Глава 17

Сервер MariaDB

ЦЕЛИ СЕРТИФИКАЦИИ

17.01 Введение в MariaDB
17.02 Управление базой данных
17.03 Простые SQL-запросы
17.04 Безопасный MariaDB
17.05 Резервное копирование и восстановление базы данных
✓ Двухминутная тренировка
Q & A Самопроверка

Реляционные базы данных, обычно называемые системами управления реляционными базами данных (**RDBMS**), предоставляют стандартизированный метод для организации постоянных данных в структурированном виде. Они используют таблицы для хранения данных, правила для обеспечения уникальности и согласованности между таблицами, а также индексы для поддержки быстрого доступа. Кроме того, большинство систем реляционных баз данных поддерживают язык структурированных запросов (**SQL**), стандартный инструмент для извлечения данных и выполнения многих других задач.

MySQL - самая популярная **СУБД** с открытым исходным кодом, и она является ключевой частью стека «**LAMP**» (**Linux, Apache, MySQL и Perl/Python/PHP**), обычно используемого для поддержки веб-приложений. Он также чрезвычайно прост в установке, настройке и использовании.

До **RHEL 7 MySQL** была стандартной **СУБД** в **Red Hat Enterprise Linux**. После того как **MySQL** был приобретен **Oracle**, **Red Hat** переехала в **MariaDB**, разрабатываемую сообществом версию **MySQL**, лицензированную по лицензии **GPL**. **MariaDB** содержит дополнительные функции и оптимизации, разработанные сообществом. Это не единственная база данных, которая поставляется с **RHEL**. Другие (наиболее очевидно включая **PostgreSQL**) также доступны, но не охвачены экзаменом **RHCE**.

ВНУТРИ ЭКЗАМЕНА

Возможность установки и настройки **MariaDB** (и ее аналога **MySQL**) является общим требованием для системных администраторов, хотя она является новой для экзамена **RHCE** в **RHEL7.** В этой главе непосредственно рассматриваются цели экзамена, относящиеся к **MariaDB**:

Установка и настройка **MariaDB**.
Резервное копирование и восстановление базы данных.
Создание простой схемы базы данных.
Выполнение простых запросов **SQL** к базе данных.

Кроме того, эта глава посвящена общему сетевому сервису. требования обсуждаются в **главе 11.**

ЦЕЛЬ СЕРТИФИКАЦИИ 17.01

Введение в MariaDB

MySQL AB, шведская компания, впервые выпустила **MySQL** в 1995 году в качестве бесплатной реализации более ранней базы данных, известной как mSQL. Первые выпуски были основаны на существующем методе индексации **ISAM** от **IBM**, который в итоге превратился в **DB2**. **MySQL** был включен **Red Hat** в свой первый выпуск **RHEL** и быстро приобрел популярность. В **RHEL 6** включена версия **MySQL 5.1.**

В 2008 году **MySQL** была приобретена **Sun Microsystems**, а в **2009** году **Oracle** приобрела **Sun Microsystems**. Поскольку **Oracle** продает альтернативную **СУБД MySQL**, это приобретение вызвало существенную обратную реакцию как со стороны регулирующих органов, так и сообщества открытого исходного кода.

В конце концов, Европейский Союз разрешил **Oracle** приобрести **Sun в 2010 году**. Чтобы удовлетворить правительственные нормативные требования, **Oracle** взяла на себя обязательство продолжать разработку **MySQL** в соответствии с существующей моделью лицензирования «с двумя источниками».

Один из основателей **MySQL**, Майкл «Монти» Видениус, выбрал **форк MySQL** в 2009 году. Он назвал его **MariaDB** в честь своей младшей дочери Марии. Ранее он назвал **MySQL** в честь своей старшей дочери. **MariaDB** получила финансирование, и значительное количество разработчиков начали переводить свою работу с **MySQL** на новый проект **MariaDB**.

MariaDB был первоначально выпущен с теми же номерами версий, что и **MySQL**, чтобы предложить полную совместимость. После выхода **MariaDB 5.5** разработчики изменили номер версии на 10, частично чтобы отойти от полной совместимости с MySQL. Для наших целей **MariaDB 5.5** полностью совместима с **MySQL 5.5.** Другими словами, клиенты и библиотеки, скомпилированные для **MySQL 5.5,** будут работать только на сервере **MariaDB 5.5**.

Установка MariaDB

RPM-пакет mariadb-server устанавливает в качестве зависимостей **mariadb-libs** и **mariadb**. Эти пакеты включают в себя все файлы, необходимые для работающей установки **MariaDB**, такие как сам сервер (**mysqld**), клиент **MariaDB (mysql)** и все библиотеки **Perl**, необходимые для связанных вспомогательных сценариев.

Если вы хотите разрабатывать приложения, использующие **MariaDB**, вам могут потребоваться пакеты **mariadb-devel** и **MySQL-python**. Тем не менее, они выходят за рамки экзамена **RHCE.**

Для целей этой главы установите сервер **MariaDB** с помощью следующей команды:

```
# yum -y install mariadb-server
```

Эта команда устанавливает сервер **MariaDB**, клиент и **более 30 модулей Perl**. На клиентских компьютерах вы можете установить клиент **MariaDB с RPM-пакетом mariadb**.

Конфигурация не требуется для основной операции. Вы можете запустить и убедиться, что служба переживет перезагрузку с помощью следующих команд:

```
# systemctl start mariadb
# systemctl enable mariadb
```

При первом запуске **MariaDB** она записывает некоторые стандартные таблицы во внутреннюю базу данных «**mysql**», вызывая скрипт **mysql_install_db**. Любые проблемы с этим процессом должны появиться в файле **mariadb.log**, расположенном в каталоге **/var/log/mariadb**.

!!!!! On the Job !!!!
Системный модуль MariaDB в /lib/systemd/system/mariadb.service содержит директиву TimeoutSec=300, которая ограничивает время запуска сервера до 300 секунд. Хотя этого небольшого значения достаточно для небольшой тестовой базы данных, TimeoutSec может привести к проблемам для большого реального сервера баз данных. Без достаточного времени восстановление транзакции может привести к бесконечному циклу неудачных запусков. К счастью, это не проблема экзамена.
!!!!!!

Поскольку **MariaDB** является «форком» MySQL, он сохраняет множество имен файлов и команд, связанных с **MySQL**. Например, клиентская команда **MariaDB** - это **mysql**, а демон сервера - **mysqld**. Среди прочего, модуль **Python** - это **MySQLdb**, и он работает как с серверами **MySQL**, так и с серверами **MariaDB**.

Теперь, когда служба работает, убедитесь, что она прослушивает **TCP-порт 3306** по умолчанию с помощью команды **ss**. Результат показан на **рисунке 17-1**. Обратите внимание на вывод команды, что по умолчанию **MariaDB** прослушивает все интерфейсы, доступные на сервере.

Чтобы убедиться, что **MariaDB** работает, подключитесь к клиенту **mysql**. Результат показан на **рисунке 17-2.** Введите **quit** или **exit**, чтобы закрыть сеанс.

Команда **mysql** имеет различные параметры команды, которые будут подробно описаны в следующих разделах. Наиболее распространенные из них описаны в **таблице 17-1**.

РИСУНОК 17-1 MariaDB прослушивает TCP-порт 3306.

```
[root@server1 ~]# ss -tpna | grep 3306
LISTEN     0      50                          *:3306                     *:*
users:(("mysqld",3584,13))
[root@server1 ~]#
```

РИСУНОК 17-2 MySQL клиент

```
[root@server1 ~]# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 10
Server version: 5.5.35-MariaDB MariaDB Server

Copyright (c) 2000, 2013, Oracle, Monty Program Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

ТАБЛИЦА 17-1 Параметры команды mysql

опция команды mysql	Описание	Значение по умолчанию
-h	Имя хоста/полное доменное имя сервера **MariaDB**	**localhost**
-p	Пароль(Password)	Попробуйте аутентификацию без пароля
-P	Пользовательский номер порта TCP (см. упражнение 17-2)	**3306**
-u	имя пользователя **MariaDB**	Текущее имя пользователя **Linux**

РИСУНОК 17-3 Файл конфигурации /etc/my.cnf

```
[root@server1 ~]# cat /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
# Settings user and group are ignored when systemd is used.
# If you need to run mysqld under a different user or group,
# customize your systemd unit file for mariadb according to the
# instructions in http://fedoraproject.org/wiki/Systemd

[mysqld_safe]
log-error=/var/log/mariadb/mariadb.log
pid-file=/var/run/mariadb/mariadb.pid

