ключа: N_b(номер книги) ссылается на первичный ключ таблицы Books, N_r(номер читателя) ссылается на первичный ключ таблицы Readers.

Задание 2

Написать SQL-операторы, позволяющие создать следующие таблицы. Для каждого поля таблицы назначить правильные имена и правильные типы данных.

1. Абоненты (номер телефона, ФИО, Адрес), где
“номер телефона” - первичный ключ.

2. Тариф (код города, Название города, цена за минуту), где поле “код города” - первичный ключ

3. Разговоры (номер телефона, дата, код города, время), где
«номер телефона», «дата» - составной первичный ключ;

«номер телефона» - внешний ключ, который ссылается на первичный ключ таблицы «Абоненты». Задать каскадное удаление строк таблицы «Разговоры» в случае удаления соответствующей строки таблицы «Абоненты»;

поле «код города» - внешний ключ, ссылающийся на первичный

ключ таблицы «Тариф».

Команда ALTER TABLE

Команда ALTER TABLE позволяет изменить структуру таблицы и имеет следующий синтаксис:

ALTER TABLE *ИмяТаблицы*

КакИзменить [, *КакИзменить*] ...

КакИзменить:

| ADD [COLUMN] (*ИмяСтолбца* *ОпределениеСтолбца*,...)
| [FIRST|AFTER *ИмяСтолбца*]

| ADD {INDEX|KEY} *ИмяИндекса* (*ИмяСтолбцаВИндексе*,...)

| ADD [CONSTRAINT [*Имя1*]] PRIMARY KEY
| (*ИмяСтолбцаВИндексе*,...)

| ADD [CONSTRAINT [*Имя2*]]
| UNIQUE [INDEX|KEY]

| *ИмяИндекса*(*ИмяСтолбцаВИндексе*,...)

| ADD [FULLTEXT|SPATIAL] [INDEX|KEY]
| *ИмяИндекса*(*ИмяСтолбцаВИндексе*,...)

| ADD [CONSTRAINT [*Имя3*]]
| FOREIGN KEY [*Имя4*] (*ИмяСтолбца*,...)
| *ОпределениеСсылки*

| ALTER [COLUMN] *ИмяСтолбца*
| {SET DEFAULT *Литерал* | DROP DEFAULT}

| CHANGE[COLUMN] *СтароеИмяСтолбца* *НовоеИмяСтолбца*
| *ОпределениеСтолбца* [FIRST|AFTER *ИмяСтолбца*]

| MODIFY [COLUMN] *ИмяСтолбца* *ОпределениеСтолбца*
| [FIRST | AFTER *ИмяСтолбца*]

| DROP [COLUMN] *ИмяСтолбца*

| DROP PRIMARY KEY

| DROP {INDEX|KEY} *ИмяИндекса*

| DROP FOREIGN KEY *Имя5*

| RENAME [TO] *НовоеИмяТаблицы*

ИмяСтолбцаВИндексе:

ИмяСтолбца [(length)] [ASC | DESC]

Таким образом, команда позволяет:

ADD [COLUMN] – добавить столбец;

ADD {INDEX|KEY} – добавить индекс;

ADD [CONSTRAINT [*Имя1*]] PRIMARY KEY – добавить ограничение первичного ключа, при этом *Имя1* – это имя ограничения. По умолчанию система сама создает системное имя ограничения;

ADD [CONSTRAINT [*Имя2*]] FOREIGN KEY [*Имя3*]
(*ИмяСтолбца*,...) *ОпределениеСсылки* – Добавить ограничение внешнего ключа, где *Имя2* и *Имя3* – имена ограничений;

ALTER [COLUMN] *ИмяСтолбца* {SET DEFAULT *Литерал* | DROP DEFAULT} – Задать новое значение по умолчанию для столбца или удалить значение по умолчанию для столбца;

CHANGE [COLUMN] *СтароеИмяСтолбца НовоеИмяСтолбца*
ОпределениеСтолбца [FIRST|AFTER *ИмяСтолбца*] - изменить имя и определение столбца, а также указать имя столбца, перед или после которого будет следовать данный столбец;

MODIFY [COLUMN] *ИмяСтолбца* *ОпределениеСтолбца* – изменить определение столбца;

DROP [COLUMN] *ИмяСтолбца* – Удалить столбец;

DROP PRIMARY KEY – Удалить первичный ключ;

DROP {INDEX|KEY} *ИмяИндекса* – Удалить индекс по имени;

DROP FOREIGN KEY *Имя4* – Удалить ограничение внешнего ключа по имени ограничения;

RENAME [TO] *НовоеИмяТаблицы* – переименовать таблицу;

Примеры

1. CREATE TABLE Otdel (Nomer SMALLINT, Nazv CHAR(50))
ENGINE=InnoDB; - оператор позволяет Создать таблицу ОТДЕЛЫ
с полями Номер_отдела , Название.

2. ALTER TABLE Otdel ADD PRIMARY KEY (Nomer); - оператор позволяет добавить определение первичного ключа для атрибута Номер_отдела.

3. ALTER TABLE Otdel ADD Nom_Nach SMALLINT; - оператор позволяет добавить новый атрибут Номер_Начальника.

4. CREATE TABLE Sotr (Tab_Nom SMALLINT NOT NULL
PRIMARY KEY AUTO_INCREMENT,
Fio CHAR(30),
Nom_Otd SMALLINT,
FOREIGN KEY(Nom_Otd) REFERENCES Otdel(nomer))
ENGINE=InnoDB;

Оператор позволяет создать таблицу СОТРУДНИКИ с полями Таб_Номер, ФИО, Номер_Отдела.

Где Таб_номер – это первичный ключ, Номер_Отдела - внешний ключ, ссылающийся на таблицу ОТДЕЛЫ.

5. ALTER TABLE Otdel

ADD FOREIGN KEY (Nom_nach) REFERENCES Sotr(Tab_Nom); - оператор позволяет добавить к таблице ОТДЕЛЫ определение внешнего ключа для атрибута Номер_Начальника, который ссылается на первичный ключ таблицы СОТРУДНИКИ.

Команда DROP TABLE

Команда DROP TABLE позволяет удалять одну или несколько таблиц. Синтаксис команды приведен ниже:

DROP TABLE *ИмяТаблицы1* [, *ИмяТаблицы2*] ...

Для удаления таблицы необходимо иметь привилегию DROP на все удаляемые таблицы. При этом удаляется определение таблицы и все данные.

