13

Администрирование MySQL

В предыдущих главах вы научились устанавливать MySQL в системах Linux и Windows, создавать реляционные базы данных и, используя язык SQL, выполнять операции добавления, изменения, удаления и извлечения данных из этих баз. В этой и нескольких последующих главах вы научитесь выполнять различные административные операции, которые позволят получать информацию о состоянии сервера и список пользователей, которым разрешен доступ к серверу MySQL, а также настроить репликацию баз данных и подготовиться к возможным аварийным ситуациям.

Эта глава знакомит с администрированием MySQL и описывает, как изменять стандартные настройки MySQL, устанавливать системные переменные и выполнять протоколирование работы сервера. В главе 14 объясняется, как обеспечить безопасность баз данных, а в главе 15 описано, как оптимизировать производительность сервера. Глава 16 посвящена вопросам резервного копирования и восстановления баз данных, а также настройке репликации. Таким образом, эта глава является отправной точкой для знакомства с основными административными функциями.

В этой главе рассмотрены перечисленные ниже вопросы.

- Использование утилиты `mysqladmin` для проверки системных настроек и выполнения операций, связанных с работой сервера.
- Применение операторов `SHOW VARIABLES`, `SELECT` и `SHOW STATUS` для получения значений системных переменных, а также использование параметров запуска сервера, файлов опций и оператора `SET` для установки системных переменных.
- Ведение журналов ошибок, запросов и обновлений.

Выполнение административных функций

После инсталляции MySQL может потребоваться просмотреть информацию о сервере, такую, например, как номер версии или данные о состоянии сервера. Возможно также, что вам понадобится выполнить административные операции вроде останова сервера или очистки кэша хоста. Для выполнения этих функций в MySQL предусмотрена утилита `mysqladmin`, которая предоставляет административный интерфейс к серверу MySQL. С использованием этого средства можно выполнять разнообразные административные операции, такие как получение информации о конфигурации сервера MySQL, установка паролей, запуск сервера, создание и удаление баз данных и перезагрузка таблицы привилегий.

Утилита `mysqladmin`, как и `mysql`, вызывается из командной строки. При запуске утилиты `mysqladmin` ей можно указать несколько необязательных программных опций и одну или более команд, как показано в следующем синтаксисе этой утилиты:

```
mysqladmin [<программная_опция> [<программная_опция>...]]
<command> [<командная_опция>] [{<команда> [<командная_опция>]}...]
```

Опции `<программная_опция>` определяют параметры подключения к серверу MySQL и различные режимы работы утилиты `mysqladmin`. Например, с помощью программных опций можно указать учетную запись пользователя MySQL, пароль для этой учетной записи и номер порта, используемый для подключения к серверу.

Большинство программных опций можно определять двумя способами. Например, опция `--force` может быть также указана как `-f`. В табл. 13.1 дается описание большинства программных опций утилиты `mysqladmin` с указанием, где это допустимо, как длинной, так и короткой формы опции. (Если представлены обе формы опции, они отделены друг от друга вертикальной чертой (`|`).) Обратите внимание, что списки опций утилиты `mysqladmin` для различных версий MySQL могут отличаться между собой, поэтому за текущим описанием опций, а также информацией о новых опциях утилиты `mysqladmin`, следует обращаться к документации по последней версии MySQL. (Дополнительную информацию об указании программных опций можно найти в главе 3.)

Таблица 13.1. Программные опции утилиты `mysqladmin`

Синтаксис опции	Описание
`--character-set-dir=<путь>`	Указывает каталог, где хранятся наборы символов.
`--compress | -C`	Использовать сжатие данных, пересылаемых между клиентом и сервером. Обе стороны, как сервер, так и клиент, должны обеспечивать сжатие.
`--connect_timeout=<количество_секунд>`	Указывает число секунд ожидания подключения к серверу.
`--count=<число> | -c <число>`	Сколько раз будет выполнена команда, указанная при вызове `mysqladmin` с опцией `--sleep`.

Окончание табл. 13.1

Синтаксис опции	Описание
`--force | -f`	Заставляет выполнить команду `drop <база_данных>` без запроса на подтверждение. Также заставляет `mysqladmin` продолжить выполнение команд, указанных при вызове утилиты, даже при возникновении ошибки в одной из них. (Обычно `mysqladmin` завершает свою работу, если одна из указанных команд приводит к ошибке.)
`--help | -?`	Показать информацию об опциях утилиты `mysqladmin`.
`--host=<имя_хоста> | -h <имя_хоста>`	Указывает имя хоста, на котором запущен сервер MySQL.
`--password[=<пароль>] | -p[<пароль>]`	Указывает пароль, используемый при подключении к серверу MySQL. При использовании краткой формы `-p` пароль должен следовать непосредственно за опцией (между именем опции и значением пароля не должно быть пробела). При использовании обеих форм опций, если пароль не указан, он запрашивается у пользователя.
`--port=<номер_порта> | -P <номер_порта>`	Указывает номер порта, используемый при подключении к серверу MySQL.
`--protocol={TCP | SOCKET | PIPE | MEMORY}`	Указывает протокол, который надлежит использовать при подключении к серверу MySQL.
`--silent | -s`	При невозможности подключиться к серверу завершить работу без вывода какого-либо сообщения.
`--sleep=<количество_секунд> | -i <количество_секунд>`	Указывает, что заданная в `mysqladmin` команда (вместе со всеми своими опциями) должна выполняться многократно. Заданное количество секунд задает интервал задержки между повторными выполнениями. Опцию `-sleep` можно использовать в сочетании с опцией `--count`.
`--user=<имя_пользователя> | -u <имя_пользователя>`	Указывает имя пользователя, применяемое при подключении к серверу MySQL.
`--version | -V`	Показать номер версии и другую информацию об утилите `mysqladmin`, а также информацию о сервере MySQL.
`--wait[=<число>] | -w[<число>]`	Повторять попытки подключения к серверу MySQL, если соединение с сервером не удалось установить. Необязательный параметр `<число>` задает количество попыток. Если количество попыток не указано, подразумевается 1. При использовании краткой формы `-w` между именем опции и ее значением не должно быть пробела.

При вызове утилиты mysqladmin кроме опций программы необходимо указать одну или несколько команд. Команды определяют действия, которые должна выполнить утилита при своем вызове. Например, можно проверить, запущен ли сервер MySQL, или перечитать таблицы привилегий. В некоторых случаях вместе с командой нужно задавать командную опцию. Случаи, когда требуется указание опции, обычно очевидны. Например, mysqladmin можно применять для создания базы данных. Для этого, очевидно, необходимо указать имя новой базы данных. Это имя задается опцией команды.

В табл. 13.2 перечислены команды, которые можно использовать в утилите mysqladmin. Команды утилиты mysqladmin, как и ее опции, могут изменяться от версии к версии, поэтому за текущим описанием команд, а также информацией о новых командах утилиты mysqladmin, следует обращаться к документации по последней версии MySQL.

Таблица 13.2. Команды утилиты mysqladmin

Синтаксис команды	Описание
create <база_данных>	Создать новую базу данных с указанным именем.
drop <база_данных>	Удалить базу данных с указанным именем.
extended-status	Показать имена и значения серверных переменных состояния.
flush-hosts	Очистить кэш хоста.
flush-logs	Закрыть, а затем повторно открыть файлы журналов. Для некоторых типов журналов MySQL создает новый файл.
flush-privileges	Перечитать таблицы привилегий. Эта команда эквивалентна команде reload.
flush-status	Очистить переменные состояния и обнулить некоторые счетчики.
flush-tables	Закрыть все открытые таблицы.
flush-threads	Очистить кэш потоков.
kill <ID_потока> [<ID_потока>...]	Завершить указанные потоки. Каждый поток связан с одним из соединений, поэтому завершение потока приводит к разрыву соответствующего ему соединения. Просмотреть список активных потоков можно с помощью команды processlist.
password <новый_пароль>	Установить новый пароль для текущего пользователя, вызвавшего утилиту mysqladmin. Если этот пользователь уже имеет пароль, то новый пароль вступит в действие после выполнения этой команды. Это означает, что при вызове mysqladmin в опции --password необходимо указывать старый пароль.
ping	Проверить, работает ли сервер MySQL.
processlist	Показать список активных потоков сервера MySQL. Каждому потоку назначается идентификатор (ID), который можно использовать в команде kill для завершения этого потока.