#
# include all files from the config directory
#
!includedir /etc/my.cnf.d

[root@server1 ~]#
```

Начальная конфигурация

Хотя вы можете сделать больше, **RHEL 7** включает рабочую конфигурацию **MariaDB** «из коробки». В процессе работы дополнительные изменения, которые вы вносите в конфигурацию **MariaDB**, относятся к настройке производительности.

Изучите конфигурационный файл **MariaDB /etc/my.cnf**, показанный на **рисунке 17-3.** По умолчанию он содержит два раздела: [**mysqld**] и [**mysqld_safe**]. Раздел [**mysqld_safe**] определяет

расположение файлов журнала и идентификатора процесса (**PID**) для **mysqld_safe**, сценария-оболочки, который отслеживает состояние процесса **mysqld** и перезапускает его в случае серьезного сбоя.

Раздел [**mysqld**] начинается с директивы **datadir**, которая определяет местоположение данных. Затем директива сокета указывает на местоположение файла сокета. В типичной установке вам не нужно изменять эти настройки. Последний параметр в этом разделе - это директива символических ссылок, которая запрещает **MariaDB** следовать символическим ссылкам по соображениям безопасности.

Обратите внимание на директиву **includeir** в конце файла **my.cnf**. Он загружает содержимое нескольких других файлов конфигурации из каталога **/etc/my.cnf.d.**

!!!! On the Job !!!!
Директива includeir в файле my.cnf по умолчанию содержит содержимое каждого файла в каталоге /etc/my.cnf.d. По умолчанию файлы в этом расположении влияют только на клиентов MariaDB, но стоит убедиться, что ни один другой пакет не поместил сюда файл при устранении неполадок. !!!!!

MariaDB поставляется со скриптом **mysql_secure_installation** для повышения безопасности конфигурации по умолчанию. После первого запуска службы **MariaDB** запустите этот сценарий от имени пользователя **root**. Он задаст ряд вопросов, связанных с безопасностью, в интерактивном режиме.

Упражнение 17-1 проведет вас через установку **MariaDB** и выполнение сценария **mysql_secure_installation**.

УПРАЖНЕНИЕ 17-1
Установить и защитить MariaDB

В этом упражнении вы установите **MariaDB** и запустите сценарий **mysql_secure_installation** для обеспечения безопасности установки. Сценарий предлагает вам ряд интерактивных вопросов, чтобы установить пароль для пользователя **root** (отличный от суперпользователя **root** Linux!), отключит удаленный вход в систему, удалит анонимных пользователей и удалит тестовую базу данных по умолчанию.

1. Установите **MariaDB:**

   # yum -y install mariadb-server

2. Запустите службу и убедитесь, что она включена для следующей загрузки системы:

   # systemctl start mariadb
   # systemctl enable mariadb

3. Запустите скрипт **mysql_secure_installation**. Когда вы увидите следующее приглашение, просто нажмите **Enter**, поскольку нет пароля для пользователя **root MariaDB**:

   # mysql_secure_installation
   [...]
   Enter current password for root (enter for none):
   OK, successfully used password, moving on...

4. Установите новый пароль **root** для **MariaDB**. Как вы можете видеть здесь, мы установили наш «**changeme**», но вы должны выбрать реальный пароль на рабочем сервере:

   Set root password? [Y/n] y
   New password: changeme
   Re-enter new password: changeme
   Password updated successfully!
   Reloading privilege tables..
   ... Success!

5. По умолчанию **MySQL** поддерживает подключения от анонимных пользователей. Это должно быть отключено, как показано здесь:

Remove anonymous users? [Y/n] y
... Success!

6. Чтобы еще больше препятствовать хакерам, вы должны отключить удаленный **root-доступ** к **MariaDB**:

   Disallow root login remotely? [Y/n] y
   ... Success!

7. Установка **MariaDB** включает в себя базу данных по умолчанию с именем **test**. Хотя сценарий **mysql_secure_installation** рекомендует удалить его, вы можете сохранить его для целей тестирования:

   Remove test database and access to it? [Y/n] n
   ... skipping.

8. Наконец, когда вы очищаете таблицы привилегий, **MariaDB** реализует ваши изменения:

   Reload privilege tables now? [Y/n] y
   ... Success!

Запустите MariaDB на нестандартном TCP-порте

По умолчанию **MariaDB** прослушивает **TCP-порт 3306**. Если вы хотите изменить порт по умолчанию, вам необходимо выполнить следующие шаги:

1. Откройте файл конфигурации **my.cnf** и добавьте директиву **port=num**.
2. Откройте указанный порт в вашем брандмауэре.
3. Измените порт **MariaDB** по умолчанию, определенный в политике **SELinux**.

Этот процесс относительно прост и проиллюстрирован в **упражнении 17-2**.

УПРАЖНЕНИЕ 17-2

Запустите MariaDB на нестандартном TCP-порте

Это упражнение состоит из трех частей: редактирование файла конфигурации **MariaDB**, изменение брандмауэра и изменение меток портов **SELinux**. Мы предполагаем, что вы хотите запустить **MariaDB** на **TCP-порту 3307**, а не на **3306** по умолчанию.

1. Добавьте строку **port=3307** в раздел [**mysqld**] в **/etc/my.cnf**:

   [mysqld]
   port=3307

2. Разрешите подключения к новому порту в конфигурации брандмауэра зоны по умолчанию:

   # firewall-cmd --permanent --add-port=3307/tcp
   # firewall-cmd --reload

3. Добавьте новый порт в список разрешенных портов для **MySQL** в политике **SELinux**:

   # semanage port -a -t mysqld_port_t -p tcp 3307

4. Теперь вы можете показать порты, которые **SELinux** позволяет использовать **MySQL** и **MariaDB**. С помощью следующей команды убедитесь, что она включает ваш новый порт:

   # semanage port -l | grep mysqld
   mysqld_port_t tcp 3307, 1186, 3306, 63132-63164

5. Перезапустите **MariaDB**:

   # systemctl restart mariadb

6. Подключитесь к серверу по новому порту:

 # mysql -u root -h 127.0.0.1 -P 3307 -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 5.5.35-MariaDB MariaDB Server
Copyright (c) 2000, 2013, Oracle, Monty Program Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current
input statement.
MariaDB [(none)]>

Предыдущая команда подключается к серверу **MariaDB**, работающему с указанным **IP-адресом (-h)** и портом **(-P),** в качестве пользователя **root (-u)**, идентифицируемого паролем **(-p)**. Обратите внимание, что если вы указываете хост для подключения как «**localhost**», клиент **MariaDB** связывается с локальным сервером через сокеты **Unix**, а не через соединение **TCP**.

7. Закройте сеанс, набрав **quit** или **exit**.
8. Удалите строку, добавленную в **/etc/my.cnf**, и перезапустите **MariaDB**, чтобы запустить службу через порт по умолчанию.

Если вы не измените политику **SELinux** для учета пользовательского порта, при попытке запуска **MariaDB** вы увидите следующую ошибку:

[root@server1 ~]# systemctl start mariadb.service
Job for mariadb.service failed. See 'systemctl status mariadb.service'
and 'journalctl -xn' for details.

Вы также должны увидеть эту соответствующую ошибку в файле **/var/log/mariadb/mariadb.log**:

150124 8:21:27 [ERROR] Can't start server: Bind on TCP/IP port.
Got error: 13: Permission denied

ЦЕЛЬ СЕРТИФИКАЦИИ 17.02

Управление базой данных

Системы управления реляционными базами данных, такие как **MariaDB**, хранят информацию очень структурированным образом. На самом высоком уровне существуют базы данных, которые служат контейнерами для связанных данных. В базах данных данные хранятся в таблицах, где каждый столбец представляет атрибут данных, а каждая строка представляет запись.

Концепции базы данных

Если вы никогда ранее не использовали **СУБД**, но работали с программным обеспечением для работы с электронными таблицами, таким как **LibreOffice Calc** или **Microsoft Excel**, вы можете заметить сходство с понятиями рабочих таблиц, столбцов и строк. Фактически, таблицы в базе данных могут, в некотором смысле, рассматриваться как гигантская электронная таблица со строками и столбцами, содержащими данные. Структура и организация базы данных в различные таблицы и столбцы называется схемой.

Столбцы могут обрабатывать различные типы данных, и это определяется для каждого столбца при его создании. Например, столбец может хранить числа до определенного размера или до определенного количества текстовых символов. Столбцы могут быть обязательными или нет, а некоторые имеют значение по умолчанию. При определении схемы вы также можете выразить ограничения. Например, вы можете указать, что строка в одной таблице должна иметь уникальный

идентификатор или «ссылку» на запись в другой таблице. Эти правила применяются, когда пользователи пытаются вставить или изменить какие-либо данные в базе данных.

Пользователи взаимодействуют с базой данных с помощью **команд SQL**: некоторые из этих команд создают базы данных, создают таблицы и корректируют схему таблиц; другие вставляют данные в эти таблицы, а третьи получают данные из базы данных. Мы рассмотрим основные операции многих из этих запросов в этой главе. **Таблица 17-2** суммирует наиболее важные концепции **RDBMS**.

ТАБЛИЦА 17-2 Терминология базы данных

Термин	Объяснение
Database	Коллекция связанных таблиц
Table	Структура данных, в которой данные организованы в столбцы и строки
Row or record	Один элемент внутри таблицы, содержащий данные, необходимые для этой схемы
Column	Атрибут записи, принадлежащей к определенному типу данных
Schema	Спецификация свойств всех данных в базе данных
SQL command	Удобная для чтения команда для управления базами данных, а также для добавления, удаления или извлечения данных в одной или нескольких таблицах

РИСУНОК 17-4 Перечисление всех баз данных

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)

MariaDB [(none)]>
```

Работа с базами данных

Установка **MariaDB** по умолчанию включает в себя несколько баз данных. Чтобы просмотреть список установленных баз данных, подключитесь к **MariaDB** с помощью клиента **mysql**:

 # mysql -p

Затем выполните команду **SHOW DATABASES SQL**:

MariaDB [(none)]> SHOW DATABASES;

Вывод показан на **рисунке 17-4**. Обратите внимание, что доступны четыре базы данных (три, если вы удалили тестовую базу данных в **упражнении 17-1**):

- **mysql** Внутренняя база данных для **MariaDB** для управления пользователями и разрешениями
- **information _schema** и **performance_schema** Специализированные базы данных, используемые **MariaDB** для проверки метаданных и выполнения запросов во время выполнения
- **test** Тестовая база данных

!!!! On the Job !!!!
Вы можете писать команды SQL, такие как SHOW DATABASES, прописными или строчными буквами. По соглашению документация определяет ключевые слова SQL в верхнем регистре. !!!!!!