Просмотр структуры таблиц

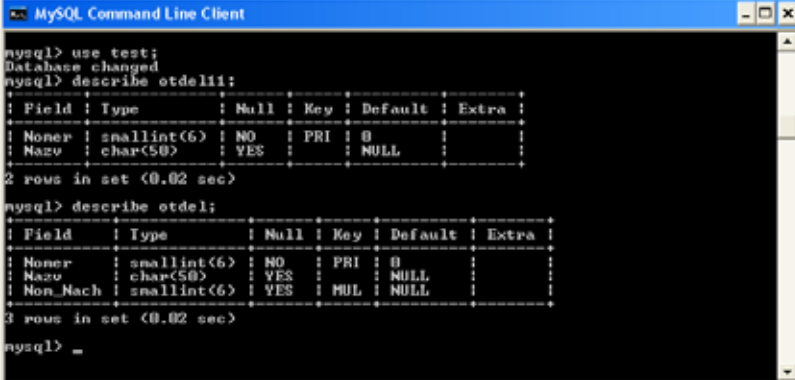
Команда *DESCRIBE*

Для просмотра структуры таблиц можно использовать команду DESCRIBE.

Синтаксис:

{DESCRIBE | DESC} [ИмяТаблицы|ИмяСтолбца]

Команда позволяет просмотреть структуру таблицы или отдельного столбца. Эта команда подобна команде SHOW. В качестве параметров команде передается имя таблицы или отдельного столбца. Пример использования команды приведен на рис 3.3.



```
mysql> use test;
Database changed
mysql> describe otdel11;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Noner | smallint(6)   | NO   | PRI | 0        |       |
| Nazv  | char(50)      | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.02 sec)

mysql> describe otdel1;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Noner | smallint(6)   | NO   | PRI | 0        |       |
| Nazv  | char(50)      | YES  |     | NULL    |       |
| Non_Nach | smallint(6) | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.02 sec)

mysql> _
```

Рис. 3.3.

Команда *SHOW*

Команда SHOW имеет многоцелевое назначение и также позволяет просматривать структуру таблиц, отдельных столбцов, индексы, созданные для отдельных столбцов таблицы. Примеры использования команды приведены в таблице 1.

Таблица 1

Команда	Описание
Show tables;	Выводит список всех таблиц текущей базы данных.
Show tables from <Имя БД>;	Выводит список всех таблиц выбранной базы данных.
Show create table <имя таблицы>\G;	Выводит структуру таблицы в столбец с указанием ключей.
Show columns from <Имя таблицы>;	Выводит описание всех столбцов заданной таблицы.
Show index from < Имя таблицы>;	Выводит описание всех индексов заданной таблицы.
Show table status;	Выводит отображение описательной информации о таблицах данной базы данных.

Контрольные вопросы

1. Перечислить основные операторы языка определения данных, которые позволяют создавать таблицы, изменять их структуру и удалять таблицы.
2. Что можно задать на уровне столбца при создании таблицы оператором CREATE TABLE? Можно ли на уровне столбца описать составной первичный ключ? Можно ли на уровне столбца описать внешний ключ? Можно ли на уровне столбца описать составной внешний ключ?
3. Какие операции изменения структуры таблицы позволяет

делать оператор ALTER TABLE? Как удалить определение первичного ключа? Как удалить определение внешнего ключа?

4. Можно ли добавить ограничение NOT NULL для столбца? Как это сделать?
5. Можно ли добавить значение по умолчанию для столбца? Как это сделать?
6. В каких случаях оператор удаления таблицы DROP TABLE не работает и таблицу не удается удалить?
7. С помощью каких команд можно посмотреть структуру таблицы, конкретного столбца, всех индексов для заданной таблицы?

Глава 4

Работа с данными

Команда INSERT

Команда INSERT относится к командам манипулирования данными и позволяет добавлять одну или несколько строк к таблице. Команда имеет два варианта синтаксиса. Оба варианта приведены ниже:

```
INSERT INTO ИмяТаблицы [(ИмяАтрибута, ...)] VALUES  
(Выражение,...)
```

или

```
INSERT INTO ИмяТаблицы [(ИмяАтрибута, ...)] SELECT ...
```

Описание

В случае 1-го варианта синтаксиса команда добавляет одну строку к таблице и заполняет ее значениями, указанными в

списке VALUES. При этом список столбцов [(ИмяАтрибута, ...)] должен точно соответствовать списку значений, указанному в круглых скобках после VALUES. В выражении, следующем после VALUES можно использовать любое поле из списка атрибутов (или таблицу, если список имен столбцов не задан).

В случае 2-го варианта синтаксиса команда добавляет к таблице подмножество строк из другой таблицы, определяемое оператором SELECT. Оператор SELECT должен использоваться без ORDER BY.

Замечания

- Список столбцов [(ИмяАтрибута, ...)] должен точно соответствовать списку значений, указанному в VALUES.
- В *Выражении* можно использовать любое поле из списка имен атрибутов (или таблицу, если список имен столбцов не задан).
- В случае второго варианта синтаксиса SELECT должен использоваться без ORDER BY.

Примеры

1. INSERT INTO Sotr (FIO, N_otd, zarpl) VALUES (“Иванов И И”, 1, 15000) – при выполнении оператора создается новая запись в таблице Sotr и поля FIO, N_otd, zarpl заполняются значениями, указанными в списке VALUES. При этом поле AUTO_INCREMENT, если оно присутствует, будет автоматически заполняться нужными значениями. Все поля, которым не назначены значения будут заполнены значением NULL(пусто).

2. INSERT INTO Otdel1 SELECT * FROM Otdel2; - при выполнении этот оператор копирует все данные из таблицы Otdel2 в таблицу Otdel1.

3. INSERT INTO Otdel VALUES (1, “Отдел кадров”), (2, “Плановый отдел”), (3, “Бухгалтерия”); - Допускается еще и

такая форма оператора, позволяющая с помощью одной команды INSERT добавить в таблицу несколько строк.

Для выполнения операции INSERT нужно иметь права доступа *insert* для таблицы *Sotr*.

Команда UPDATE

Команда UPDATE относится к командам манипулирования данными и позволяет обновлять одну или несколько записей в таблице. Команда также имеет многотабличный синтаксис, позволяющий обновлять строки нескольких таблиц. Синтаксис команды приведен ниже:

```
UPDATE ИмяТаблицы1 [ ,ИмяТаблицы2 ]  
  SET ИмяСтолбца1 = { Выражение1|DEFAULT }  
  [ ,ИмяСтолбца2= { Выражение1|DEFAULT } ] ...  
  [WHERE Условие]  
  [ORDER BY ...]  
  [LIMIT КолвоСтрок]
```

Описание

После оператора UPDATE указывается список таблиц, столбцы которых будут обновлены. Ключевое слово SET задает список столбцов с указанием значений, которые должны быть им назначены. Одной командой можно обновить несколько столбцов. Условие WHERE задает подмножество строк, которое будет обновлено.

В случае многотабличного синтаксиса, необходимо после оператора UPDATE указать список таблиц, столбцы которых будут обновлены, а в опции WHERE указать условие, которое задает подмножество строк, которые будут обновлены. Все обновления выполняются слева направо.