Окончание табл. 13.2

Синтаксис команды	Описание
`reload`	Перечитать таблицы привилегий. Эта команда эквивалентна команде `flush-privileges`.
`refresh`	Сбросить на диск все таблицы, закрыть и открыть заново все файлы журналов, и перечитать таблицы привилегий.
`shutdown`	Завершить работу сервера MySQL.
`slave-start`	Запустить процесс репликации на подчиненном сервере.
`status`	Показать отчет о текущем состоянии сервера MySQL.
`slave-stop`	Остановить процесс репликации на подчиненном сервере.
`variables`	Показать имена и значения переменных сервера.
`version`	Показать информацию о версии сервера MySQL.

Теперь, после того, как вы познакомились с опциями и командами утилиты `mysqladmin`, рассмотрим несколько примеров, демонстрирующих различные задачи, которые можно выполнять с помощью этой утилиты. В первом примере показано, как с помощью опции `--help` получить список всех опций и команд, которые можно использовать в утилите `mysqladmin`:

```
mysqladmin --help
```

Как видите, необходимо лишь указать имя утилиты `mysqladmin` и опцию `--help`. Даже если для подключения к серверу MySQL требуются имя пользователя и пароль, здесь их указывать необязательно, поскольку цель опции `--help` состоит в том, чтобы помочь найти всю информацию, необходимую при использовании этой утилиты для выполнения других операций. После выполнения этой команды вы получите информацию вроде следующей:

```
mysqladmin.exe  Ver 8.41 Distrib 5.0.37, for Win32 on ia32
Copyright (C) 2000-2006 MySQL AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license

Administration program for the mysqld daemon.
Usage: C:\Program Files\MySQL\MySQL Server 5.0\bin\mysqladmin.exe [OPTIONS]
command command....
  -c, --count=#               Number of iterations to make. This works with -i
                              (--sleep) only.
  -#, --debug[=name]          Output debug log. Often this is 'd:t:o,filename'.
  -f, --force                 Don't ask for confirmation on drop database; with
                              multiple commands, continue even if an error occurs.
  -C, --compress              Use compression in server/client protocol.
  --character-sets-dir=name   Directory where character sets are.
  --default-character-set=name Set the default character set.
  -?, --help                  Display this help and exit.
  -h, --host=name             Connect to host.
  -p, --password[=name]
```

Приведенные здесь результаты представляют только часть всей информации, которая будет выведена при указании опции `--help`. Как видите, здесь перечислены все

опции программы вместе с кратким описанием их назначения. Опция `--help` также выводит и описывает все допустимые команды утилиты `mysqladmin`.

В следующем примере показано совместное использование опций программы и команды:

```
mysqladmin -u root -p version
```

Сейчас в команду `mysqladmin` включена опция `-u` вместе со следующим за ней именем пользователя, которым является `root`. Затем идет опция `-p`. Опция `-p` отличается от других опций программы, которые требуют указания значения, тем, что она позволяет указать это значение отдельно от самой команды `mysqladmin`. При указании `-p` без пароля выдается подсказка на ввод пароля. Во время ввода пароля каждый введенный символ отображается в командной строке в виде звездочки. Чтобы сохранить пароль в секрете, MySQL рекомендует задавать его именно таким способом.

Кроме этих двух опций программы в упомянутый выше пример включена также команда `version`. Как видно из результата выполнения этого примера, показанного ниже, команда `version` выводит информацию о версии сервера MySQL:

```
mysqladmin.exe  Ver 8.41 Distrib 5.0.37, for Win32 on ia32
Copyright (C) 2000-2006 MySQL AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license

Server version        5.0.37-community-nt
Protocol version      10
Connection            localhost via TCP/IP
TCP port              3306
Uptime:               2 hours 15 min 1 sec

Threads: 1  Questions: 24  Slow queries: 0  Opens: 26  Flush tables: 1
Open tables: 0  Queries per second avg: 0.003
```

Как видите, кроме информации о версии сервера результаты включают в себя данные о версии протокола, соединении, номер порта, используемого для подключения к серверу, а также время безотказной работы сервера MySQL. Кроме того, команда `version` возвращает также информацию о состоянии сервера, описание которой приведено в табл. 13.3.

Таблица 13.3. Переменные состояния сервера MySQL

Переменная состояния	Описание
`Threads`	Количество открытых в текущий момент времени соединений.
`Questions`	Количество запросов, посланных серверу с момента его запуска.
`Slow queries`	Количество запросов, выполнение которых заняло времени больше, чем указано в системной переменной `long_query_time`.
`Opens`	Количество таблиц, открытых с момента запуска сервера.
`Flush tables`	Количество команд `flush`, `refresh` и `reload`, выполненных с момента запуска сервера.
`Open tables`	Количество таблиц, открытых в текущий момент времени. Это таблицы, к которым осуществляется доступ в настоящее время.
`Queries per second avg`	Среднее количество запросов в секунду.

Как было упомянуто ранее в этой главе, для указания большинства программных опций можно использовать длинную и краткую формы записи. В предыдущем примере для каждой опции применялась краткая форма. Того же результата можно достигнуть, используя длинную форму, как показано в следующем примере:

```
mysqladmin --user=root --password version
```

Как видите, в этой команде используется `--user=root` вместо `-u root` и `--password` вместо `-p`.

Кроме выполнения информационных команд, утилита `mysqladmin` может запускать команды, которые выполняют определенные действия. Так, следующий пример содержит команду `reload`:

```
mysqladmin -u root -p reload
```

Команда `reload` перечитывает таблицы привилегий, в результате чего все внесенные в эти таблицы изменения немедленно вступают в силу. Таблицы привилегий определяют, кто имеет доступ к серверу MySQL и каков тип этого доступа. (За более подробной информацией, касающейся таблиц привилегий и обеспечения безопасности MySQL, обращайтесь в главу 14.)

При запуске утилиты `mysqladmin` ей можно передать несколько команд. Например, следующая команда перечитывает таблицы привилегий и выводит информацию о версии `mysqladmin`:

```
mysqladmin -u root -p reload version
```

По этой команде MySQL перечитывает таблицы привилегий и возвращает ту же информацию, которую вы видели в предыдущем примере, когда была указана одна только команда `version`.

Теперь, после того как вы познакомились с примерами использования утилиты `mysqladmin` для выполнения административных функций, можно приступить к самостоятельной работе с этим средством. В следующем упражнении вы опробуете утилиту `mysqladmin`, просмотрев текущее состояние сервера MySQL и список потоков сервера. Потом с помощью этой утилиты вы сначала создадите базу данных, а затем удалите ее из системы.

Практическое занятие

Использование `mysqladmin` для администрирования MySQL

Следующие шаги описывают применение утилиты `mysqladmin` для решения некоторых административных функций.

1. Первая команда возвращает информацию о текущем состоянии сервера MySQL. Введите следующую команду в ответ на приглашение операционной системы:

   ```
   mysqladmin -u root -p status
   ```

   После появления подсказки введите пароль и нажмите клавишу <Enter>. Вы должны получить отчет о текущем состоянии сервера MySQL вроде следующего:

   ```
   Uptime: 8859 Threads: 1 Questions: 25 Slow queries: 0 Opens: 26 Flush tables: 1
   Open tables: 0 Queries per second avg: 0.003
   ```

2. Следующая команда возвращает информацию о состоянии и список потоков. Введите следующую команду в ответ на приглашение операционной системы:

   ```
   mysqladmin -u root -p status processlist
   ```

   После появления подсказки введите пароль и нажмите клавишу <Enter>. Вы должны получить отчет о текущем состоянии сервера MySQL, за которым следует список активных потоков сервера, что-то вроде следующего:

   ```
   Uptime: 8870 Threads: 1 Questions: 26 Slow queries: 0 Opens: 26 Flush tables: 1
   Open tables: 0 Queries per second avg: 0.003
   +--+------+----------------+--+----------+------+--------+----------------+
   |Id| User | Host           |db| Command  | Time | State  | Info           |
   +--+------+----------------+--+----------+------+--------+----------------+
   |18| root | localhost:2695 |  | Query    | 0    |        | show processlist|
   +--+------+----------------+--+----------+------+--------+----------------+
   ```