Вы можете создать новую базу данных из клиента **mysql** с помощью команды **CREATE DATABASE *db_name***, как показано ниже. Новая база данных не содержит данных, пока вы не создадите таблицу и не добавите в нее некоторые данные:

MariaDB [(none)]> CREATE DATABASE myapp;

Практически для каждой команды в оболочке **MariaDB**, кроме команд для создания пользователей и баз данных, вы должны сначала сообщить клиенту **MariaDB**, что вы работаете в данной базе данных с помощью команды **USE**. В **MariaDB** по умолчанию подсказка оболочки сообщает, в какой базе данных вы находитесь (**none** и **mysql** в следующих примерах):

MariaDB [(none)]> USE mysql;
Database changed
MariaDB [mysql]>

ТАБЛИЦА 17-3 Типы данных MariaDB

Тип данных	Описание
INT	32-разрядное целое число
FLOAT	Число с плавающей запятой одинарной точности
VARCHAR	Строка переменной длины
TEXT	Большая строка
BLOB	Бинарный объект
DATETIME	Дата и время

Аналогично, базу данных можно удалить с помощью команды **DROP DATABASE *db_name***:

MariaDB [(none)]> DROP DATABASE test;

Работа с таблицами

База данных не очень полезна без одной или нескольких таблиц. Для сдачи экзамена **RHCE** вам необходимо «создать простую схему базы данных». Если вы хотите узнать больше об этой теме, обратитесь на веб-сайт **MariaDB (https://mariadb.com/kb/en/mariadb/create-table).**

Таблицы **MariaDB** состоят из столбцов, которые можно настроить как различные типы данных. Эти типы данных определяют, какие данные могут храниться в столбце. Вы можете представить большинство различных форматов данных с типами данных, перечисленными в Таблице 17-3.

Вы также должны добавить индексы в таблицу для извлечения данных без необходимости читать каждую строку (это называется «сканирование таблицы»). Это критично для производительности для больших таблиц. Как правило, они включают два типа индексов: уникальные индексы и вторичные индексы.

Уникальный индекс (**unique index**) должен указывать что-то уникальное в строке, например, идентификационный номер. Специальный тип уникального индекса - это индекс, созданный с помощью ключевого слова **PRIMARY KEY**, которое используется **MariaDB** для идентификации данной строки. Если вы не укажете первичный ключ, механизм хранения по умолчанию в **MariaDB** автоматически создаст первичный ключ для часто используемого столбца.

И наоборот, вторичные индексы **(secondary indexes)** не указывают уникальный элемент в строке и используются для ускорения запросов, которые основаны на ключе, отличном от первичного ключа, и избегают сканирования таблицы.

Чтобы создать новую таблицу, используйте команду **CREATE TABLE**. Вот синтаксис этой команды в ее простейшей форме:

CREATE TABLE table_name (
col_name1 INT|FLOAT|VARCHAR|TEXT|BLOB|DATETIME [NOT NULL|AUTO_INCREMENT],
col_name2 INT|FLOAT|VARCHAR|TEXT|BLOB|DATETIME [NOT NULL|AUTO_INCREMENT],
...
PRIMARY KEY (col_name1)
);

Команда определяет каждый столбец в таблице, идентифицируемый по имени и типу, и необязательное ограничение, такое как **NOT NULL**, которое не позволяет записям в столбце принимать неопределенное значение, или **AUTO_INCREMENT**, который автоматически вставляет новый уникальный номер, когда Новая запись добавлена в таблицу.

Ограничение **PRIMARY KEY** сообщает **MariaDB**, что указанный столбец является первичным ключом. Другими словами, данный столбец должен содержать только уникальные ненулевые значения.

В **таблице 17-4** приведены команды, связанные с управлением базой данных и таблицами. Каждая команда должна заканчиваться точкой с запятой. Некоторые из этих команд будут рассмотрены **в упражнении 17-3.**

ТАБЛИЦА 17-4 Команды базы данных и таблицы SQL

Команда SQL	Описание
CREATE DATABASE *db_name*	Создает базу данных ***db_name***
DROP DATABASE *db_name*	Удаляет базу данных ***db_name***
SHOW DATABASES	Перечисляет все базы данных
USE *db_name*	Следующие команды SQL окажут влияние на базу данных ***db_name***
CREATE TABLE *table_name* (...)	Создает таблицу ***table_name***
DROP TABLE *table_name*	Удаляет таблицу ***table_name***
SHOW TABLES	Перечисляет все таблицы в текущей базе данных
DESCRIBE *table_name*	Отображает схему таблицы ***table_name***

УПРАЖНЕНИЕ 17-3

Создать таблицу

В этом упражнении вы создадите простую таблицу. Начните с подключения к **MariaDB** с помощью клиента **mysql**.

1. Создайте базу данных с именем «**myapp**»:

   CREATE DATABASE myapp;

2. Скажите **MariaDB**, что следующие команды будут выполняться в базе данных **myapp**:

   USE myapp;

3. Создайте простую таблицу: список элементов управления ***widgests***, каждый с автоматически сгенерированным **ID**. Для этого используйте оператор **CREATE TABLE**:

   ```
   CREATE TABLE widgets (
           id INT AUTO_INCREMENT,
           name VARCHAR(255),
           PRIMARY KEY (id)
   );
   ```

   Обратите внимание, что столбец «**id»** помечен как первичный ключ. Существует также второй столбец («**имя (name)**»), который может содержать строку переменной длины, **до 255 символов**.

4. Показать вновь созданную таблицу с помощью **SHOW TABLES**:

   ```
   SHOW TABLES;
   +----------------+
   | Tables_in_myapp |
   +----------------+
   | widgets |
   +----------------+
   ```

5. Вы можете отобразить полную схему таблицы с помощью команды **DESCRIBE *tablename***. Он распечатает схему, которую вы ввели ранее. Вывод показан на **рисунке 17-5**.

РИСУНОК 17-5 Показать схему существующей таблицы.

```
MariaDB [myapp]> DESCRIBE widgets;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| id    | int(11)      | NO   | PRI | NULL    | auto_increment |
| name  | varchar(255) | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

MariaDB [myapp]>
```

ЦЕЛЬ СЕРТИФИКАЦИИ 17.03

Простые SQL-запросы

SQL - это специализированный язык программирования, который работает как в качестве языка манипулирования данными, для изменения данных или схемы в базе данных, так и в качестве языка запросов, для извлечения данных из базы данных.

В предыдущем разделе мы показали, как вы можете использовать команды **SQL** для управления базами данных и таблицами. В этом разделе мы представим краткое введение в несколько команд **SQL** для извлечения и вставки данных.

После создания базы данных и таблицы вы можете вносить изменения в данные с помощью ключевых слов **SQL INSERT, SELECT, UPDATE** и **DELETE**. Это основные ключевые слова **SQL** запроса, необходимые для экзамена **RHCE**.

!!!! On the Job !!!!
В компьютерном программировании операторы SQL INSERT, SELECT, UPDATE и DELETE также называются операциями «CRUD», где буквы аббревиатуры обозначают «Create, Read, Update, и Delete».
!!!!!!

Команда SQL INSERT

Оператор **INSERT** добавляет запись в таблицу. Синтаксис команды следующий:

INSERT INTO table_name (field1, field2) VALUES ('a', 'b');

Например, вы можете вставить новую запись в таблицу **widget** с помощью следующей команды:

MariaDB [myapp]> INSERT INTO widgets (id, name) VALUES (1, "widget A");
Query OK, 1 row affected (0.01 sec)

Эта команда добавляет новую запись в таблицу **widget** с целочисленным значением «**1**» в столбце **id** и строкой «**widget A**» в столбце имени.