Примеры

1. Следующий оператор демонстрирует пример обновления строк одной таблицы:

```
UPDATE Sotr SET Zarpl=Zarpl*1,2 WHERE N_OTD=3;
```

С помощью этого оператора поле Zarpl будет увеличено в 1,2 раза для тех записей, у которых N_OTD=3.

2. Приведем пример обновления строк нескольких таблиц. Например, отдел с номером 3 переименован в отдел с номером 33. Для этого требуется изменить значение поля «номер отдела» в таблице «Отделы» и значение поля «номер отдела» в таблице «Сотрудники» таким образом: все сотрудники, которые ранее работали в 3 отделе, теперь будут работать в 33 отделе. Таким образом, необходимо обновить строки двух таблиц Otdel и Sotr одновременно. Оператор, реализующий эту операцию, приведен ниже:

```
UPDATE Otdel, Sotr SET Otdel.N_otd=33, Sotr.N_Otd=33  
WHERE Otdel. N_otd=3 and Sotr.N_Otd=3;
```

Команда DELETE

Команда DELETE относится к командам манипулирования данными и позволяет удалять одну или несколько записей как в одной таблице, так и в нескольких. Команда также имеет многотабличный синтаксис.

Синтаксис удаления записей в одной таблице:

```
DELETE FROM ИмяТаблицы  
[WHERE условие]  
[ORDER BY ...]  
[LIMIT КолСтрок]
```

Синтаксис удаления записей в нескольких таблицах:

```
DELETE ИмяТаблицы1 [. *] [, ИмяТаблицы2 [. *]] ...  
FROM СписокТаблиц  
[WHERE условие]
```

Описание

Команда удаляет записи, удовлетворяющие условию, заложенному в WHERE. Команда возвращает количество обработанных записей. Если команда DELETE используется без WHERE, то все записи в таблице будут удалены. В этом случае DELETE вернет 0 для числа обработанных записей.

В случае многотабличного синтаксиса после DELETE указывается список таблиц, данные которых должны быть удалены. После FROM указывается список таблиц, которые используются для удаления. В конструкции WHERE необходимо указывать условие соединения таблиц и условие отбора тех записей, которые подлежат удалению.

Для выполнения этой операции необходимо иметь права доступа *delete*.

Примеры

1. DELETE FROM Otdel; - в результате работы оператора будут удалены все строки таблицы Otdel.

2. DELETE otdel, sotr FROM otdel1, sotr1

WHERE otdel.nomer =sotr.nom AND sotr.nom =55; - многотабличный оператор, в результате работы будет удалена одна строка в таблице Otdel и все строки из таблицы Sotr, которые на нее ссылаются. Фактически этот оператор реализуется каскадное удаление.

Команда SELECT

Команда SELECT является ключевой командой языка SQL. Собственно с нее и начинался язык SQL. Команда позволяет осуществлять выборки данных из одной или нескольких соединенных по условию таблиц. Команда имеет довольно сложный синтаксис.

SELECT
 [ALL | DISTINCT | DISTINCTROW]
 [FROM *СсылкиНаТаблицы*
 [WHERE *Условие*]
 [GROUP BY {*ИмяАтрибута* | *Выражение* | *Позиция*}]
 [HAVING *Условие*]
 [ORDER BY { *ИмяАтрибута* | *Выражение* | *Позиция* }

Или:

SELECT ...
 UNION [ALL | DISTINCT] SELECT ...
 [UNION [ALL | DISTINCT] SELECT ...]

СсылкиНаТаблицы:

СсылкиНаТаблицы [,*СсылкиНаТаблицы*] ...

СсылкиНаТаблицы:

ОписаниеТаблицы

| *СоединениеТаблиц*

ОписаниеТаблицы:

ИмяТаблицы [[AS] *Алиас*]

| (*СсылкиНаТаблицы*)

| { OJ *СсылкиНаТаблицы* LEFT OUTER JOIN

СсылкиНаТаблицы

ON *УсловиеСоединения* }

СоединениеТаблиц:

СсылкиНаТаблицы [INNER] JOIN *ОписаниеТаблицы*
 [*УсловиеСоединения*]
 | *СсылкиНаТаблицы* {LEFT|RIGHT} [OUTER] JOIN
СсылкиНаТаблицы *УсловиеСоединения*

Описание

Разберем подробнее основные ключевые слова.

ALL – означает, что результирующий набор данных будет содержать все записи, в том числе и одинаковые.

DISTINCT или DISTINCTROW – это синонимы, которые означают, что результирующая выборка будет содержать только разные записи.

FROM – задает список таблиц, из которых выбираются данные.

WHERE – задает условие выборки данных из таблиц, а также условие соединения таблиц.

GROUP BY – задает поле(поля), по которому данные объединяются в группы для подведения итогов.

HAVING – задает условие, накладываемое на группы.

ORDER BY – задает поле(поля), которым выполняется сортировка данных в результирующей выборке.

UNION – используется для выполнения операции объединения таблиц.

INNER JOIN – задает внутреннее соединение таблиц по условию.

{LEFT|RIGHT} OUTER JOIN – задает внешнее(левое или правое) соединение таблиц по условию.

AS – при описании таблиц позволяет задать второе имя для таблицы(алиас). Алиасы удобно использовать при описании списка столбцов в выборке, помечая каждый столбец алиасом той таблицы, из которой он выбирается.

Для использования команды SELECT необходимо иметь права SELECT.

Рассмотрим использование оператора SELECT на примере выборок из следующих таблиц:

Таблица «Темы» содержит информацию о темах, к которым

относится та или иная книга. Имеет следующую структуру:

Tem(*N_t(номер темы), Nazv_t(название темы))

Таблица «Книги» содержит информацию о книгах, имеющихся в библиотеке. Имеет следующую структуру:

Books(*N_b(номер книги), Nazv(название), Avtor(автор), Cena(цена), God_izd(год издания), N_tem(номер темы))

Таблица «Читатели» содержит информацию о читателях библиотеки. Имеет следующую структуру:

Readers(*N_r(номер читателя), Fio(ФИО), Adres(Адрес), Tel(телефон), Pol(пол))

Таблица «Регистрация выдачи и возврата книг» содержит информацию о том кому какие книги выдавались. Имеет следующую структуру:

Registr(*N_b(номер книги), *N_r(номер читателя), *Data_v(Дата выдачи), Priznak(признак возврата)).

Первичные ключи в таблицах отмечены «звездочками»(*).

Простая выборка. Если не задается ключевое слово WHERE, то выбираются все строки из таблицы. Если вместо имен столбцов задается *(звездочка), то выбираются все столбцы таблицы.

Пример 1. Выбрать всю информацию из таблицы «Читатели».

Решение 1. SELECT * FROM Readers;

Пример 2. Выбрать ФИО всех читателей библиотеки:

Решение 2. SELECT Fio FROM Readers;

Выборка с исключением дубликатов. Для того, чтобы в выборке не содержались одинаковые записи, используется ключевое слово DISTINCT.