3. Следующая команда аналогична предыдущей за исключением того, что аргументы `status` и `processlist` указаны в обратном порядке. Введите следующую команду в ответ на приглашение операционной системы:

   ```
   mysqladmin -u root -p processlist status
   ```

   После подсказки наберите пароль и нажмите клавишу <Enter>. Вы должны получить список активных потоков сервера, за которым следует отчет о текущем состоянии:

   ```
   +--+------+----------------+--+----------+------+--------+----------------+
   |Id| User | Host           |db| Command  | Time | State  | Info           |
   +--+------+----------------+--+----------+------+--------+----------------+
   |19| root | localhost:2698 |  | Query    | 0    |        | show processlist|
   +--+------+----------------+--+----------+------+--------+----------------+
   Uptime: 8880 Threads: 1 Questions: 29 Slow queries: 0 Opens: 26 Flush tables: 1
   Open tables: 0 Queries per second avg: 0.003
   ```

4. Следующая команда создает базу данных с именем `db1`. Введите следующую команду в ответ на приглашение операционной системы:

   ```
   mysqladmin -u root -p create db1
   ```

   После появления подсказки введите пароль и нажмите клавишу <Enter>. Вы должны вернуться обратно в командную строку операционной системы.

5. Чтобы убедиться в том, что база данных создана, откройте утилиту `mysql`. Для этого введите следующую команду в ответ на приглашение операционной системы:

   ```
   mysql
   ```

   Запустится утилита `mysql`.

6. Затем просмотрите список существующих в системе баз данных. Введите следующую команду в ответ на приглашение утилиты `mysql`:

   ```
   SHOW DATABASES;
   ```

   Полученный список должен содержать, по крайней мере, три базы данных, как показано в следующем результирующем наборе:

```
+---------+
| Database |
+---------+
| db1     |
| mysql   |
| test    |
+---------+
3 rows in set (0.00 sec)
```

7. Теперь закройте утилиту `mysql`, выполнив следующую команду:

   ```
   exit
   ```

   Вы должны вернуться в командную строку операционной системы.

8. Последняя команда этого упражнения удаляет базу данных `db1` из системы. Введите следующую команду в ответ на приглашение операционной системы:

   ```
   mysqladmin -u root -p -f drop db1
   ```

   После появления подсказки введите пароль и нажмите клавишу <Enter>. Вы должны получить сообщение о том, что база данных `db1` удалена.

Описание полученных результатов

В этом упражнении вы создали ряд команд для выполнения различных административных функций. Эти команды основаны на утилите `mysqladmin`. Первая выполненная административная функция состояла в том, чтобы с помощью следующей команды получить данные о текущем состоянии сервера MySQL:

```
mysqladmin -u root -p status
```

Эта команда содержит несколько опций. Первая опция (`-u`) указывает на то, что для подключения к серверу MySQL должна использоваться учетная запись пользователя `root`. Вторая опция (`-p`) указывает на необходимость использования пароля. При таком задании опции `-p`, без указания самого пароля, после запуска этой команды появится подсказка на ввод пароля. Следует отметить, что вы можете использовать `--user=root` вместо `-u root` и `--password` вместо `-p`.

Третьей опцией, включенной в предыдущий пример, является команда `status`. Эта команда выводит данные о текущем состоянии сервера MySQL. Данные включают в себя информацию о времени непрерывной работы, активных потоках, количестве обработанных запросов и другую информацию о состоянии сервера. Выводимая по этой команде информация в основном подобна той, которую возвращает команда `version`. Главное отличие между ними состоит в том, что команда `version` показывает время непрерывной работы сервера (uptime) в часах, минутах и секундах, а команда `status` — в секундах.

Следующая созданная вами команда `mysqladmin` аналогична предыдущей, за исключением того, что в нее добавлен еще параметр `processlist`, как показано ниже:

```
mysqladmin -u root -p status processlist
```

Параметр `processlist` выводит список активных потоков сервера. Поскольку эта команда содержит параметры `status` и `processlist`, в ее результатах присутствует информация, возвращаемая обоими этими параметрами, в том порядке, в котором эти параметры указаны в команде. Если эти параметры поменять местами, как показано в следующей команде, то и результаты будут представлены в обратном порядке:

```
mysqladmin -u root -p processlist status
```

Теперь вывод команды будет содержать сначала список потоков, а затем информацию о состоянии. Следующая созданная вами команда добавляет в систему новую базу данных по имени db1:

```
mysqladmin -u root -p create db1
```

Эта команда содержит параметр create, за которым следует имя новой базы данных (db1). После выполнения этой команды вы с помощью утилиты mysql просмотрели список существующих баз данных, убедившись в том, что база db1 действительно была добавлена. Затем вы вышли из утилиты mysql и с помощью следующей команды удалили эту базу данных:

```
mysqladmin -u root -p -f drop db1
```

Эта команда содержит параметр drop и имя базы данных. В результате выполнения этой команды база данных db1 удаляется из системы. Обратите внимание, что эта команда также содержит опцию -f. Указание этой опции, которую еще можно задать в виде --force, приводит к тому, что утилита mysqladmin выполнит команду drop без запроса на подтверждение удаления. В результате база данных db1 будет удалена из системы сразу же по выполнении этой команды, причем система не запросит подтверждение удаления базы данных.

Управление системными переменными

При запуске MySQL считывает стандартные и пользовательские настройки, которые определяют работу сервера. Эти настройки в значительной степени определяют окружение сервера MySQL и способ подключения пользователей к базе данных. Настройки сохраняются в системных переменных и содержат такие детали работы сервера, как время ожидания подключения, местоположение корневого каталога базы данных, включен ли режим протоколирования работы сервера, максимально возможное количество соединений, размеры кэшей и многие другие параметры.

Для всех системных переменных установлены значения по умолчанию. Вместо значений по умолчанию можно задавать другие значения, указав их в командной строке при запуске сервера или в файлах опций. Значения многих переменных можно изменять во время работы сервера с помощью оператора SET. (Более подробно об этих трех способах задания системных переменных вы узнаете позднее в этой главе.)

Системные переменные MySQL могут быть двух видов — глобальные переменные и переменные сеанса. Глобальные переменные влияют на работу всего сервера. Переменные сеанса оказывают воздействие на работу конкретных пользовательских соединений. При запуске сервер инициализирует глобальные системные переменные значениями по умолчанию либо значениями, указанными в командной строке. После установки соединения MySQL инициализирует набор переменных сеанса для этого соединения.

Для большинства переменных сеанса имеются соответствующие глобальные переменные. В качестве начальных значений этих переменных сеанса берутся значения соответствующих глобальных переменных, которые они имели на момент установки соединения. Например, если на момент установки соединения глобальная переменная wait_timeout имела значение 600, то значение переменной сеанса wait_timeout для этого соединения тоже будет равно 600. (Переменная wait_timeout указывает количество секунд ожидания активности в соединении, прежде чем закрыть это соеди-

нение.) Для некоторых переменных сеанса не предусмотрено соответствующей глобальной переменной. В этом случае этим переменным MySQL присваивает значение по умолчанию.

Не все системные переменные одинаковы с точки зрения выполняемых ими функций или возможности их изменения. В MySQL предусмотрены следующие три типа системных переменных.

- **Серверные системные переменные.** Набор системных переменных, которые определяют работу самого сервера MySQL и способы взаимодействия клиентов с сервером. По сути, к серверным системным переменным относятся все системные переменные, которые не используются явно для возврата данных о состоянии сервера.
- **Динамические системные переменные.** Подмножество множества серверных системных переменных, значения которых можно изменять в процессе работы сервера.
- **Серверные переменные состояния.** Набор системных переменных, основное назначение которых состоит в предоставлении информации о текущем состоянии сервера MySQL. Все переменные состояния являются глобальными переменными.

Просмотр этих трех типов системных переменных в MySQL производится по-разному. В следующем разделе вы научитесь просматривать значения переменных каждого типа.

Извлечение значений системных переменных

Способ, который используется для просмотра значений системных переменных, зависит от их типа. Для просмотра серверных системных переменных используется оператор `SHOW VARIABLES`. Для просмотра динамических системных переменных служит оператор `SELECT`. И, наконец, для просмотра серверных переменных состояния применяется оператор `SHOW STATUS`.