Поскольку в **упражнении 17-3** мы определили столбец идентификатора как **AUTO_INCREMENT**, **MariaDB** автоматически присваивает уникальный и инкрементный идентификатор для следующей вставляемой строки. Следовательно, вам даже не нужно указывать поле **id** при добавлении строки:

MariaDB [myapp]> INSERT INTO widgets (name) VALUES ("widget B");
Query OK, 1 row affected (0.01 sec)

Этот оператор **SQL** добавляет новую запись в таблицу **widgets** со строкой «**widget B**» в имени столбца. MariaDB автоматически присвоит значение «**2**» полю **id**.

Поскольку мы определили столбец **id** как **PRIMARY KEY**, это означает, что каждое значение в столбце должно быть уникальным. Если вы создадите новую строку с тем же идентификатором, что и предыдущая, **MariaDB** вернет ошибку:

```
MariaDB [myapp]> INSERT INTO widgets (id, name) VALUES (2, "widget C");
ERROR 1062 (23000): Duplicate entry '2' for key 'PRIMARY'
```

Команда SELECT SQL

Поскольку у вас есть несколько записей, хранящихся в таблице **widget**, теперь вы можете использовать инструкцию **SELECT** для извлечения данных из таблицы. В простейшей форме синтаксис команды выглядит следующим образом:

```
SELECT field1, field2 FROM table_name [WHERE field2 = "value"];
```

Например, следующая команда перечисляет все строки в таблице с именем widgets:

```
MariaDB [myapp]> SELECT id, name FROM widgets;

+ ---- + ---------- +
| id | имя |
+ ---- + ---------- +
| 1 | виджет A |
| 2 | виджет B |
| 3 | виджет C |
+ ---- + ---------- +
3 rows in set (0.00 sec)
```

Вы также можете использовать подстановочный знак «**star**», чтобы указать все столбцы в таблице. Следующий оператор **SQL** эквивалентен последней команде:

```
MariaDB [myapp]> SELECT * FROM widgets;
```

Чтобы отфильтровать результаты, передайте команду условию **WHERE**. В следующем примере показано, как извлечь столбец из строки с определенным идентификатором:

```
MariaDB [myapp]> SELECT name FROM widgets WHERE id=2;
+----------+
| name |
+----------+
| widget B |
+----------+
1 row in set (0.00 sec)
```

MariaDB поддерживает множество операторов, которые вы можете включить в предложение **WHERE**. Например, оператор <> сопоставляет все записи, которые не равны данному значению.

В качестве примера следующий оператор возвращает все записи из таблицы **widgets**, чей идентификатор не равен значению **«2»**:

```
MariaDB [myapp]> SELECT * FROM widgets WHERE id<>2;
+----+----------+
| id | name |
+----+----------+
| 1 | widget A |
| 3 | widget C |
+----+----------+
2 rows in set (0.00 sec)
```

В **таблице 17-5** перечислены наиболее часто используемые операторы.

ТАБЛИЦА 17-5 Операторы MariaDB

Оператор MariaDB	Описание
=	Равный
<>	Не равный
>	Больше, чем
<	Меньше, чем
>=	Больше или равно, чем
<=	Меньше или равно, чем
LIKE	Ищет шаблон - например, **WHERE name LIKE “pattern”**
IN	Перечисляет все возможные значения для поля, например, **WHERE id IN (1,2,4)**

Команда SQL DELETE

Оператор **DELETE** работает аналогично **SELECT**, за исключением того, что удаляет соответствующие записи. Синтаксис проиллюстрирован в следующей строке:

```
DELETE FROM tablename WHERE field1 = "value";
```

Например, если вы хотите удалить **строку** из таблицы **widgets** со значением «**1**» в столбце **id**, выполните следующее:

```
MariaDB [myapp]> DELETE FROM widgets WHERE id=1;
Query OK, 1 row affected (0.01 sec)
```

Следующий запрос **SELECT** подтверждает, что соответствующая строка была удалена из таблицы:

```
MariaDB [myapp]> SELECT * widgets;
+----+----------+
| id | name |
+----+----------+
| 2 | widget B |
| 3 | widget C |
+----+----------+
2 rows in set (0.00 sec)
```

Команда SQL UPDATE

Наконец, оператор **SQL UPDATE** позволяет обновить одну или несколько строк. Эта команда немного сложнее - вы должны включить таблицу, которую вы изменяете, изменения, которые вы хотите внести, и соответствующие строки:

```
UPDATE table_name SET field1="value" WHERE field2="value";
```

Например, следующая команда устанавливает значение в столбце имени на новое значение для записи, идентификатор которой равен «**2**»:

```
MariaDB [myapp]> UPDATE widgets SET name='Widget with a new name' WHERE id=2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

Следующая оператор **SELECT** подтверждает, что изменение было применено:

```
MariaDB [myapp]> SELECT * from widgets;
+----+-----------------------+
| id | name |
```

+----+----------------------+
| 2 | Widget with a new name |
| 3 | Widget C |
+----+----------------------+
2 rows in set (0.00 sec)
Table 17-6 summarizes the SQL queries we have described so far.

ТАБЛИЦА 17-6. Таблица общих запросов SQL

Оператор SQL	Пример
INSERT	**INSERT INTO table_name (field1, field2) VALUES (“value1”, "value2”);**
SELECT	**SELECT field1, field2 FROM table_name WHERE field1=“value”**
UPDATE	**UPDATE table_name SET field1=“value” WHERE field2=“value”**
DELETE	**DELETE FROM table_name WHERE field=“value”;**

УПРАЖНЕНИЕ 17-4

Практика с простыми запросами SQL

В этом упражнении вы импортируете свободно доступную тестовую базу данных, чтобы предоставить достаточно данных, чтобы иметь возможность исследовать несколько более сложные запросы **SQL**.

1. Подключитесь к клиенту **MySQL** от имени пользователя **root**:

   # mysql –u root -p

2. Создайте новую базу данных с именем «**employees**»:

   MariaDB [(none)]> CREATE DATABASE employees;

3. Вернитесь в **shell** (с помощью команды **quit**). Мы будем использовать стандартную тестовую базу данных, доступную на носителе, который сопровождает эту книгу. Вставьте носитель, перейдите в подкаталог **Chapter17/** и скопируйте файл **employee_db-full-1.0.6.tar.bz2** на локальный диск.
4. Извлеките и импортируйте базу данных, используя следующие команды:

   # tar xvfj employees_db-full-1.0.6.tar.bz2
   # cd employees_db
   # cat employees.sql | mysql -u root -p employees

5. Дождитесь загрузки файлов и убедитесь, что новые таблицы существуют, как показано ниже:

   # mysql -u root -p
   MariaDB [(none)]> USE employees;

```
MariaDB [employees]> SHOW TABLES;
+----------------------+
| Tables_in_employees |
+----------------------+
| departments          |
| dept_emp             |
| dept_manager         |
| employees            |
| salaries             |
| titles               |
+----------------------+
6 rows in set (0.00 sec)

MariaDB [employees]>
```

6. Найдите схему таблицы отделов:

```
MariaDB [employees]> DESCRIBE departments;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| dept_no   | char(4)     | NO   | PRI | NULL    |       |
| dept_name | varchar(40) | NO   | UNI | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

MariaDB [employees]>
```

7. Показать все содержимое таблицы отделов:

```
MariaDB [employees]> SELECT * FROM departments;
+---------+--------------------+
| dept_no | dept_name          |
+---------+--------------------+
| d009    | Customer Service   |
| d005    | Development        |
| d002    | Finance            |
| d003    | Human Resources    |
| d001    | Marketing          |
| d004    | Production         |
| d006    | Quality Management |
| d008    | Research           |
| d007    | Sales              |
+---------+--------------------+
9 rows in set (0.00 sec)

MariaDB [employees]>
```

Теперь попробуйте чуть более сложный пример. Вы будете искать сотрудника с самой высокой зарплатой. Сначала отобразим схему таблицы «зарплаты (**salary**)»:

```
MariaDB [employees]> DESCRIBE salaries;
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+-----------+---------+------+-----+---------+-------+
| emp_no    | int(11) | NO   | PRI | NULL    |       |
| salary    | int(11) | NO   |     | NULL    |       |
| from_date | date    | NO   | PRI | NULL    |       |
| to_date   | date    | NO   |     | NULL    |       |
+-----------+---------+------+-----+---------+-------+
4 rows in set (0.00 sec)

MariaDB [employees]>
```

8. Определите сотрудника с самой высокой зарплатой. Для этого мы вводим новое предложение, поле **ORDER BY**, которое упорядочивает результаты запроса **SELECT** на основе значений указанного столбца. Необязательное ключевое слово **DESC** сортирует результаты в порядке убывания. Кроме того, количество записей, возвращаемых запросом, может быть ограничено максимальной суммой с помощью предложения **LIMIT *num***.

9. Результат показан здесь:

```
MariaDB [employees]> SELECT * FROM salaries ORDER BY salary DESC LIMIT 5;
+--------+--------+------------+------------+
| emp_no | salary | from_date  | to_date    |
+--------+--------+------------+------------+
|  43624 | 158220 | 2002-03-22 | 9999-01-01 |
|  43624 | 157821 | 2001-03-22 | 2002-03-22 |
| 254466 | 156286 | 2001-08-04 | 9999-01-01 |
|  47978 | 155709 | 2002-07-14 | 9999-01-01 |
| 253939 | 155513 | 2002-04-11 | 9999-01-01 |
+--------+--------+------------+------------+
5 rows in set (1.16 sec)