Пример 3. Выдать список фамилий авторов всех книг библиотеки.

Решение 3. SELECT DISTINCT Avtor FROM Books;

Использование вычисляемых полей. Вычисляемое поле, это поле, которого нет в исходной таблице, которое создается на основе существующих полей таблицы, путем использования арифметических операций.

Пример 4. Выбрать дату возврата каждой книги, если учесть, что книги выдается ровно на 15 дней.

Решение 4. SELECT Data_v+15 as “Возврат” FROM Registr;

Ограниченная выборка позволяет задать условия, накладываемые на строки таблицы. Условия задаются после ключевого слова WHERE. При задании условий можно использовать:

- Логические связки AND и OR.
- Оператор отрицания NOT.
- Операторы BETWEEN, LIKE, IN.

Использование BETWEEN, LIKE, IN.

- a. Оператор BETWEEN ... AND ... позволяет ограничить диапазон изменения числового поля в результирующей выборке. Иными словами в выборке будут содержаться только те значения, которые попадают в диапазон, заданный в BETWEEN.
- b. Оператор LIKE позволяет задать шаблон, которому должны соответствовать значения поля. Шаблон это символьная строка, в которой символ «_»(подчерк) означает одиночный символ, а символ «%» означает любое количество символов.
- c. Оператор IN(содержится в списке) позволяет задать список значений через запятую, которому должны удовлетворять значения поля.

Пример 5. Выбрать все книги, цена которых располагается в интервале от 100 до 500 рублей.

Решение 5. SELECT * FROM Books WHERE Cena BETWEEN 100 AND 500.

Тот же результат даст следующий оператор:

SELECT * FROM Books WHERE Cena >= 100 AND Cena <= 500;

Пример 6. Выбрать всех читателей, фамилии которых начинаются на букву «К».

Решение 6. SELECT * FROM Readers WHERE Fio LIKE "К*";

Пример 7. Выбрать все книги Толстого Л.Н., Чехова А.П., Достоевского Ф.М.

Решение 7. SELECT * FROM Books WHERE Avtor IN ("Толстой Л.Н.,"Чехов А.П.,"Достоевский Ф.М.");

Выборка с упорядочением. Если в результирующей выборке необходимо упорядочить значения по одному или нескольким полям, используется ключевое слово ORDER BY, после которого указывается список полей через запятую, по которым необходимо отсортировать результирующий набор данных.

Пример 8. Выбрать все книги Дюма А. и отсортировать по названиям.

Решение 8. SELECT * FROM Books WHERE Avtor="Дюма А" ORDER BY Nazv;

Пример 9. Выбрать все книги, изданные за период с 2000 по 2005 годы и отсортировать по авторам и названиям.

Решение 9. SELECT * FROM Books WHERE God_izd BETWEEN 2000 AND 2005 ORDER BY Avtor, Nazv;

Запросы к нескольким таблицам. С помощью оператора SELECT можно выполнять операцию соединения (JOIN) нескольких таблиц по условию и операцию объединения таблиц (UNION).

Соединения бывают внутренние (INNER) и внешние (OUTER). Внутренние соединения содержат только те строки исходных таблиц, которые удовлетворяют условию соединения. Внешние соединения могут содержать строки, не удовлетворяющие условию соединения.

Внутренние соединения (INNER JOIN) можно выполнять двумя способами:

1. С помощью ключевого слова JOIN, при этом после ключевого слова ON указывается условие соединения таблиц. Например, выбрать названия всех детективов, которые имеются в фонде библиотеки. Для решения данной задачи необходимо выполнить соединение двух таблиц: «Темы» и «Книги» по полю «Номер темы» и ограничить выборку только детективами:

```
SELECT * FROM Books B INNER JOIN Tem T
      ON B.N_tem=T.N_t
      WHERE Nazv_t="Детектив";
```

2. Без использования JOIN, но с помощью ключевого слова WHERE, которое задает условие соединения таблиц и условие выборки.

Например, решим задачу предыдущего пункта:

```
SELECT * FROM Books B, Tem T
      WHERE B.N_tem=T.N_t AND Nazv_t="Детектив";
```

Внешние соединения таблиц. Внешние соединения (OUTER JOIN) бывают двух видов: левые (LEFT) и правые (RIGHT). Левое внешнее соединение содержит все строки левой таблицы, а из правой выбираются только те строки, которые удовлетворяют условию соединения. Те строки левого отношения, для которых не нашлось соответствующих условию строк правого отношения, будут соединены с пустой (NULL) строкой, соответствующей схеме правого отношения. Правое внешнее соединение, наоборот,

будет содержать все строки правого отношения, а из левого выбираются только те строки, которые удовлетворяют условию соединения.

Например, имеем таблицы следующей структуры, в которых содержится информация о парке машин организации.

Таблица «Парк» содержит описание всех машин организации:

Park(*N_m, FIO, Type_m, Dvig, Probeg), где

N_m – номер машины, первичный ключ;

FIO – ФИО водителя;

Type_m – тип машины (марка);

Dvig – двигатель;

Probeg – пробег.

Таблица «Рейсы» содержит описание всех рейсов, которые выполнялись машинами данной организации:

Reis(*N_m, *Data, Vrema, Desc), где

N_m – номер машины,

Data – дата поездки,

Vrema – время в пути,

Desc – описание поездки.

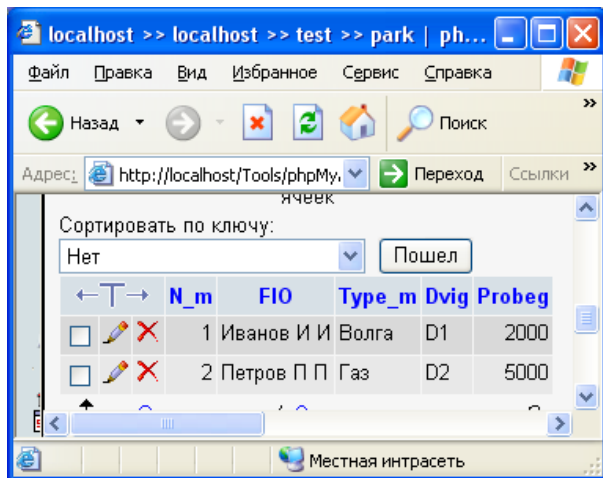
Таблица имеет составной первичный ключ N_m и Data.