В MySQL имеются сотни системных переменных, описание которых заняло бы намного больше места, чем позволяют размеры этой книги. Поэтому если вам нужна информация о какой-либо конкретной системной переменной или необходимо получить список всех поддерживаемых переменных, необходимо использовать один из трех операторов, описываемых в следующих разделах, или обратиться к документации, поставляемой вместе с сервером MySQL.

Использование оператора `SHOW VARIABLES` для извлечения значений серверных системных переменных

Оператор `SHOW VARIABLES` внутри утилиты `mysql` позволяет просмотреть большинство серверных системных переменных, а также их текущие значения. (Этот оператор не отображает некоторые из динамических системных переменных. Для просмотра значений этих переменных следует применять оператор `SELECT`, который описывается далее в этой главе.) С помощью оператора `SHOW VARIABLES` можно просматривать глобальные переменные или переменные сеанса, а также уточнять запрос, добавляя в него конструкцию `LIKE`, как показано в следующем описании синтаксиса этого оператора:

```
SHOW [GLOBAL | SESSION] VARIABLES [LIKE '<значение>']
```

Как видите, единственной обязательной частью этого оператора являются ключевые слова SHOW VARIABLES. Кроме них можно также задавать одно из ключевых слов GLOBAL или SESSION. Если указать ключевое слово GLOBAL, MySQL выведет глобальные системные переменные. Если указать SESSION, MySQL отобразит системные переменные сеанса. Если же не указывать ни одну из этих альтернатив, то при выводе будут показаны переменные сеанса, если они существуют, и глобальные переменные в противном случае.

Синтаксис оператора SHOW VARIABLES содержит также необязательную конструкцию LIKE. Конструкция LIKE позволяет вывести информацию о конкретной системной переменной или показать системные переменные, имена которых удовлетворяют заданному шаблону.

Позже в этом разделе будет показан пример использования конструкции LIKE, но сначала мы рассмотрим простой пример использования оператора SHOW VARIABLES. Следующий оператор возвращает все глобальные переменные и их значения:

```
SHOW GLOBAL VARIABLES;
```

Как видите, этот оператор содержит обязательные ключевые слова SHOW VARIABLES и необязательное ключевое слово GLOBAL. После выполнения этого оператора вы должны получить примерно следующие результаты:

```
+------------------------------+-----------------------------------------+
| Variable_name                | Value                                   |
+------------------------------+-----------------------------------------+
| back_log                     | 50                                      |
| basedir                      | C:\Program Files\MySQL\MySQL Server 5.0\  |
| binlog_cache_size            | 32768                                   |
| bulk_insert_buffer_size      | 8388608                                 |
| character_set_client         | latin1                                  |
| character_set_connection     | latin1                                  |
| character_set_database       | latin1                                  |
| character_set_results        | latin1                                  |
| character_set_server         | latin1                                  |
| character_set_system         | utf8                                    |
```

Приведенные здесь результаты содержат неполный список всех серверных системных переменных. Обычно этот оператор возвращает примерно 180 системных переменных. Просмотреть системные переменные сеанса можно с помощью следующего оператора:

```
SHOW SESSION VARIABLES;
```

Результаты, возвращаемые эти оператором, обычно напоминают результаты, которые возвращает оператор с ключевым словом GLOBAL. В большинстве случаев бывает достаточно показать только часть списка, возвращаемого этими операторами. Этой цели служит конструкция LIKE. LIKE позволяет указать значение, которое сравнивается с именами переменных. Например, предположим, что нужно просмотреть все переменные, связанные с запросами к базе данных. Для этого достаточно включить в оператор конструкцию LIKE, содержащую строку *запрос*, как показано в следующем операторе:

```
SHOW VARIABLES LIKE '%запрос%';
```

Как видите, в LIKE указана строка, окруженная шаблонным символом %. Вследствие этого имя переменной может начинаться или заканчиваться любыми символами, но должно содержать строку *запрос*. После выполнения этого оператора вы должны получить примерно такие результаты:

```
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| ft_query_expansion_limit     | 20      |
| have_query_cache             | YES     |
| long_query_time              | 10      |
| query_alloc_block_size       | 8192    |
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
| query_cache_size             | 0       |
| query_cache_type             | ON      |
| query_cache_wlock_invalidate | OFF     |
| query_prealloc_size          | 8192    |
+------------------------------+---------+
10 rows in set (0.00 sec)
```

Эта таблица содержит только те переменные, в имени которых присутствует строка *запрос*. Информацию о назначении каждой переменной можно найти в документации, поставляемой с сервером MySQL. Теперь посмотрим, как с помощью оператора SELECT можно получить значения динамических системных переменных.

Использование оператора SELECT для извлечения значений динамических системных переменных

Как и все системные переменные, динамические переменные подразделяются на глобальные и специфичные для конкретного сеанса. Многие динамические переменные действуют как глобально, так и на уровне сеанса, в то время как другие являются либо глобальными, либо переменными сеанса, но не теми и другими одновременно. Например, переменная autocommit (которая задает режим автоматического завершения транзакций) является переменной сеанса, а переменная binlog_cache_size (которая указывает размер кэша для бинарного журнала) является глобальной и не имеет отношения к отдельному сеансу. В то же время переменная wait_timeout является допустимой как на глобальном уровне, так и на уровне сеанса, причем значения для каждой из них могут отличаться между собой.

Текущее значение динамической системной переменной, независимо от того используется она глобально или в рамках сеанса, можно узнать с помощью оператора SELECT. Для просмотра значения конкретной переменной этот метод является более простым, чем использование оператора SHOW VARIABLES. Получить же значения динамических переменных сеанса, не отображаемых командой SHOW VARIABLES, можно только с помощью этого метода.

Ниже приведен синтаксис оператора SELECT для получения значения динамической переменной:

```
SELECT @@[global.]<переменная> [{, @@[global.]<переменная>}...]
```

Как видите, необходимо указать ключевое слово SELECT, удвоенный символ @ и имя переменной. Чтобы получить значение глобальной переменной, имя этой пере-

менной потребуется предварить ключевым словом global со следующей за ним точкой, как показано в следующем примере:

```
SELECT @@global.max_binlog_size;
```

Этот оператор SELECT извлекает значение глобальной переменной max_binlog_size, которая определяет максимальный размер файла бинарного журнала, выраженный в байтах. После выполнения этого оператора вы должны получить примерно следующие результаты:

```
+-------------------------+
| @@global.max_binlog_size|
+-------------------------+
|              1073741824 |
+-------------------------+
1 row in set (0.00 sec)
```

Поскольку эта переменная принадлежит к числу системных переменных, возвращаемых оператором SHOW VARIABLES, те же самые результаты можно получить и с применением этого оператора, как показано в следующем примере:

```
SHOW GLOBAL VARIABLES LIKE 'max_binlog_size';
```

Как видите, конструкция LIKE содержит конкретное значение без символов шаблона. Результаты выполнения этого оператора имеют примерно следующий вид:

```
+--------------------+-------------+
| Variable_name      | Value       |
+--------------------+-------------+
| max_binlog_size    | 1073741824  |
+--------------------+-------------+
1 row in set (0.00 sec)
```

Несмотря на то что результаты представлены в иной форме, нежели при использовании оператора SELECT, по сути, они содержат ту же самую информацию. Запомните, однако, что этот метод работает только для тех системных переменных, которые возвращает оператор SHOW VARAIBLES.

Чтобы получить значение переменной сеанса, а не глобальной переменной, необходимо просто опустить ключевое слово global и точку, как показано в следующем примере:

```
SELECT @@tx_isolation;
```

В этом примере оператор SELECT извлекает значение переменной сеанса tx_isolation, которая определяет уровень изоляции транзакций текущего соединения:

```
+--------------------+
| @@tx_isolation     |
+--------------------+
| REPEATABLE-READ    |
+--------------------+
1 row in set (0.01 sec)
```

Уровень изоляции транзакций в данном случае равен REPEATABLE READ. (За более подробной информацией об уровнях изоляции транзакций обращайтесь в главу 12.)