MariaDB [employees]>
```

10. Из результатов последнего запроса видно, что у сотрудника с идентификатором **43624** зарплата **составляет 158 220 долларов**.
11. Следующим шагом является поиск сведений о таком сотруднике в соответствующей таблице «**employee**». Для этого выполните запрос **SELECT** с оператором **WHERE**, чтобы отобразить запись для сотрудника с идентификатором **43624**:

```
MariaDB [employees]> SELECT * FROM employees WHERE emp_no=43624;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  43624 | 1953-11-14 | Tokuyasu   | Pesch     | M      | 1985-03-26 |
+--------+------------+------------+-----------+--------+------------+
1 row in set (0.00 sec)

MariaDB [employees]>
```

!!!! On the Job !!!
Чтобы объединить данные из нескольких таблиц, вы можете использовать предложение SQL join, а не пошаговую процедуру, показанную в упражнении 17-4. Например, окончательный результат в упражнении 17-4 можно получить одним запросом:
SELECT * FROM employees NATURAL JOIN salaries ORDER BY salary DESC LIMIT 1;
Однако это выходит за рамки экзамена RHCE.
!!!!!

ЦЕЛЬ СЕРТИФИКАЦИИ 17.04
Безопасный MariaDB

При установке по умолчанию **MariaDB** принимает подключения от любой системы в сети. Доступ предоставляется пользователю **root без пароля**.

Понятно, что это не безопасная конфигурация. В предыдущем разделе мы объяснили, как защитить установку **MariaDB** с помощью сценария **mysql_secure_installation**. Тем не менее, есть еще много возможностей для безопасной установки.

У вас могут быть приложения, которые необходимо подключить к **MariaDB**. Например, веб-сервису может потребоваться доступ. Хотя вы можете поддерживать удаленный доступ некоторыми системами, вы должны убедиться, что доступ запрещен для всех других хостов. **MariaDB** предоставляет гибкую схему разрешений, которая позволяет вам указывать все типы команд, которые пользователь может выполнять в системе.

Безопасность на базе хоста

Вы должны начать с запрета удаленного доступа к **MariaDB**, если это возможно. Кроме того, вы можете ограничить доступ только к системам, которые должны иметь право на подключение к нему. На этот счет в **/etc/my.cnf** доступны две ключевые директивы:

- **skip-network** Запрещает **MariaDB** прослушивать любое **TCP-соединение**. Это не ограничивает доступ из локальной системы через **сокеты Unix**.
- **bind-address** Позволяет **MariaDB** прослушивать **определенный IP-адрес**. Если вы установите эту директиву на 0.0.0.0, **MariaDB** будет прослушивать соединения на всех локальных **IPv4-адресах**. Это значение по умолчанию. Если вы установите его **::, MariaDB** будет прослушивать трафик на все адреса **IPv4** и **IPv6**. В системах с несколькими интерфейсами и IP-адресами вам может потребоваться, чтобы **MariaDB** прослушивала только один конкретный IP-адрес.

Конечно, вы также можете использовать **firewall-cmd** для ограничения доступа к **MariaDB**. В следующем примере устанавливается правило расширенного межсетевого экрана, разрешающее подключения только с хоста с **IP-адресом 192.168.122.1**:

 # firewall-cmd --permanent --add-rich-rule='rule family=ipv4 source □
address=192.168.122.1 service name=mysql accept'
 # firewall-cmd --reload

If you need to enable remote access to MariaDB for all hosts, run the following:
 # firewall-cmd --permanent --add-service=mysql
 # firewall-cmd –reload

Безопасность на основе пользователя

Доступ к **MariaDB** поддерживается через внутреннюю базу данных пользователей и привилегии, известные как «**grants**».

В клиенте **MariaDB mysql** именем пользователя по умолчанию является имя пользователя, с которым вы вошли в систему. Итак, если вы вошли на сервер как **root**, это имя пользователя по умолчанию. Вы можете подключиться как конкретный пользователь с помощью **ключа -u**. Вы можете передать **-p**, чтобы запросить у клиента **MariaDB пароль**, и **-P**, чтобы передать пользовательский порт **TCP**. Последний аргумент, который является необязательным, указывает имя базы данных для подключения.

Например, чтобы подключиться к базе данных **myapp** на **сервере 192.168.122.1** через **порт 3307** с именем пользователя **myuser** и паролем **changeme**, выполните следующую команду:

 # mysql –u myuser –pchangeme –P 3307 –h 192.168.122.1 myapp

Обратите внимание, что между ключом -p и паролем не должно быть пробела.

Управление пользователями MariaDB

MariaDB использует внутреннюю базу данных **mysql** для управления пользователями и разрешениями. Чтобы получить список текущих пользователей, запустите эти операторы **SQL**:

MariaDB [(none)]> USE mysql;
MariaDB [mysql]> SELECT user, host from mysql.user;

Вы можете создать нового пользователя с помощью команды **CREATE USER**. Синтаксис иллюстрируется в следующем примере:

CREATE USER appuser@'192.168.122.1' IDENTIFIED BY 'changeme';

Эта команда **SQL** создает пользователя с именем «**appuser**», который может подключаться только с хоста **с IP-адресом 192.168.122.1** с паролем «**changeme**». Новым пользователям не назначаются никакие привилегии, поэтому вы должны специально назначить разрешения, на что пользователь должен иметь право.

Управление привилегиями пользователей

Каждому пользователю может быть назначен список разрешений («**grants**»), которые вы можете отобразить с помощью команды **SQL SHOW GRANTS [FOR username]**. Пример вывода показан на **рисунке 17-6**.

Сосредоточитьмся на первой строке вывода. Это говорит нам о том, что пользователю **root**, подключающемуся с локального хоста, предоставляются ВСЕ привилегии (**ALL PRIVILEGES**) для всех баз данных и всех таблиц **(*.*)** С дополнительным разрешением, известным как **GRANT OPTION**, которое позволяет этому пользователю создавать новых пользователей и назначать им привилегии.

РИСУНОК 17-6 Стандартные grants для пользователя root MariaDB

```
MariaDB [(none)]> SHOW GRANTS;
+---------------------------------------------------------------------+
| Grants for root@localhost                                           |
+---------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
| GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION        |
+---------------------------------------------------------------------+
2 rows in set (0.00 sec)

MariaDB [(none)]>
```

!!!!! Exam watch !!!!!
Если вам нужна дополнительная информация о синтаксисе оператора GRANT, запустите SHOW GRANTS. Он отображает права доступа текущего подключенного пользователя, который можно использовать в качестве шаблона для изменения прав доступа других учетных записей пользователей.
!!!!

Список наиболее распространенных привилегий приведен в **таблице 17-7.**

Каждый оператор **GRANT** применяется либо глобально **(*.*)**, к данной базе данных (***db_name.****), либо к заданной таблице (***db_name.table_name***). Операторы **GRANT** добавляют больше привилегий; чтобы отозвать привилегию, используйте команду **REVOKE**.

Чтобы применить это на практике, мы создадим пользователя с именем «**appowner**», который сможет входить в **MariaDB** с любого хоста **('%')** с полными привилегиями в базе данных **myapp** и «**password123**» в качестве пароля:

```
MariaDB [(none)]> CREATE USER appowner@'%' IDENTIFIED BY 'password123';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON myapp.* TO appowner@'%';
```

Предыдущие команды могут быть объединены в одну команду **GRANT**. Другими словами, следующее действие имеет тот же эффект, что и предыдущие:

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON myapp.* TO appowner@'%'
-> IDENTIFIED BY 'password123';
```

ТАБЛИЦА 17-7 Предоставление привилегий

Предоставить привилегию	Описание
ALL PRIVILEGES	Предоставляет все привилегии, за исключением **GRANT OPTION**.
WITH GRANT OPTION	Позволяет создать нового пользователя и назначить права доступа до уровня текущего пользователя.
CREATE	Дает разрешение на создание новых баз данных и таблиц.
DROP	Позволяет удалять базы данных и таблицы.
ALTER	Используется для изменения таблицы, например, для добавления или удаления столбцов.
DELETE	Используйте оператор **SQL DELETE** для удаления строк из таблицы.
INSERT	Используйте оператор **SQL INSERT** для создания строк в таблице.
SELECT	Используйте оператор **SQL SELECT** для извлечения данных из таблицы.
UPDATE	Используйте оператор **SQL UPDATE** для изменения строк в таблице.

Если вы хотите, чтобы пользователь мог войти в **MariaDB** с локального хоста через соединения **TCP и сокеты Unix**, вам нужно дважды выполнить команду **GRANT** (разрешения) и указать хост как **127.0.0.1** и **localhost**. Пример этого синтаксиса приведен в **упражнении 17-5**.