Требуется выдать отчет за определенный период, скажем за период с dat1 по dat2, в котором должна присутствовать информация обо всех машинах и выполняющихся ими рейсах, даже если машина ни разу не покидала гараж за этот период. Это можно реализовать с помощью левого внешнего соединения. В качестве левой таблицы используется таблица «Парк», а в качестве правой таблицы «Рейсы». Эта выборка даст точную информацию о занятости каждой машины в течение указанного периода:

```
SELECT * FROM Park P LEFT OUTER JOIN Reis R
```

ON P.N_m=R.N_m
 WHERE Data >=dat1 AND Data <= dat2;

Например, содержимое таблицы «Park» на рис 4.1.:



Содержимое таблицы «Рейсы»:

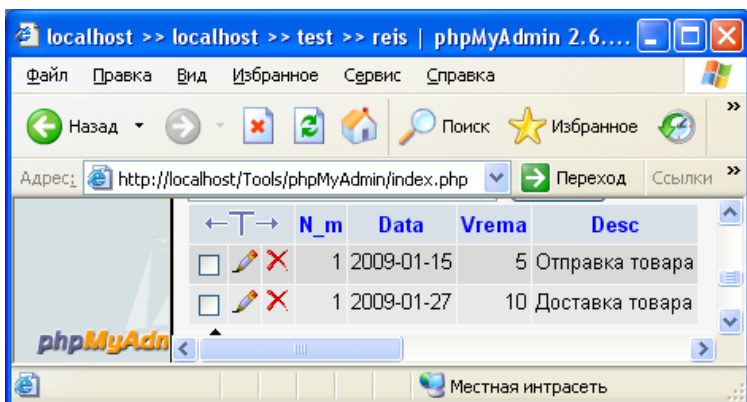
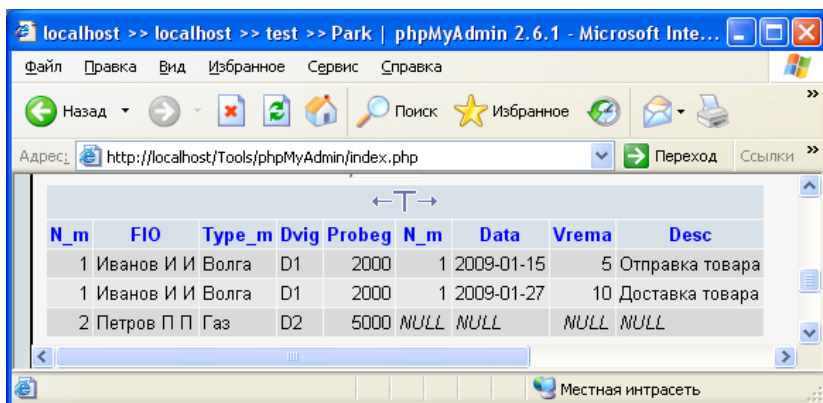


Рис. 4.1.

Результат выполнения запроса на рис 4.2.:



N_m	FIO	Type_m	Dvig	Probeg	N_m	Data	Vrema	Desc
1	Иванов И И	Волга	D1	2000	1	2009-01-15	5	Отправка товара
1	Иванов И И	Волга	D1	2000	1	2009-01-27	10	Доставка товара
2	Петров П П	Газ	D2	5000	NULL	NULL	NULL	NULL

Рис.4.2.

Объединение таблиц(UNION). Для выполнения этой операции необходимо, чтобы таблицы имели идентичные схемы, т.е. имели одинаковое количество столбцов, и соответствующие столбцы должны быть определены на одних и тех же типах данных. Операция объединения таблиц позволяет исключить строки дубликаты.

Например, даны две таблицы:

Student1(*Nom1, FIO1) и
Student2(*Nom2, FIO2).

Атрибуты Nom1 и Nom2 должны иметь одинаковые типы данных, FIO1 и FIO2 также должны иметь одинаковые типы данных. Тогда результат объединения отношений будет содержать строки, которые входят хотя бы в одно из отношений операндов. Следующий оператор реализует эту операцию:

```
SELECT * FROM Student1 UNION  
SELECT * FROM Student2;
```

Пример применения этой операции к таблицам Student1 и Student2 приведен на рис. 4.3.

```

MySQL Command Line Client
mysql> select * from Student1;
+----+-----+
| nom1 | FI01 |
+----+-----+
| 101 | Dora |
| 102 | Catarine |
| 103 | Joker |
+----+-----+
3 rows in set (0.00 sec)

mysql> select * from Student2;
+----+-----+
| nom2 | FI02 |
+----+-----+
| 103 | Joker |
| 201 | Stive |
| 202 | Mike |
+----+-----+
3 rows in set (0.00 sec)

mysql> select * from Student1 UNION select * from student2;
+----+-----+
| nom1 | FI01 |
+----+-----+
| 101 | Dora |
| 102 | Catarine |
| 103 | Joker |
| 201 | Stive |
| 202 | Mike |
+----+-----+
5 rows in set (0.00 sec)

mysql>

```

Рис. 4.3.

Использование функций в запросах

СУБД MySQL предоставлять широкий набор встроенных функций, который можно использовать для преобразования данных. Описание арифметических операций, операторов и некоторых встроенных функций приведено в таблице 2:

Таблица 2

Операторы и функции	Описание
+ - * /	Арифметические операции.
%	Остаток от деления (как в C)
, &	Битовые функции (используется 48 бит).

Продолжение таблицы 2

- <число>	Замена знака числа.
()	Скобки.
BETWEEN(A,B,C)	Аналогично (A >= B) AND (A <= C).
BIT_COUNT()	Возвращает количество бит.
ELT(N,a,b,c,d)	Возвращает a, если N = 1, b, если N = 2 и т. д. Например: ELT(3,»First»,»Second»,»Third»,»Fourth») вернет «Third».
FIELD(Z,a,b,c)	Возвращает a, если Z = a; b, если Z = b и т. д. Например: Z=»Second» FIELD(Z, «First», «Second», «Third», «Fourth») вернет «Second».
IF(A,B,C)	Если A = истина, то вернет B, иначе вернет C.
IFNULL(A,B)	Если A не NULL, вернет A, иначе вернет B.
ISNULL(A)	Вернет истину(1), если A = NULL, иначе вернет ложь(0). Эквивалент ('A == NULL').
NOT	Операция отрицания
OR, AND	Операции «ИЛИ» и «И»
SIGN()	Возвращает знак аргумента
SUM()	Сумма столбца.
=, <, <=, >, >=	Операции сравнения
<строка> [NOT] REGEXP <рег выраж>	Проверяет строку на соответствие или не соответствие регулярному выражению <рег выраж>.

Математические функции приведены в таблице 3:

Таблица 3

Функции	Описание
ABS(X)	Возвращает абсолютное значение (модуль) числа X.
CEILING(X)	Возвращает наименьшее целое, не меньшее, чем X.
EXP(X)	Экспонента от X.
FORMAT(X,NUM)	Преобразует число X в формат '#,###,###.##' с NUM десятичных цифр. При этом точка это разделитель целой и дробной части, запятая это разделитель разрядов.
LOG(X)	Логарифм от X.
LOG10(X)	Логарифм X по основанию 10.
MIN(X),MAX(X)	Минимум или максимум соответственно. Должна иметь при вызове два или более аргументов, иначе рассматривается как групповая функция.
MOD(X, Y)	Остаток от деления X на Y (аналог %).
POW(X, Y)	Возвращает X в степени Y.
ROUND(X)	Округление X до ближайшего целого числа.
RAND([X])	Возвращает случайное число типа float, в интервале от 0 до 1.0, X используется как начальное значение генератора.
SQRT(X)	Квадратный корень от X.