Как следует из описания синтаксиса оператора SELECT, в этом операторе можно указывать несколько динамических системных переменных. Например, следующий оператор SELECT возвращает уровень изоляции для текущего сеанса, а также максимально допустимое значение размера файла бинарного журнала:

```
SELECT @@tx_isolation, @@global.max_binlog_size;
```

Как видно из результатов выполнения этого оператора, приведенных ниже, на экран будут выведены значения обеих переменных:

```
+-------------------+----------------------------+
| @@tx_isolation    | @@global.max_binlog_size   |
+-------------------+----------------------------+
| REPEATABLE-READ   |               1073741824   |
+-------------------+----------------------------+
1 row in set (0.00 sec)
```

Операторы SELECT и SHOW VARIABLES представляют собой удобное средство для просмотра системных переменных; однако ни один из них не позволяет показать значения серверных переменных состояния. Для этой цели предназначен оператор SHOW STATUS.

Использование оператора SHOW STATUS для извлечения значений серверных переменных состояния

Большинство серверных переменных состояния представляют собой счетчики, значение которых устанавливается равным нулю при запуске сервера MySQL. Начиная с этого момента, они подсчитывают количество возникновений конкретного события. Например, переменная состояния Connections содержит общее количество соединений с момента запуска сервера, а в переменной Bytes_sent хранится суммарное количество байтов, переданных всем клиентам. Кроме переменных состояния, которые подсчитывают, сколько раз произошло то или иное событие, существуют также переменные состояния, которые предоставляют информацию другого рода, такую как количество памяти, доступной для кэша запросов, или количество открытых таблиц. Для просмотра текущих значений переменных состояния служит оператор SHOW STATUS, синтаксис которого имеет следующий вид:

```
SHOW STATUS [LIKE '<значение>']
```

Как показывает синтаксис, вы должны указать ключевые слова SHOW STATUS. Кроме них вы можете включить конструкцию LIKE. LIKE работает точно так же, как и аналогичная конструкция оператора SHOW VARIABLES, как вы увидите позднее в этом разделе.

Возвращаясь к синтаксису, можно обнаружить, что стандартный оператор SHOW STATUS очень прост, как показывает следующий пример:

```
SHOW STATUS;
```

Этот оператор возвращает все переменные состояния и их текущие значения. Следующий результирующий набор представляет собой лишь часть всего вывода оператора SHOW STATUS:

```
+-------------------------+---------+
| Variable_name           | Value   |
+-------------------------+---------+
| Aborted_clients         | 0       |
| Aborted_connects        | 0       |
| Binlog_cache_disk_use   | 0       |
| Binlog_cache_use        | 0       |
| Bytes_received          | 993     |
| Bytes_sent              | 879     |
| Com_admin_commands      | 0       |
| Com_alter_db            | 0       |
| Com_alter_table         | 0       |
| Com_analyze             | 0       |
| Com_backup_table        | 0       |
| Com_begin               | 0       |
| Com_change_db           | 0       |
| Com_change_master       | 0       |
| Com_check               | 0       |
| Com_checksum            | 0       |
| Com_commit              | 0       |
| Com_create_db           | 0       |
| Com_create_function     | 0       |
| Com_create_index        | 0       |
| Com_create_table        | 4       |
| Com_dealloc_sql         | 0       |
```

Всего в MySQL насчитывается более 150 переменных состояния. По этой причине конструкция `LIKE` является удобным средством для сокращения количества строк, выводимых этим оператором. Например, следующий оператор `SHOW STATUS` выводит на экран все переменные состояния, в имени которых содержится строка `select`:

```
SHOW STATUS LIKE '%select%';
```

После выполнения этого оператора вы должны получить примерно следующие результаты:

```
+-------------------------+---------+
| Variable_name           | Value   |
+-------------------------+---------+
| Com_insert_select       | 0       |
| Com_replace_select      | 0       |
| Com_select              | 44      |
| Select_full_join        | 0       |
| Select_full_range_join  | 0       |
| Select_range            | 0       |
| Select_range_check      | 0       |
| Select_scan             | 42      |
+-------------------------+---------+
8 rows in set (0.00 sec)
```

Как видите, этот оператор возвращает только те переменные состояния, которые в своем имени содержат строку `select`.

Сейчас, после того, как вы получили представление о различных операторах SQL, используемых для просмотра системных переменных и их значений, можно приступить к практическому применению этих операторов. В следующем упражнении вы с

помощью операторов SHOW VARIABLES, SELECT и SHOW STATUS попробуете получить значения заданных системных переменных.

Практическое занятие

Просмотр системных настроек

В приведенных ниже шагах описано использование различных операторов SQL для просмотра значений системных переменных.

1. Откройте утилиту mysql.

2. Создайте оператор SHOW VARIABLES, который возвращает глобальные переменные, в имени которых присутствует строка log. Выполните следующий SQL-оператор из командной строки mysql:

```
SHOW GLOBAL VARIABLES LIKE '%log%';
```

Вы должны получить примерно такие результаты:

```
+---------------------------------+------------+
| Variable_name                   | Value      |
+---------------------------------+------------+
| back_log                        | 50         |
| binlog_cache_size               | 32768      |
| expire_logs_days                | 0          |
| innodb_locks_unsafe_for_binlog  | OFF        |
| innodb_flush_log_at_trx_commit  | 1          |
| innodb_log_arch_dir             |            |
| innodb_log_archive              | OFF        |
| innodb_log_buffer_size          | 1048576    |
| innodb_log_file_size            | 10485760   |
| innodb_log_files_in_group       | 2          |
| innodb_log_group_home_dir       | .\         |
| innodb_mirrored_log_groups      | 1          |
| log                             | ON         |
| log_bin                         | ON         |
| log_error                       | .\ws01.err |
| log_slave_updates               | OFF        |
| log_slow_queries                | OFF        |
| log_update                      | OFF        |
| log_warnings                    | 1          |
| max_binlog_cache_size           | 4294967295 |
| max_binlog_size                 | 1073741824 |
| max_relay_log_size              | 0          |
| relay_log_purge                 | ON         |
| sync_binlog                     | 0          |
+---------------------------------+------------+
24 rows in set (0.00 sec)
```

3. Создайте оператор SELECT, который возвращает значение глобальной системной переменной max_connections. Переменная max_connections задает максимально возможное количество одновременно подключенных клиентов. Выполните следующий SQL-оператор из командной строки mysql:

```
SELECT @@global.max_connections;
```

Вы должны получить примерно следующие результаты:

```
+--------------------------+
| @@global.max_connections  |
+--------------------------+
| 100                      |
+--------------------------+
1 row in set (0.00 sec)
```

4. Создайте оператор SHOW STATUS, возвращающий переменные состояния, в имени которых присутствует строка thread. Выполните следующий SQL-оператор из командной строки mysql:

```
SHOW STATUS LIKE '%thread%';
```

Вы должны получить приблизительно такие результаты:

```
+------------------------+----------+
| Variable_name          | Value    |
+------------------------+----------+
| Delayed_insert_threads | 0        |
| Slow_launch_threads    | 0        |
| Threads_cached         | 0        |
| Threads_connected      | 1        |
| Threads_created        | 1        |
| Threads_running        | 1        |
+------------------------+----------+
6 rows in set (0.00 sec)
```

Описание полученных результатов

Первым созданным вами в этом упражнении SQL-оператором был SHOW VARIABLES:

```
SHOW GLOBAL VARIABLES LIKE '%log%';
```

Этот оператор содержит обязательные ключевые слова SHOW VARIABLES вместе с опцией GLOBAL. Вследствие этого он возвращает только глобальные переменные и их значения. Кроме того, в этом операторе используется конструкция LIKE для указания того, что возвращаемые переменные должны содержать в своем имени строку log. (Шаблонные символы % указывают на то, что этой строке может предшествовать или следовать за ней любое количество символов, включая ноль.)

Очередным созданным в этом упражнении SQL-оператором был показанный ниже SELECT:

```
SELECT @@global.max_connections;
```

Этот оператор SELECT извлекает значение переменной max_connections. Поскольку имя переменной указано с префиксом global, этот оператор ссылается на глобальную переменную, а не на переменную сеанса. Чтобы просмотреть значение переменной сеанса, префикс global необходимо опустить.

Далее следовал такой SQL-оператор SHOW STATUS:

```
SHOW STATUS LIKE '%thread%';
```

Кроме указания обязательных ключевых слов SHOW STATUS в этот оператор включена конструкция LIKE, которая ограничивает набор возвращаемых переменных состояния только теми переменными, в имени которых содержится строка thread.