MariaDB хранит привилегии внутри себя в базе данных, называемой «**mysql**». Когда вы вносите изменения в разрешения пользователя, они отражаются в таблице базы данных. Однако **MariaDB** не реализует эти изменения, пока вы не «**flush** (**сбросите)**» эти привилегии (или не перезапустите службу). В командной строке **MariaDB** необходимо выполнить: **FLUSH PRIVILEGES**:

```
MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

Затем вы можете проверить, работает ли новая учетная запись пользователя, подключившись к клиенту **mysql** и посмотртеть права текущего пользователя.

Удаление пользователей MariaDB

Чтобы удалить пользователя **MariaDB**, запустите оператор **DROP USER**. Пример показан далее:

```
MariaDB [(none)]> DROP USER appowner;
```

Эта команда имеет немедленный эффект и не требует сброса привилегий пользователя.

!!!! Exam watch !!!!
Не забудьте запустить FLUSH PRIVILEGES после изменения пользовательских разрешений (grants).
!!!!!

УПРАЖНЕНИЕ 17-5

Практика MariaDB Пользовательские разрешения

В этом упражнении мы предполагаем, что вы выполнили **упражнение 17-3** и создали базу данных «**myapp**». Вы создадите двух пользователей **MariaDB**:

- **apprw** Этот пользователь идентифицируется паролем «**pass123**» и имеет права на чтение, запись, обновление и удаление всех таблиц в базе данных **myapp**. Пользователь может войти в систему с любого хоста.
- **appro** Этот пользователь идентифицируется паролем «**pass456**» и имеет права на чтение всех таблиц базы данных **myapp**. Пользователь может войти только с локального хоста.

1. Подключитесь к клиенту **MySQL** от имени пользователя **root**:

   # mysql –u root -p

2. Создайте пользователя **apprw** с помощью следующей команды:

   MariaDB [(none)]> GRANT SELECT, INSERT, UPDATE, DELETE ON myapp.*
   -> TO apprw@'%' IDENTIFIED BY 'pass123';

   Не забудьте запустить **FLUSH PRIVILEGES** после изменения пользовательских грантов.

3. Создайте соответствующего пользователя:

   MariaDB [(none)]> GRANT SELECT ON myapp.* TO appro@'127.0.0.1' IDENTIFIED
   -> BY 'pass456';
   MariaDB [(none)]> GRANT SELECT ON myapp.* TO appro@'localhost';

4. Примените новые привилегии:

   MariaDB [(none)]> FLUSH PRIVILEGES;

5. Откройте новое окно терминала и убедитесь, что новые пользователи могут подключаться к **MariaDB** с помощью клиента **mysql**. Например, чтобы подключиться как пользователь с правами доступа, выполните следующее:

   # mysql -u appro -h localhost -ppass456 myapp

6. Запустите простой запрос **SELECT**, например, следующий:

   MariaDB [myapp]> SELECT * from widgets;

   Эта команда работает для одобрения и одобрения пользователей?

7. Запустите **INSERT**-запрос:

   MariaDB [myapp]> INSERT INTO widgets (name) VALUES ("test widget");

   Эта команда работает для одобрения и одобрения пользователей?

8. Выйдите из клиента **mysql** с помощью команды **quit**.

ЦЕЛЬ СЕРТИФИКАЦИИ 17.05

Резервное копирование и восстановление базы данных

MariaDB поставляется с программой резервного копирования **mysqldump**, которая преобразует все содержимое одной или нескольких таблиц или баз данных в **операторы SQL**, которые потребуются для их повторного создания.

Данные также можно экспортировать, перенаправив результат запроса **SELECT** в файл. Это можно сделать с помощью оператора **SELECT INTO OUTFILE** или путем выполнения запроса от команды **mysql** и перенаправления вывода в файл.

Резервное копирование и восстановление с mysqldump

Команда **mysqldump** выводит **операторы SQL** в стандартный вывод. Чтобы сделать этот вывод полезным, вы можете перенаправить вывод в файл **.sql** или зафиксировать любые ошибки, отправленные в **stderr**. Например, вы можете сохранить содержимое таблицы виджетов, созданной ранее, с помощью следующей команды:

[root@server1 ~]# mysqldump -u appowner -p myapp widgets > /tmp/widgets.sql

Если **mysqldump** возвращает какие-либо ошибки, убедитесь, что база данных и таблица существуют, и что у пользователя есть разрешения на доступ к базе данных и получение ее содержимого.

На **рисунке 17-7** показано содержимое файла, созданного предыдущей командой после некоторые строки комментариев были удалены.

РИСУНОК 17-7 Резервная копия, сгенерированная mysqldump

```
[root@server1 ~]# cat /tmp/widgets.sql
--
-- Table structure for table `widgets`
--

DROP TABLE IF EXISTS `widgets`;
CREATE TABLE `widgets` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;