Функции для работы со строками приведены в таблице 4:

Таблица 4

Функции	Описание
CONCAT(стр1, стр2, ...)	Объединение строк стр1, стр2, ...
INTERVAL(A,a,b,c,d)	Возвращает 1, если $A == a$, 2, если $A == b$, и т.д. Если совпадений нет, вернет 0. A,a,b,c,d,... строки.
INSERT(str, pos, len, new_str)	Возвращает строку str, в которой подстрока, начинающаяся с позиции pos и имеющая длину len символов, заменяется подстрокой new. Первая позиция строки=1.
LCASE(str)	Приводит строку str к нижнему регистру.
LEFT(str, len)	Возвращает len крайних левых символов строки str.
LENGTH(str)	Возвращает длину строки str.
LOCATE(substr, str)	Возвращает позицию первого вхождения подстроки substr в строку str.
LOCATE(substr, str, pos)	Возвращает позицию первого вхождения подстроки substr в строку str. Поиск начинается с позиции pos.
LTRIM(str)	Возвращает строку str без левых пробелов.
REPLACE(str, from_str, to_str)	Возвращает строку str, в которой все подстроки from_str заменены на to_str.
RIGHT(str)	Возвращает строку str без правых пробелов.
RTRIM(str)	Удаляет хвостовые пробелы из строки str.
STRCMP(str1, str2)	Возвращает 0, если строки одинаковые; -1, если $str1 < str2$; 1, если $str1 > str2$.
SUBSTRING(str, pos1, pos2)	Возвращает подстроку из str, с позиции , pos1 до позиции pos2.
UCASE(str)	Переводит строку str в верхний регистр.

И, наконец, несколько просто полезных функций в таблице 5:

Таблица 5

Функции	Описание
CURDATE()	Возвращает текущую дату.
DATABASE()	Возвращает имя текущей базы данных, из которой выполняется выбор.
ROW_COUNT()	Возвращает число записей, которые подверглись изменению в последнем SQL-запросе. Под изменением здесь подразумевается вставка, удаление, изменение.
LAST_INSERT_ID(), LAST_INSERT_ID(par)	Возвращает последнее автоматически сгенерированное значение для столбца с атрибутом AUTO_INCREMENT. Функция может принимать параметр par, который возвращается функцией и запоминается как следующее значение, которое функция LAST_INSERT_ID() вернет при повторном вызове.
NOW()	Возвращает текущее время в форматах YYYYMMDDHHMMSS или «YYYY-MM-DD HH:MM:SS». Формат зависит от того в каком контексте используется NOW(): числовом или строковом.
PASSWORD(str)	Шифрует строку str.
PERIOD_ADD(P, N)	Добавить N месяцев к периоду P (в формате YYMM).
PERIOD_DIFF(P1, P2)	Возвращает разницу в месяцах между двумя датами P1 и P2. Обратите внимание, что PERIOD_DIFF работает только с датами в форме YYMM или YYYYMM.
TO_DAYS(date)	Меняет дату date (YYMMDD) на номер дня.
USER()	Возвращает логин текущего пользователя.
WEEKDAY()	Возвращает день недели (0 = понедельник, 1 = вторник, ...).

Контрольные вопросы и упражнения

1. Можно ли одним оператором INSERT к одной таблице добавить все строки из другой таблицы? Существуют ли при этом ограничения на структуру таблиц, т.е. таблицы могут иметь произвольную структуру или обязаны иметь идентичную структуру?
2. Можно ли при добавлении новой записи в таблицу с помощью оператора INSERT не заполнять новым значением поле AUTO_INCREMENT. Если да, то какое значение получит это поле?
3. Можно ли при добавлении новой записи в таблицу с помощью оператора INSERT не указывать список имен полей, в которые заносятся значения перед VALUES?
4. В каких случаях требуется явное указание списка полей перед VALUES при добавлении новой записи в таблицу с помощью оператора INSERT?
5. Можно при обновлении записей командой UPDATE обновить сразу несколько полей таблицы?
6. Можно ли с помощью оператора UPDATE обновить записи нескольких таблиц? Приведите пример.
7. Можно ли при удалении записей оператором DELETE удалить записи нескольких таблиц? Приведите пример.
8. Используя таблицы, введенные в разделе «Команда SELECT» выбрать фамилии и телефоны должников библиотеки. Должником считается читатель, который не сдал книгу во время, при этом книги выдаются на 15 дней.
9. Используя таблицы, введенные в разделе «Команда SELECT» выбрать ФИО и телефоны читателей, у которых в данный момент находится книга Дюма А. «Граф Монте-Кристо».
10. Используя таблицы, введенные в разделе «Команда SELECT» выбрать информацию обо всех книгах, которые когда-либо брал читатель с номером билета 101. То есть необходимо получить читательскую карточку этого читателя. Результат должен содержать следующие поля: Номер читателя, ФИО читателя, Телефон читателя, Название книги, Автор, Дата выдачи, Признак возврата.

Глава 5

Индексы и представления

Индексы(Ключи)

Индексы это объекты базы данных, которые используются для быстрого доступа к нужным данным. Индексы играют очень важную роль в реляционных базах данных. Идея индекса заключается в том, чтобы создать для столбца копию, которая будет постоянно поддерживаться в отсортированном состоянии. Это позволяет очень быстро осуществлять поиск по такому столбцу, т.к. значения в индексе отсортированы.

Обратной стороной медали является то, что добавление или удаление записи требует дополнительного времени на сортировку столбца, кроме того, создание копии увеличивает объем памяти, необходимый для размещения таблицы на жестком диске.

MySQL таблица может иметь до 16 ключей, каждый из которых может иметь до 15 полей. Следует иметь в виду, что длинные ключи могут привести к низкой эффективности.

Существует несколько видов индексов:

- Первичный ключ;
- Внешний ключ;
- Обычный индекс;
- Полнотекстовый индекс.

Полнотекстовый индекс – специальный вид индекса для полей типа TEXT, позволяющий производить полнотекстовый поиск.

Создание индекса

Для полей первичного и внешнего ключей индексы создаются автоматически при объявлении этих полей. Таблица может иметь только один первичный ключ. Первичный и внешний индексы могут быть заданы либо при создании таблицы командой CREATE TABLE, либо добавлены после создания таблицы с помощью команды ALTER TABLE. Таблица может содержать несколько обычных и уникальных индексов, в отличие от первичного ключа. Обычные индексы могут быть созданы следующими способами:

1. При создании таблицы с помощью команды CREATE TABLE.
2. Добавлены с помощью команды ALTER TABLE.
3. С помощью команды CREATE INDEX.