Конфигурирование сервера

В главе 3 вы научились указывать параметры при запуске программ, входящих в состав MySQL, включая и сам сервер MySQL. Из всех рассмотренных методов двумя наиболее распространенными являются определение параметров в командной строке и в файле опций. Большинство параметров, указываемых в командной строке или файле опций, являются системными переменными, влияющими на работу сервера MySQL. При функционирующем сервере можно изменять настройки динамических системных переменных с помощью оператора `SET`. В этом разделе дается обзор методов задания параметров во время запуска сервера, после чего рассматривается использование оператора `SET` для установки системных переменных в процессе работы сервера.

В этом разделе вопросы задания параметров в командной строке или в файле опций затронуты лишь мельком. Каждая из этих тем подробно рассматривалась в главе 3. Эта информация включена сюда только для того, чтобы дать связанный обзор настройки системных переменных.

Задание системных настроек в командной строке

Как говорилось в главе 3, при запуске сервера MySQL ему можно передать параметры. Если параметр требует указания значения, то перед параметром ставится двойной дефис (--), а после него — знак равенства и присваиваемое значение. Например, в следующей команде значение присваивается системной переменной `query_cache_limit`:

```
mysqld --query_cache_limit=1000000
```

Переменная `query_cache_limit` ограничивает размер (в байтах) кэшируемого результирующего набора. Результаты, превышающие значение этой переменной, не кэшируются. Значение переменной `query_cache_limit`, заданное в командной строке, перекрывает значение этой переменной по умолчанию, равное 1 048 576 байт. Вы можете также задать несколько параметров в командной строке, как показано в следующем примере:

```
mysqld --query_cache_limit=1000000 --wait_timeout=600
```

В этом примере при запуске сервера максимальный размер кэшируемого результата запроса устанавливается равным 1 000 000 байт, а время ожидания соединения — 600 секундам. Несмотря на возможность указания значений этим и другим переменным в командной строке, все же предпочтительным способом определения начальных значений является использование файла опций.

Задание системных настроек в файле опций

Преимущество использования файла опций по сравнению с применением параметров командной строки заключается в том, что в файле опций значения переменным задаются только один раз. Параметры же, передаваемые в командной строке, необходимо указывать каждый раз при запуске сервера. Задание параметров в командной строке оправдывает себя только в том случае, если нужно запустить сервер с этими параметрами только один раз. Во всех остальных случаях лучше иметь дело с файлом опций.

При определении значений системных переменных в файле опций их необходимо поместить в раздел [mysqld] этого файла. Например, в следующих строках устанавливаются значения для двух системных переменных:

```
[mysqld]
query_cache_limit=1000000
wait_timeout=600
```

При определении системных переменных в файле опций перед именем переменной не нужно ставить два знака дефиса, как это делается при указании переменных в командной строке. Однако знак равенства между именем переменной и соответствующим ей значением является необходимым.

При использовании файла опций для определения настроек сервера в системе Windows необходимо помнить о том, где служба MySQL ищет файл опций. По умолчанию эта служба ищет настройки сервера в файле `my.ini` *в каталоге* `C:\Program Files\MySQL\MySQL Server <версия>`*. Клиентские утилиты MySQL ищут свои настройки в файле* `my.ini` *каталога* `C:\WINDOWS`*. Один из вариантов объединения этих файлов состоит в следующем: нужно удалить текущую службу MySQL, а затем создать ее заново так, чтобы в качестве файла конфигурации она использовала тот же файл, что и клиентские утилиты. В результате вам понадобиться сопровождать только один файл опций. (Этот прием описан в упражнении, приведенном в конце этой главы.) В противном случае придется поддерживать два файла опций.*

Задание системных настроек во время выполнения

Иногда возникает необходимость изменить настройки системных переменных во время работы сервера — то ли на глобальном уровне, то ли на уровне сеанса. Если сделать изменения на глобальном уровне, то внесенные изменения окажут влияние на всех клиентов. Однако они повлияют только на новые соединения и не затронут соответствующих переменных в текущих соединениях. Если же изменения сделать на уровне сеанса, то они окажут влияние только на соединение, в котором эти изменения были произведены. Другие соединения не будут затронуты.

Установить значение для динамической системной переменной в процессе работы сервера можно с помощью оператора SET, синтаксис которого имеет следующий вид:

```
SET [GLOBAL | SESSION] <насптройка_переменной>
```

Этот оператор состоит из ключевого слова SET и настройки переменной, включающей в себя имя переменной, за которой следуют знак равенства и новое значение. Если задать ключевое слово GLOBAL, то MySQL присвоит новое значение указанной глобальной переменной. Если же вместо него указать SESSION или вообще ничего не указывать, то новое значение присвоится соответствующей переменной сеанса.

Полный список динамических системных переменных, а также описание того, действуют они на глобальном уровне, на уровне сеанса или же и там и там, приведены в документации по MySQL.

Например, установить переменной query_cache_limit значение 1000000 так, чтобы оно действовало глобально, можно с помощью следующего оператора SET:

```
SET GLOBAL QUERY_CACHE_LIMIT=1000000;
```

Как видите, этот оператор содержит ключевые слова SET GLOBAL, имя переменной, знак равенства и новое значение. После выполнения этого оператора новое значение будет действовать на глобальном уровне и затронет все соединения. В итоге теперь максимальный размер результата запроса, который может быть помещен в кэш, меньше значения по умолчанию.

Чтобы установить значение переменной текущего сеанса, нужно либо указать ключевое слово SESSION, либо вообще ничего не указывать, как показано в следующем примере:

```
SET WAIT_TIMEOUT=600;
```

Этот оператор устанавливает для переменной wait_timeout значение 600 секунд, которое намного меньше его умалчиваемого значения, равного 28800 секундам.

Теперь, после того как вы получили представление о том, как использовать оператор SET для установки значений динамических системных переменных, можно попробовать применить его на практике. В следующем упражнении оператор SET используется для изменения значения системной переменной foreign_key_checks. Эта переменная определяет, нужно ли при изменении данных выполнять проверку ограничений внешних ключей для таблиц InnoDB. Если значение этой переменной равно 1, проверка выполняется. Если оно равно 0, то нет. По умолчанию эта переменная равна 1.

Практическое занятие

Изменение системных настроек в процессе работы

1. Откройте утилиту mysql.
2. Сначала с помощью оператора SELECT просмотрите текущее значение системной переменной foreign_key_checks. Введите следующий SQL-оператор в командной строке утилиты mysql:

   ```
   SELECT @@foreign_key_checks;
   ```

   Вы должны получить примерно такой результат:

   ```
   +----------------------+
   | @@foreign_key_checks |
   +----------------------+
   | 1                    |
   +----------------------+
   1 row in set (0.00 sec)
   ```

3. Чтобы отключить проверку внешнего ключа, установите значение переменной foreign_key_checks в 0. Введите следующий SQL-оператор в командной строке утилиты mysql:

   ```
   SET FOREIGN_KEY_CHECKS=0;
   ```

   Вы должны получить сообщение об успешном выполнении этого оператора.

4. Снова воспользуйтесь оператором SELECT для просмотра текущего значения переменной foreign_key_checks. Введите тот же самый SQL-оператор, который вы вводили на шаге 2. Вы должны получить примерно следующие результаты:

```
+----------------------+
| @@foreign_key_checks |
+----------------------+
| 0                    |
+----------------------+
1 row in set (0.00 sec)
```

5. Завершите текущий сеанс MySQL, закрыв утилиту `mysql`, а затем начните новый сеанс, перезапустив утилиту.

6. Наконец, проверьте текущее значение `foreign_key_checks` в последний раз. Введите следующий SQL-оператор в командной строке утилиты `mysql`:

   ```
   SELECT @@foreign_key_checks;
   ```

   Вы должны получить примерно такой результат:

   ```
   +----------------------+
   | @@foreign_key_checks |
   +----------------------+
   | 1                    |
   +----------------------+
   1 row in set (0.00 sec)
   ```

Описание полученных результатов

В нескольких местах этого упражнения для проверки текущего значения системной переменной `foreign_key_checks` применялся следующий оператор `SELECT`:

```
SELECT @@foreign_key_checks;
```

С помощью аналогичной команды можно получить значение любой динамической системной переменной. Как видно из примера, эта команда состоит из ключевого слова `SELECT`, за которым следует имя переменной. Имени переменной должен предшествовать удвоенный символ `@`. Поскольку имя переменной указано без префикса `global`, будет возвращено значение переменной текущего сеанса.