--
-- Dumping data for table `widgets`
--

LOCK TABLES `widgets` WRITE;
INSERT INTO `widgets` VALUES (1,'widget A'),(2,'widget B'),(3,'widget C');
UNLOCK TABLES;
[root@server1 ~]#
```

Первая команда - это оператор **DROP TABLE IF EXISTS**. Эта строка удаляет таблицу виджетов, только если она уже существует, чтобы избежать каких-либо сообщений об ошибках, если таблица отсутствует.

Далее вы увидите команду **CREATE TABLE**, которая должна напоминать команду из **упражнения 17-3**.

Следующие операторы **LOCK** и **UNLOCK** не позволяют другим командам изменять содержимое таблицы, пока ее содержимое восстанавливается с помощью команды **INSERT**.

С помощью файла резервной копии, созданного **mysqldump**, вы можете заново создать каждую запись в вашей базе данных из этого файла. Например, если вы хотите импортировать эту резервную копию в базу данных с **именем myapp_restored**, выполните следующие три шага:

1. Создайте новую базу данных:

   MariaDB [(none)]> CREATE DATABASE myapp_restored;

2. Добавить разрешения для учетной записи владельца::

   MariaDB [(none)]> GRANT ALL PRIVILEGES ON myapp_restored.* TO appowner@'%';

3. Выполните содержимое файла дампа из клиента **mysql**:

   MariaDB [(none)]> USE myapp_restored;
   MariaDB [(нет)]> SOURCE /tmp/widgets.sql

   В качестве альтернативы последний шаг может быть выполнен из оболочки **Bash** с помощью следующей команды:

   # cat /tmp/widgets.sql | mysql -u appowner -p myapp_restored

Пока что мы сделали резервную копию и восстановили одну таблицу. Однако **mysqldump** также может создавать резервные копии всей базы данных. Например, следующая команда создает полную резервную копию базы данных сотрудников (employee):

 # mysqldump -u root -p employees > /tmp/employees.sql

Если вы хотите выполнить резервное копирование всех баз данных в вашей системе **MariaDB**, замените имя базы данных на флаг **--all-database**:

 # mysqldump --all-databases -u root -p > /tmp/full-backup.sql

Резервное копирование данных в текстовый файл

Если у вас большой объем данных, вы можете создать дамп данных в текстовом файле (например, для импорта другим приложением). Есть два способа создать файл с определенными строками в нем: с помощью оператора **SELECT INTO OUTFILE и флага -e** команды **mysql**.

SELECT INTO OUTFILE создает файл на сервере, который содержит запрошенные строки таблицы. Например, следующая команда выбирает все идентификаторы и имена сотрудников и сохраняет результат в файле **/tmp/employees.data**:

MariaDB [employees]> SELECT emp_no, first_name, last_name FROM employees
-> INTO OUTFILE '/tmp/employees.data';
Query OK, 300024 rows affected (0.12 sec)

В качестве другого варианта вы можете использовать стандартное перенаправление вывода и **флаг -e** для команды **mysql**:

**# mysql employees -e "SELECT emp_no, first_name, last_name **
FROM employees" > /tmp/employees.data

Вы должны знать, что хотя команда **mysqldump** может выполнять резервное копирование и восстановление данных и схемы базы данных, команды, показанные в этом разделе, не могут создавать резервные копии схемы. Кроме того, не существует стандартной и простой процедуры для восстановления данных, сгенерированных оператором **SELECT INTO OUTFILE**, в базу данных или таблицу.

РЕЗЮМЕ СЕРТИФИКАЦИИ

MariaDB - это очень популярная система управления реляционными базами данных, созданная и полностью совместимая с **MySQL**. **RPM-пакет mariadb-server** устанавливает компоненты сервера, тогда как клиент и библиотеки включены в пакеты **mariadb** и **mariadb-libs**.

Конфигурация по умолчанию в **RHEL 7** работает «**из коробки**», и никаких изменений в файле конфигурации **/etc/my.cnf** не требуется. Однако, как минимум, вы должны обезопасить установку, запустив скрипт **mysql_secure_installation**.

Как и во многих других системах управления реляционными базами данных, база данных **MariaDB** организована в разные таблицы. Каждая таблица состоит из столбцов различных типов данных и строк (или записей). Спецификация свойств всех данных в базе данных называется схемой. Базы данных и таблицы могут быть созданы с помощью операторов **CREATE DATABASE** и **CREATE TABLE**. Другие операторы **SQL** выполняют наиболее распространенные операции «**CRUD**» (создание, чтение, обновление, удаление (**create, read, update, delete**)). Это **INSERT**, **SELECT**, **DELETE** и **UPDATE**.

MariaDB поддерживает некоторые директивы безопасности на основе хоста в файле конфигурации **/etc/my.cnf**, такие как **skip-network** для отключения TCP-соединения и **bind-address**, для прослушивания соединений по определенному **IP-адресу**. Доступ к серверу также может быть ограничен в брандмауэре на основе локальной зоны.

Доступ пользователя осуществляется с помощью инструкции **GRANT** (разрешение). Эта команда может назначить определенный набор разрешений каждому пользователю, либо для каждой базы данных, либо для каждой таблицы. После изменения пользовательских разрешений вы должны применить изменения с помощью команды **FLUSH PRIVILEGES**.

Команда **mysqldump** может выполнить полное резервное копирование содержимого и схемы одной таблицы, базы данных или всех баз данных в системе. Резервная копия может быть сохранена в файл, который может быть передан клиенту **MySQL** в качестве сценария для восстановления резервной копии в **MariaDB**.

Пару минут проверки

Вот некоторые из ключевых моментов целей сертификации в главе 17.

Введение в MariaDB

- **MariaDB** - это **СУБД**, включенная в базовые репозитории **RHEL7**. Это разработанная сообществом версия **MySQL**, выпущенная под лицензией **GPL**.
- Пакет сервера в RPM **mariadb-server**, а пакет клиент находится в RPM-версии **mariadb**.
- Основной конфигурационный файл **MariaDB - /etc/my.cnf**.
- Директива **port=*num*** в **/etc/my.cnf** может использоваться для запуска службы на другом порту.
- Сценарий **mysql_secure_installation** можно использовать для защиты сервера **MariaDB**
- установка путем назначения пароля для пользователя **root MariaDB**, отключение удаленного входа в систему, удаление анонимных пользователей и удаление тестовой базы данных по умолчанию.

Управление базой данных

- Базы данных хранят данные в таблицах.
- Таблицы являются своего рода гигантской электронной таблицей со строками и столбцами, содержащими данные.
- Схема определяет, как данные организованы и структурированы в базу данных.
- Команды **SQL CREATE DATABASE** и **CREATE TABLE** создают новый база данных и таблица соответственно.

Простые SQL-запросы

- Данные могут быть извлечены, вставлены, отредактированы и изменены с помощью **SQL SELECT**, **INSERT**, **UPDATE** и **DELETE**.
- Ключевое слово **WHERE** фильтрует результаты или применяет условие к оператору **SQL**.
- Ключевое слово **ORDER BY** сортирует записи запроса по возрастанию или убыванию (с ключевым словом **DESC**) порядок.
- Ключевое слово **LIMIT** ограничивает количество записей, возвращаемых запросом.

Безопасность MariaDB

- Директива **skip-network** в **/etc/my.cnf** запрещает TCP-соединения с базой данных и разрешает доступ только через **сокеты Unix**.
- Директива **bind-address** указывает **IP-адрес**, который должна прослушивать **MariaDB** для соединений.
- Пользователям **MariaDB** может быть назначен список разрешений («**grants**») с помощью команды **GRANT.**
- Права доступа должны применяться с командой **FLUSH PRIVILEGES.**

Резервное копирование и восстановление базы данных

- Резервные копии всей базы данных или отдельных таблиц могут быть сделаны с помощью команды **mysqldump.**
- Базы данных могут быть восстановлены из файла **SQL** (например, созданного **mysqldump**) с помощью перенаправления его содержимое в команду **mysql**.
- Данные могут быть сохранены в файл с помощью оператора **SELECT INTO OUTFILE.**

САМОПРОВЕРКА

Следующие вопросы помогут оценить ваше понимание материалов, представленных в этой главе. Поскольку на экзаменах Red Hat не появляется вопросов с несколькими вариантами ответов, вопросы с несколькими вариантами ответов не появляются в этой книге. Эти вопросы исключительно проверяют ваше понимание главы. Это нормально, если у вас есть другой способ выполнения задачи. Получение результатов, а не запоминание пустяков - вот на что рассчитывает Red Hat экзамены. На многие из этих вопросов может быть более одного ответа.

Введение в MariaDB

1. Какой **RPM-пакет** предоставляет сервер **MariaDB**?
   _____

2. Какие четыре действия выполняет скрипт **mysql_secure_installation?**
   _____

3. Какая директива конфигурации запускает **MariaDB** на **TCP-порту 33066**?
   _____

Управление базой данных

4. Какую команду **SQL** вы бы использовали для создания базы данных с именем **foo**?
   _____

5. Какую команду **SQL** вы бы использовали для создания таблицы с именем **person**, содержащей два столбца для сохранения имени и фамилии?
   _____

Простые SQL-запросы

6. Какую команду **SQL** вы бы запустили, чтобы напечатать все записи в таблице зарплат, где значение в графе зарплата больше или равна 10000?
   _____

7. Какую команду **SQL** вы бы запустили, чтобы вставить значения 7 и «финансы» в идентификатор столбца в таблице отделов?
   _____

8. Какую команду **SQL** вы бы запустили, чтобы удалить все записи в таблице сотрудников, где столбец **last_name** равен «**Smith**»?
   _____

9. Какую команду **SQL** вы бы запустили, чтобы изменить значение столбца **first_name** на «**Adam**» в таблица сотрудников, где столбец **id равен 5**?
   _____

Безопасность MariaDB

10. Чтобы отключить все TCP-соединения, какую директиву вы бы включили в **/etc/my.cnf**?
    _____

11. Какую команду вы бы использовали, чтобы настроить пользователя с именем «**redhat**» с паролем «**redhat**»? Также, предоставить этому пользователю доступ только для чтения к таблице с именем **bar** в базе данных **foo** и предоставить только доступ с IP-адреса 192.168.1.1.
    _____

12. Как вы увидите, какие привилегии у вас есть, когда пользователь вошел в клиент **MariaDB**?
    _____

Резервное копирование и восстановление базы данных

13. Что такое команда для резервного копирования всей базы данных **foo** в текстовый файл /tmp/foo.sql?
    _____

ВОПРОСЫ ЛАБОРОТОРНО РАБОТЫ

Некоторые из этих лабораторных работ включают в себя упражнения по настройке. Вы должны делать эти упражнения только на тестовых машинах. Предполагается, что вы выполняете эти упражнения на виртуальных машинах, таких как KVM. В этой главе также предполагается, что вы можете изменять конфигурацию физической хост-системы для этих виртуальных машин.

Red Hat представляет свои экзамены в электронном виде. По этой причине лаборатории в этой главе доступны в подкаталог **Chapter17/** на носителе, сопровождающего книгу. Если вы еще не настроили **RHEL 7** в системе, обратитесь к **Главе 1** за инструкциями по установке.

Ответы для каждой лаборатории следуют за ответами самопроверки.

Лучше всего начинать эти лаборатории с новой установки **MariaDB**. Если вы хотите удалить существующую установку **MariaDB** и начать заново, выполните следующие команды:

```
# yum erase mariadb-server
# rm –rf /var/lib/mysql/*
# rm -f /etc/my.cnf
```

Лабораторная работа 1

Установите **MariaDB** на **server1.example.com**. Установите пароль для **root** «**letmein**» и разрешите **root-доступ** через **локальные сокеты** и **TCP-соединения со всех хостов**. Убедитесь, что сервис **MariaDB** запускается автоматически при загрузке. Проверьте подключение с удаленного компьютера.

Лабораторная работа 2

Скопируйте файл базы данных сотрудников на локальный диск. Этот файл находится в каталоге **Chapter17/ DVD-диска**, прилагаемого к этой книге. Импортируйте содержимое базы данных в **MariaDB** с помощью следующих команд:

 # tar xvfj employees_db-full-1.0.6.tar.bz2
 # cd employees_db
 # cat employees.sql | mysql -u root -p employees

Создайте нового пользователя **labuser** с паролем **changeme**. Предоставьте новому пользователю доступ только для чтения со всех хостов ко всем таблицам в базе данных сотрудников, за исключением таблицы **salaries**.

Лабораторная работа 3

Используя базу данных сотрудников, импортированную в **Лабораторной работе 2**, выполните запросы **SELECT**, чтобы ответить на следующие вопросы:

1. Сколько сотрудников (**employees**) родилось (**born**) 31 октября 1963 года?
2. Сколько работниц (**female**) было рождено (**born**) 20 октября 1963 года?
3. В какой день родился (**born**) самый молодой сотрудник?
4. Какова должность (**job title**) и зарплата (**salary**) **Eran Fiebach**?

Лабораторная работа 4

Вставьте в базу данных сотрудников следующую информацию для нового сотрудника:

1. Новым сотрудником является **Julia Chan**, родившаяся (**born**) 9 июня 1990 года. Она была принята на работу 1 июня 2015 года и присвоила сотруднику 500 000 человек.
2. Ее должность - старший инженер (**Senior Enginee**), она работает в отделе развития (**Development department**).
3. Ее зарплата (**salary**) составляет 60 000 долларов.

Лабораторная работа 5

Резервное копирование всего содержимого (структура и данные) таблицы сотрудников из базы данных сотрудников. Сохраните результат в сжатый файл с именем **emp.sql.gz** в каталоге **/root.**

Как пользователь **root** в **MariaDB**, создайте новую базу данных **emp_restored** и импортируйте резервную копию в эту базу данных.

Убедитесь, что число сотрудников, родившихся 31 октября 1963 года, такое же, как в лаборатории 3.

ОТВЕТЫ НА САМОПРОВЕРКУ

Введение в MariaDB

1. **RPM-пакет mariadb-server** устанавливает сервер **MariaDB**.
2. Сценарий **mysql_secure_installation** устанавливает пароль для пользователя root MariaDB, отключает удаленный входит в систему, удаляет анонимных пользователей и удаляет тестовую базу данных по умолчанию.
3. Директива **port=33066** в **/etc/my.cnf** запускает **MariaDB** на **TCP-порту 33066**. Вам также потребуется настроить локальный брандмауэр и настроить политику **SELinux** по умолчанию, чтобы **MariaDB** могла принимать соединения на этом порту.

Управление базой данных

4. Следующая команда **SQL** создает базу данных с именем **foo**:

   CREATE DATABASE foo;

5. Следующая команда создает таблицу с именем person, с двумя столбцами для хранения первого и последнего названия:

   CREATE TABLE person (

first_name VARCHAR(255),
last_name VARCHAR(255)
);

Простые SQL-запросы

6. Следующий оператор **SQL** печатает все записи в таблице зарплаты, где значение в зарплата в столбце больше или равна **10000**:

SELECT * FROM salaries WHERE salary >=10000;

7. Следующий оператор **SQL** добавляет запись со значениями **7 и «finance»** в столбцах **id** и **department** таблицы **departments**:

INSERT INTO departments (id, department) VALUES (7, "finance");

8. Следующая инструкция **SQL** удаляет все записи в таблице сотрудников, где last_name столбец равен «Смит»:

DELETE FROM employees WHERE last_name="Smith";

9. Следующий оператор SQL изменяет значение столбца **first_name** на «**Adam**» в таблице **employees**, где столбец идентификатора (**id**) равен 5:

   UPDATE employees SET first_name="Adam" WHERE id=5;

Безопасный MariaDB

10. Чтобы отключить все удаленные **TCP-соединения**, добавьте директиву skip-network в раздел **[mysqld] /etc/my.cnf.**
11. Следующая команда настраивает пользователя с именем «**redhat**» с паролем «**redhat**» только для чтения доступ к таблице с именем **bar** в базе данных **foo** только с **IP-адреса 192.168.1.1**:

    GRANT SELECT ON foo.bar TO redhat@192.168.1.1 IDENTIFIED BY 'redhat';

    Не забудьте выполнить **FLUSH PRIVILEGES**, чтобы изменения вступили в силу.

12. Чтобы получить список прав текущего пользователя, введите команду **SHOW GRANTS**.

Резервное копирование и восстановление базы данных

13. Следующая команда выполняет резервное копирование всей базы данных foo в текстовый файл /tmp/foo.sql:

    # mysqldump -uuser -ppass foo> /tmp/foo.sql

Ответы Лабораторной работы

Лабораторная работа 1

Эта лабораторная работа - тренировка навыков - практикуйте ее до тех пор, пока вы не сможете делать это, не задумываясь. Установите пакет **mariadb-сервер**, запустите и включите службу **MariaDB**, запустите **mysql_secure_installation** и убедитесь, что локальный брандмауэр разрешает подключения **MySQL**.

Затем подключитесь, как **root-пользователь MariaDB** к клиенту **mysql** с локального хоста и выполните следующие команды:

GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'letmein' ↵
WITH GRANT OPTION;

FLUSH PRIVILEGES;

Для проверки подключитесь к серверу базы данных с удаленного хоста:

 # mysql -h 192.168.122.50 -uroot –pletmein

Лабораторная работа 2

Первая часть этой лаборатории была рассмотрена в **упражнении 17-4**.

Чтобы создать нового пользователя и назначить необходимые разрешения, выполните следующие команды **SQL**:

GRANT SELECT ON employees.departments TO labuser@'%' IDENTIFIED BY'changeme';
GRANT SELECT ON employees.dept_emp TO labuser@'%';
GRANT SELECT ON employees.dept_manager TO labuser@'%;
GRANT SELECT ON employees.employees TO labuser@'%';
GRANT SELECT ON employees.titles TO labuser@'%';
FLUSH PRIVILEGES;

Лабораторная работа 3

Запросы в вопросе 4 этой лабораторной работы могут быть решены с помощью одного объединённого **SQL-запроса**. Однако, объединение пунктов **SQL-запроса** выходят за рамки экзамена **Red Hat**. Следовательно, мы предоставили ответы, используя простые **SELECT** ключевые слова.

Чтобы изучить структуру базы данных сотрудников, используйте команды **SHOW TABLES** и **DESCRIBE *table_name***.

1. Выполните следующий запрос, чтобы найти всех сотрудников, родившихся 31 октября 1963 года:

   SELECT * FROM employees WHERE birth_date='1963-10-31';

   Этот запрос должен вернуть 61 запись.

2. Второй вопрос аналогичен предыдущему, но требует второго условия в пункте **WHERE**:

   SELECT * FROM employees WHERE birth_date='1963-10-20' AND gender='F';

   Запрос должен вернуть 25 записей.

3. Чтобы найти самого младшего сотрудника, извлеките все первые несколько записей из таблицы сотрудников, отсортированной по данным о рождении в порядке убывания:

   SELECT * FROM employees ORDER BY birth_date DESC LIMIT 5;

   Самые молодые сотрудники родились 1 февраля 1965 года.

4. Этот вопрос требует ответа на несколько вопросов. Сначала найдите соответствующую запись для **Eran Fiebach** в таблице **employee**:

   SELECT * FROM employees WHERE first_name="Eran" AND last_name="Fiebach";

   Этот запрос должен вернуть номер сотрудника **50714** для **Eran Fiebach**. Затем получите название должности используя эту информацию:

   SELECT * FROM titles WHERE emp_no='50714';

   Название работы, возвращаемое запросом, является **Technique Leader**. Последний шаг - найти информацию о зарплате по номеру сотрудника:

SELECT * FROM salaries WHERE emp_no='50714';

Этот запрос должен вернуть 14 зарплат **Eran Fiebach**. Вы должны найти, что ее начальная зарплата была 40000 долларов США, в то время как текущая зарплата составляет 57 744 доллара США.

Лабораторная работа 4

Как обсуждалось в ответах к лабораторной работе 3, вам может потребоваться изучить структуру базы данных, используя команды **SHOW TABLES** и **DESCRIBE *table_name.***

Затем добавьте запись для нового сотрудника в таблицу **employees**:

INSERT INTO employees (emp_no, birth_date, first_name, last_name, gender, hire_date) VALUES ('500000', '1990-06-09', 'Julia', 'Chan', 'F', '2015-06-01');

Затем добавьте название вакансии:

INSERT INTO titles (emp_no, title, from_date, to_date) VALUES ('500000', 'Senior Engineer', '2015-06-01', '9999-01-01');

Обратите внимание на специальную **дату 9999-01-01**, чтобы указать, что это текущая запись для сотрудника.

Чтобы назначить нового сотрудника в отдел разработки, нам нужен код отдела. следующий запрос говорит нам, что это **d005**:

SELECT * FROM departments;

С помощью этой информации мы назначаем сотрудника в отдел развития:

INSERT INTO dep_emp (emp_no, dept_no, from_date, to_date) VALUES ('500000', «d005», «2015-06-01», «9999-01-01»);

Последний шаг состоит из добавления информации о зарплате:

INSERT INTO salaries (emp_no, salary, from_date, to_date) VALUES ('500000', «60000», «2015-06-01», «9999-01-01»);

Лабораторная работа 5

Создайте резервную копию с помощью следующей команды:

 # mysqldump -p employees employees | gzip >> /root/emp.sql.gz

Также вполне приемлемо сохранить необработанный файл **SQL** и затем запустить gzip для сжатия файла, убедитесь, что резервная копия действительна, изучите содержимое файла:

 # less /root/emp.sql.gz

Убедитесь, что вы создали резервную копию только содержимого таблицы сотрудников из базы данных сотрудников.

Чтобы восстановить резервную копию, сначала создайте новую базу данных:

CREATE DATABASE emp_restored;

Затем импортируйте содержимое резервной копии:

 # gunzip /root/emp.sql.gz
 # cat emp.sql | mysql -p emp_restored

В качестве окончательной проверки убедитесь, что данные выглядят одинаково, запустив **SQL-запрос**, который вы использовали в ответах в части 1 из лабораторной работы 3.