Рассмотрим создание индекса с помощью команды CREATE TABLE, например:

```
CREATE TABLE tb1 (  
  Id INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,  
  Id_tb1 INT(11) NOT NULL DEFAULT '0',  
  Id_cat INT(11) NOT NULL DEFAULT '0',  
  KEY id_tb1 (id_tb1),  
  UNIQUE KEY id_cat (id_cat);
```

После ключевого слова KEY следует имя индекса, а в круглых скобках указывается имя столбца, для которого создается индекс.

В таблице tb1 наряду с индексом для первичного ключа будут созданы два индекса id_tb1 и id_cat. Индекс id_tb1 создается по полю id_tb1. Индекс id_cat создается по полю id_cat и является уникальным, т.е. запрещает ввод повторяющихся значений в поле id_cat. Обратите внимание, что имена индексов могут совпадать с именами полей.

Рассмотрим создание индекса с помощью команды ALTER TABLE, например:

```
ALTER TABLE Sotr ADD INDEX Idx_Fam( Fio(10));
```

К таблице Sotr добавляется индекс с именем Idx_Fam по полю Fio по первым 10 символам.

Рассмотрим создание индекса с помощью команды CREATE INDEX. Синтаксис команды CREATE INDEX:

```
CREATE [UNIQUE|FULLTEXT]
INDEX <ИмяИндекса>
ON <ИмяТаблицы> (<СтолбецВИндексе>,...)
СтолбецВИндексе:
ИмяСтолбца [(длина)] [ASC | DESC]
```

Описание

Ключевое слово UNIQUE – задает уникальный индекс, это означает, что не допускаются повторяющиеся значения в столбцах, на основе которых создан индекс.

Ключевое слово FULLTEXT – задает полнотекстовый индекс.

Можно задать индекс не по всей длине поля, а только первым несколькими символам. Длина ключа указывается в скобках после имени поля.

Ключевое слово ASC означает, что значения ключевого поля в индексе отсортированы по возрастанию.

Ключевое слово DESC означает, что значения ключевого поля в индексе отсортированы по убыванию.

Например:

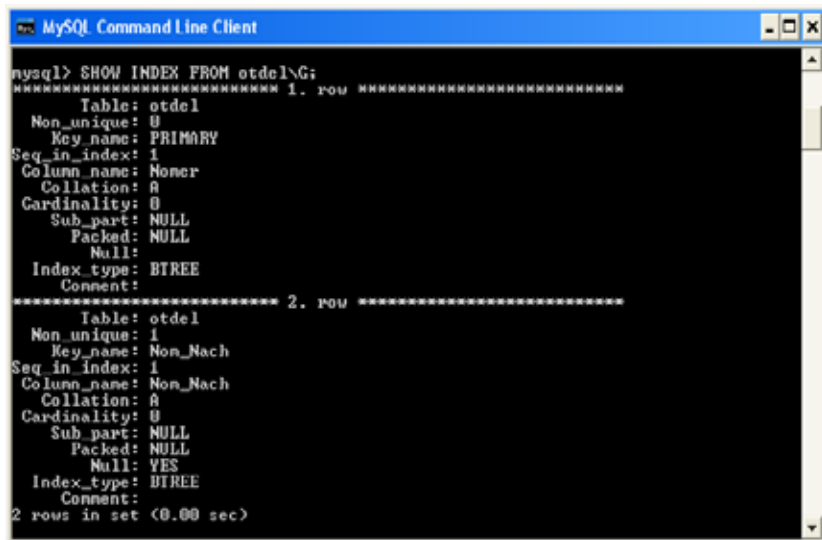
```
CREATE INDEX Idxfio1 ON Sotr (Fio(5) ASC);
```

Создается индекс с именем Idxfio1 для таблицы Sotr для поля Fio по первым его 5-ти символам. Данные в индексе отсортированы по возрастанию, на что указывает ключевое слово ASC.

Просмотр существующих индексов

Просмотр созданных индексов для таблицы можно выполнить с помощью команды SHOW INDEX FROM <ИмяТаблицы>\G;

Пример использования этой команды показан на рис. 5.1. Ключевое слово G указывает, что информация о каждом индексе будет выводиться построчно.



```
mysql> SHOW INDEX FROM otdel\G;
+-----+-----+
Table: otdel
Non_unique: 0
Key_name: PRIMARY
Seq_in_index: 1
Column_name: Noner
Collation: A
Cardinality: 0
Sub_part: NULL
Packed: NULL
Null:
Index_type: BTREE
Comment:
+-----+-----+ 1. row +-----+-----+
Table: otdel
Non_unique: 1
Key_name: Non_Nach
Seq_in_index: 1
Column_name: Non_Nach
Collation: A
Cardinality: 0
Sub_part: NULL
Packed: NULL
Null: YES
Index_type: BTREE
Comment:
+-----+-----+ 2. row +-----+-----+
2 rows in set (0.00 sec)
```

Рис. 5.1.

Удаление индекса

С помощью команды DROP INDEX можно удалить существующий индекс. При этом необходимо указывать таблицу, для которой создан данный индекс. Синтаксис:

DROP INDEX <ИмяИндекса> ON <ИмяТаблицы>;

Восстановление индекса

Файл индексов может быть поврежден, в связи с чем запросы с использованием индексов либо вообще не будут выполняться, либо будут выполняться неправильно. Для восстановления индексов используются команды:

CHECK TABLE <СписокИменТаблиц>;
REPAIR TABLE <СписокИменТаблиц>;

Для каких полей следует создавать индексы

Объявление ненужных ключей (индексов) только займет дополнительное место и замедлит выполнение запроса. Поэтому следует продумать вопрос, а для каких полей имеет смысл создавать индексы.

Следует создавать индексы для полей, по которым часто выполняется поиск (выборки), сортировка, соединение таблиц. Для остальных полей индексы создавать не стоит, это займет дополнительное место и замедлит выполнение запросов, т.к. при добавлении, обновлении, удалении данных индексы так же обновляются.

Следует помнить, что индексация столбца, в том числе с индексом FULLTEXT, требует дополнительного объема памяти для хранения индекса, иногда превышающего объем основных данных в несколько раз, и приводит к замедлению операций вставки и удаления с помощью команд INSERT и DELETE. Полнотекстовый поиск в СУБД MySQL на сегодняшний день поддерживается только для таблиц типа MyISAM и столбцов CHAR, VARCHAR и TEXT.

Представления (просмотры)

Представление (просмотр) это объект базы данных, SELECT-запрос, который сохраняется в базе данных. Представление позволяет ограничить доступ пользователя к подмножеству строк и столбцов одной или нескольких таблиц. С представлениями работают 3 команды SQL:

CREATE VIEW – создать представление.