Когда вы просматривали переменную `foreign_key_checks` в первый раз, ее значение было равно 1. Затем с помощью оператора `SET` вы изменили ее значение на 0:

```
SET FOREIGN_KEY_CHECKS=0;
```

После этого вы опять проверили значение `foreign_key_checks` и убедились, что теперь оно равно 0. Затем вы завершили текущий сеанс и сразу же открыли новый. При просмотре переменной `foreign_key_checks` в этот раз оказалось, что теперь ее значение равно 1. Задавая значение для переменной `foreign_key_checks`, вы не включили ключевое слово `GLOBAL`, поэтому была изменена переменная текущего сеанса. В результате после завершения сеанса это изменение исчезло и, когда стартовал новый сеанс, для переменной `foreign_key_checks` было использовано глобальное значение.

Управление файлами журналов

При установке MySQL, независимо от того, в какой из систем Linux или Windows она производится, автоматически настраивается ведение журнальных файлов ошибок. Кроме этого при установке системы можно указать другие типы регистрации событий, такие как регистрация запросов и бинарная регистрация. Этот раздел по-

священ работе с журналами ошибок, журналами запросов и бинарными журналами. За информацией о других имеющихся в MySQL типах журналов обращайтесь к документации по MySQL.

Отметим, что по умолчанию все файлы журналов располагаются в каталоге данных MySQL.

Работа с файлами журнала ошибок

Журнал ошибок — это текстовый файл, в котором регистрируется информация о времени запуска и останова сервера MySQL, а также информация об ошибках в процессе работы сервера. Журнал ошибок можно просмотреть в любом текстовом редакторе, таком как Notepad или Vim. По умолчанию журнал ошибок сохраняется в каталоге данных. В Linux этот файл называется `<хост>.err`. Другой каталог и имя файла можно указать, добавив в раздел `[mysqld]` файла опций следующую команду:

```
log-error=<путь/имя_файла>
```

В Windows файл журнала ошибок называется `mysql.err`, причем изменить его местоположение или имя нельзя. Кроме него MySQL создает файл с именем `<хост>.err`. Этот файл содержит лишь часть данных, регистрируемых в файле `mysql.err`, и в настоящее время не особенно полезен. Согласно документации MySQL файл `<хост>.err` может скоро заменить файл `mysql.err`. Кроме того, можно будет изменять имя этого файла и задавать путь к нему, но пока же все это не доступно в Windows.

Файл журнала ошибок содержит примерно такие записи:

```
c:\program files\mysql\mysql server 5.0\bin\mysqld-nt: ready for connections.
Version: '5.0.37-community-nt' socket: '' port: 3306 Source distribution
061002 10:33:29 [ERROR] DROP USER: Can't drop user: 'user1'@'%'; No such user
061006 9:04:10 [NOTE] c:\program files\mysql\mysql server 5.0\bin\mysqld-nt:
Normal shutdown

061006 9:04:10 InnoDB: Starting shutdown...
061006 9:04:13 InnoDB: Shutdown completed; log sequence number 0 84629
061006 9:04:13 [NOTE] c:\program files\mysql\mysql server 5.0\bin\mysqld-nt:
Shutdown complete
```

Первые две из приведенных выше строк записываются в файл при запуске сервера. Содержащаяся в них информация относится к самому серверу и включает такие детали, как путь и имя исполняемого файла, номер версии и номер порта TCP/IP. Третья строка содержит информацию об ошибке, которая произошла 2 октября 2006 года (`061002`). В данном случае эта ошибка возникла при попытке удалить несуществующего пользователя. Всякий раз, когда возникает ошибка, в журнал добавляется соответствующая строка. Оставшиеся строки связаны с завершением работы сервера. При каждом запуске или останове сервера в журнал ошибок записывается информация, подобная показанной в этих строках.

Включение регистрации запросов и бинарной регистрации

Для того чтобы сервер выполнял регистрацию запросов и бинарную регистрации, ее, в отличие от регистрации ошибок, необходимо сначала явно активизировать. В этом разделе объясняется, как реализовать каждый из этих типов регистрации и как затем можно просматривать файлы журналов.

Настройка регистрации запросов

В журнале запросов, который еще называют общим журналом запросов, регистрируются все соединения с сервером, выполненные сервером SQL-операторы и другие события, такие как запуск и выключение сервера. После установки MySQL регистрация запросов не включена. Чтобы ее активизировать, в раздел `[mysqld]` файла опций необходимо добавить следующую команду:

```
log[=<путь/имя_файла>]
```

Если путь и имя файлы не указаны, то файл журнала запросов будет создан в каталоге данных под именем `<хост>.log`. Поскольку журнал запросов является текстовым файлом, его содержимое можно просматривать в любом текстовом редакторе, таком как Notepad или Vim. Запросы заносятся в журнал в порядке их поступления на сервер. (Он может отличаться от порядка, в котором сервер выполняет эти запросы.) Каждый запрос сохраняется в журнале в виде одной строки. Например, следующие две записи в журнале запросов представляют выполненные сервером SQL-операторы `SELECT` и `SET`:

```
061006 14:59:58    16 Query    SELECT @@foreign_key_checks
061006 15:00:08    16 Query    SET FOREIGN_KEY_CHECKS=0
```

Журнал запросов особенно полезен, когда нужно проследить, кто подключается к серверу, откуда этот пользователь пришел, и какие операторы он выполняет. Это помогает при поиске и устранении неисправностей или отладке системы, поскольку позволяет точно определить происхождение и вид выполненного оператора. Так как в журнале запросов регистрируется каждый выполненный оператор, а также данные обо всех соединениях и другая информация, файл журнала может достичь значительных размеров и повлиять на производительность системы. Поэтому вместо журнала запросов предпочтительнее использовать бинарный журнал.

Настройка бинарного журнала

По сравнению с записью в журнал запросов, в бинарный журнал данные заносятся в более рациональном двоичном формате и регистрируются только те операторы, которые изменяют данные или могут их изменить. Например, в этот журнал будет добавлен оператор `DELETE`, не повлекший за собой удаление строк, потому что потенциально он мог привести к изменению данных. После установки MySQL бинарная регистрация, как и регистрация запросов, не включена. Для ее активизации в раздел `[mysqld]` файла опций потребуется добавить следующую команду:

```
log-bin[=<путь/имя_файла>]
```

Если путь и имя файлы не указаны, то файл будет создан в каталоге данных с именем `<хост>-bin.000001`. Если файл бинарного журнала уже существует, то у нового файла расширение будет на 1 больше номера последнего из существующих файлов этого журнала. Кроме того, при настройке бинарного журнала создается индексный файл с именем `<хост>-bin.index`, который содержит имена всех находящихся в работе файлов бинарного журнала. Имя и расположение этого файла можно изменить, добавив в раздел `[mysqld]` файла опций следующую команду:

```
log-bin-index[=<путь/имя_файла>]
```

Если имя задаваемого файла журнала или индексного файла содержит расширение, то это расширение автоматически удаляется, а вместо него MySQL использует для файла журнала числовое расширение, а для индексного файла — расширение `.index`.