ALTER VIEW - изменить представление.

DROP VIEW - удалить представление.

Создание представления

Команда CREATE VIEW имеет следующий синтаксис:

```
CREATE [OR REPLACE] VIEW ИмяПредставления  
[(СписокИменСтолбцов)]  
AS select-предложение [WITH CHECK OPTION]
```

Команда CREATE VIEW создает новое представление или замещает существующее, если используется OR REPLACE. Если представление уже существует, то команда CREATE OR REPLACE VIEW это то же самое, что и ALTER VIEW.

СписокИменСтолбцов – это имена столбцов в представлении.

select-предложение – задает оператор SELECT, который определяет тело представления. Выборка может выполняться как из таблиц, так и из других представлений.

Ключевые слова WITH CHECK OPTION не позволяют вставлять или изменять через представление строки базовой таблицы, которые не удовлетворяют уточнителю WHERE в select-предложении.

Для создания представления необходимо иметь привилегию CREATE VIEW.

Например, создадим представление VSotr3 для таблицы Sotr(Tab_nom, Fio, Nom_Otd), которое позволяет видеть только сотрудников 3 отдела:

```
mysql> CREATE VIEW VSotr3 AS SELECT * FROM Sotr  
WHERE Nom_Otd=3;
```

С помощью команды SHOW TABLES; - убедимся с том, что представление создано.

С помощью команды SELECT * FROM VSotr3; убедимся в том, представление позволяет видеть только сотрудников 3 отдела.

Создадим еще одно представление для этой же таблицы, но с опцией WITH CHECK OPTION.

```
mysql> CREATE VIEW VSotr3Ch AS SELECT * FROM Sotr  
WHERE Nom_otd=3 WITH CHECK OPTION;
```

Некоторые представления являются модифицирующими (типа `updatable`), т.е. позволяют с помощью операторов `UPDATE`, `DELETE`, или `INSERT` изменять данные базовой таблицы. Для того, чтобы представление было представлением типа `updatable` необходимо, чтобы существовали связи один-к-одному между строками представления и базовой таблицы. Существуют также и другие ограничения.

Представление является представлением типа `updatable`, т.е. позволяющим изменять строки таблицы, на которой оно основано, если оно удовлетворяет следующим условиям:

- Является подмножеством единственной таблицы или другого представления типа `updatable`
- Оператор `select`, формирующий тело представления не содержит агрегатные функции (`SUM()`, `MIN()`, `MAX()`, `COUNT()` и т.д.), `DISTINCT`, `GROUP BY`, `HAVING`, `UNION` или `UNION ALL`, подзапросы, не `updatable`-представления.
- Столбцы, не входящие с представление должны допускать ввод `NULL` значений.
- В противном случае представление является представлением типа `read only`, т.е. не позволяющим изменять строки базовой таблицы.

Оба представления `VSotr3Ch` и `VSotr3`, созданные нами, являются `updatable`-представлениями. Единственная разница между ними заключается в том, что представление `VSotr3` позволяет вводить строки, не удовлетворяющие опции `WHERE` оператора `SELECT`, а представление `VSotr3Ch` не позволяет этого делать. Проверим это. Исходные данные таблицы `Sotr` приведены в таблице 6:

Таблица 6

T_nom	fio	nom_otd
101	Sid	1
104	Tom	3

С помощью команды INSERT добавим новую строку в таблицу, используя представление VSotr3:

```
mysql> INSERT INTO VSotr3 VALUES (105,"Min",5);
```

С помощью команды SELECT убедимся в том, что строка действительно добавлена в таблицу.

```
mysql> SELECT * FROM Sotr;
```

Результат работы оператора SELECT * FROM Sort приведен в таблице 7:

Таблица 7

T_nom	fio	nom_otd
101	Sid	1
104	Tom	3
105	Mih	5

Как видим, строка в базовую таблицу добавлена через представление VSotr3.

При добавлении строки, не удовлетворяющей опции WHERE через представление VSotr3Ch, будет выдано сообщение об ошибке.

```
mysql> insert into VSotr3Ch values (106, «Dan», 2);
ERROR 1369 (HY000): CHECK OPTION failed 'proba.vsotr3ch'
```

Команда ALTER VIEW синтаксически подобна команде CREATE VIEW и работает также.

Удаление представления

Для удаления представлений используется команда DROP VIEW, в которой задается имя представления. Например:

```
DROP VIEW VSotr3Ch;
```

Просмотр структуры представления

Структуру конкретного представления можно просмотреть с помощью команды:

```
SHOW CREATE VIEW <ИмяПредставления>\G;
```

Результат работы команды приведен на рис 5.2.

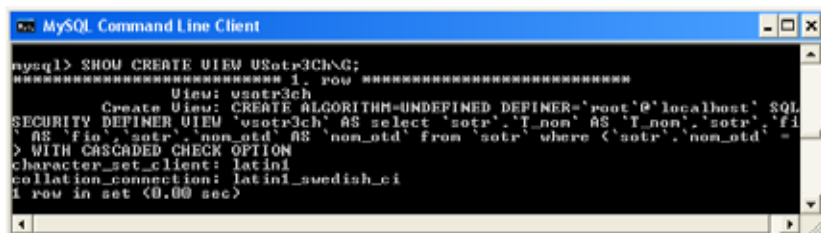


Рис. 5.2.

Для просмотра всех созданных в базе данных представлений необходимо просмотреть содержимое системной таблицы VIEWS из базы данных Information_schema.

Контрольные вопросы и упражнения

1. Что такое индекс? Для чего используются индексы?
2. Перечислить и кратко охарактеризовать типы индексов в MySQL.
3. Каким образом можно создать индекс? Для каких полей следует создавать индексы?
4. Какие команды языка SQL работают с индексами?
5. С помощью какой команды можно просмотреть созданные для таблицы индексы?
6. Что такое представление(просмотр)? Для чего используются представления?
7. Какие команды языка SQL работают с представлениями?
8. Можно ли используя представление изменять строки базовой таблицы, т.е. таблицы, на основе которой создано представление?
9. Какие команды позволяют просмотреть созданные в базе данных представления?

Литература

1. М. Кузнецов, И. Симдянов «MySQL на примерах», С-Пб, 2007
2. www.mysql.com – электронная документация
(на английском языке) по языку SQL СУБД MySQL
3. www.mysql.ru – подборка статей по языку SQL СУБД MySQL
4. Дейт К. «Введение в системы баз данных». М: Наука, 1980.
5. Дейт К. «Руководство по реляционной СУБД DB-2». М: Финансы и статистика, 1988.
6. Грабер М. «Введение в SQL», 2000.
7. Дюбуа П. «MySQL», М, СПб, Киев, 2007

Татьяна Александровна Иваньчева

Методическое пособие по языку SQL

(Диалект СУБД MySQL)

Часть 1