Поскольку файлы бинарного журнала и индексные файлы создаются в двоичном виде, для их просмотра необходимо использовать утилиту `mysqlbinlog`. При вызове `mysqlbinlog` ей нужно указать имя просматриваемого файла журнала, как показано в следующем описании синтаксиса этой утилиты:

```
mysqlbinlog <хост>-bin.<числовое_расширение>
```

Например, предположим нужно просмотреть содержимое файла бинарного журнала с именем `Server21-bin.000001`. Это можно сделать с помощью такой команды:

```
mysqlbinlog Server21-bin.000001
```

MySQL записывает данные для каждого события, отслеживаемого в бинарном журнале. Например, следующая запись бинарного журнала свидетельствует о том, что в таблицу `t1` были добавлены новые данные:

```
# at 138
#061006 9:09:57  server id 1  log_pos 138  Query  thread_id=1  exec_time=0  error_
SET TIMESTAMP=1097078997;
INSERT INTO t1 VALUES (12);
```

Приведенные данные включают идентифицирующую информацию о записи журнала и выполнении оператора, временную метку и фактический оператор `INSERT`. (Обратите внимание, что здесь вторая строка усечена после `error_`, но в других случаях она может быть перенесена на следующую строку или усечена в другой позиции.) Каждый раз при регистрации события в этот журнальный файл добавляется аналогичная запись. После перезапуска сервера или очистки журналов создается новый файл журнала с числовым расширением, увеличенным на 1. (Операция очистки приводит к закрытию и повторному открытию файлов журналов. В случае файлов бинарного журнала также создается новый журнальный файл. Очистку журналов можно выполнить либо с помощью оператора `FLUSH LOG`, либо с помощью команды `flush-logs` утилиты `mysqladmin`.)

Очистка журналов — это составная часть общего процесса обслуживания файлов журналов. При включенном режиме журнальной регистрации файлы журналов могут достигать огромных размеров. Вследствие этого администраторы должны тщательным образом продумывать стратегию обслуживания журнальных файлов. Планирование обслуживания такого рода выходит за рамки этой книги; однако если вам придется заниматься этой задачей, за информацией об очистке журналов, ротации журнальных файлов, истечении срока хранения и других аспектах обслуживания журнальных файлов вам следует обратиться к документации, поставляемой в составе сервера MySQL, а также другой связанной каким-либо образом с MySQL документации.

После того, как вы получили представление о работе с журналом запросов и бинарным журналом обновлений, можно приступить к использованию полученных знаний на практике. В следующих двух упражнениях вы попробуете активизировать ведение журналов запросов и обновлений в системах Linux и Windows. Первое упражнение посвящено работе с журналами в системе Linux, а второе — в системе Windows. В обоих упражнениях вы останавливаете и перезапускаете сервер MySQL, а также

вносите изменения в файл опций. Поскольку в Windows сервер MySQL инсталлируется в виде службы, то второе упражнение содержит дополнительный шаг, в котором вы сначала удаляете эту службу, а затем создаете ее вновь так, что она использует нужный файл опций.

Практическое занятие

Активизация ведения журнала запросов и бинарного журнала в Linux

В последующих шагах описано, как в Linux активизировать регистрацию работы сервера в журнале запросов и бинарном журнале обновлений.

1. Откройте окно терминала (если вы работаете в графической среде) или перейдите в командную строку используемой оболочки. В любом случае вы должны находиться в домашнем каталоге пользователя root.

2. Сначала необходимо добавить несколько параметров в файл опций. Для этого с помощью текстового редактора Vim откройте на редактирование файл опций `.my.cnf`, выполнив следующую команду:

   ```
   vi .my.cnf
   ```

   Команда `vi` системы Linux открывает текстовый редактор Vim, предназначенный для редактирования текстовых файлов.

3. Для того чтобы можно было приступить к редактированию, включите режим вставки, введя следующую команду:

   ```
   i
   ```

4. Перейдите к разделу, начинающемуся с заголовка `[mysqld]` и добавьте под ним пустую строку.

5. Чтобы активизировать журнал запросов и бинарный журнал обновлений, добавьте под заголовком `[mysqld]` следующие строки:

   ```
   log
   log-bin
   ```

6. Нажмите клавишу <Esc>, чтобы выйти из режима вставки.

7. Чтобы указать редактору Vim, что редактирование закончено, введите следующую команду:

   ```
   :
   ```

   После ввода этой команды внизу экрана появится символ двоеточия, а курсор будет находиться в позиции сразу за двоеточием.

8. После этого введите следующую команду и нажмите клавишу <Enter>:

   ```
   wq
   ```

   По команде `w` редактор Vim сохраняет изменения при выходе, а по команде `q` завершает свою работу. После выполнения этих команд вы вернетесь в командную строку.

9. Для того чтобы сделанные в файле опций изменения вступили в силу, сервер MySQL потребуется сначала завершить, после чего запустить его вновь. Для завершения работы сервера выполните следующую команду:

   ```
   mysqladmin -u root -p shutdown
   ```

   На запрос ввода пароля введите пароль пользователя root и нажмите клавишу <Enter>. Должно появиться сообщение, свидетельствующее о том, что сервер MySQL завершил свою работу.

10. Теперь сервер MySQL нужно перезапустить. Для этого выполните следующую команду:

    ```
    mysqld_safe -u mysql &
    ```

    Вы должны получить сообщение, указывающее на то, что сервер успешно стартовал. Если после выполнения команды `mysqld_safe` приглашение на ввод команд не появилось, нажмите клавишу <Enter> и приглашение появится.

11. Перейдите в каталог данных и убедитесь, что в нем появились следующие три файла:

    ```
    <хост>.log
    <хост>-bin.000001
    <хост>-bin.index
    ```

12. Следующая часть этого упражнения состоит в проверке того, что сделанные изменения зарегистрированы в журнальных файлах. Используя следующую команду, подключитесь к базе данных `test`:

    ```
    mysql test
    ```

    Должна появиться командная строка `mysql`.

13. Затем создайте таблицу, добавьте в нее строку, удалите таблицу и выйдите из утилиты `mysql`. Для этого выполните следующие команды:

    ```
    CREATE TABLE t1 (c1 INT);
    INSERT INTO t1 VALUES (12);
    SELECT * FROM t1;
    DROP TABLE t1;
    exit
    ```

    Вы должны вернуться в командную строку оболочки (в каталог данных).

14. Просмотрите с помощью редактора Vim файл `<хост>.log`, выполнив следующую команду:

    ```
    vi <хост>.log
    ```

    Кроме информации о версии сервера, номере порта и соединении, это файл должен содержать следующие данные:

    ```
    Time             Id Command    Argument
    061001 4:33:09    1 Connect    root@localhost on test
    061001 4:33:49    1 Query      CREATE TABLE t1 (c1 INT)
    061001 4:34:01    1 Query      INSERT INTO t1 VALUES (12)
    061001 4:34:10    1 Query      SELECT * FROM t1
    061001 4:34:17    1 Query      DROP TABLE t1
    061001 4:34:22    1 Quit
    ```

15. Закончив просмотр содержимого файла журнала, введите следующую команду:

    :

    После ввода этой команды внизу экрана появится символ двоеточия, а курсор будет находиться в позиции сразу за двоеточием.

16. Наберите следующую команду после двоеточия и нажмите <Enter>:

    q

    Вы вернетесь в командную строку (в каталог данных).

17. С помощью утилиты mysqlbinlog просмотрите файл <хост>-bin.000001. Для этого выполните следующую команду:

    mysqlbinlog <хост>-bin.000001

    Помимо информации о версии и соединении этот файл должен содержать следующие данные:

```
# at 79
#061005 18:40:08 server id 1  log_pos 79  Query  thread_id=1  exec_time=0  error_
use test;
SET TIMESTAMP=1097026808;
CREATE TABLE t1 (c1 INT);
# at 138
#061005 18:40:08 server id 1  log_pos 138 Query  thread_id=1  exec_time=0  error_
SET TIMESTAMP=1097026808;
INSERT INTO t1 VALUES (12);
# at 199
#061005 18:40:08 server id 1  log_pos 199 Query  thread_id=1  exec_time=0  error_
SET TIMESTAMP=1097026808;
DROP TABLE t1;
```

    Отметим, что в зависимости от размеров окна просмотра, длинные строки могут быть усечены или перенесены на следующую строку.

Описание полученных результатов

Чтобы активизировать процесс регистрации работы сервера в Linux, необходимо сначала отредактировать файл опций, который расположен в домашнем каталоге пользователя root. (Файл опций .my.cnf был создан вами в главе 3. За подробным описанием этого файла обращайтесь к этой главе.) Вы отредактировали файл опций, добавив в раздел [mysqld] следующие команды:

```
log
log-bin
```

Команда log включает ведение журнала запросов и создает журнальный файл <хост>.log. Команда log-bin активизирует ведение бинарного журнала обновлений и создает журнальный файл <хост>-bin.000001 и индексный файл <хост>-bin.index. Если файл бинарного журнала уже существует, то у нового файла расширение будет на 1 больше номера последнего существующего файла этого журнала.

После того, как изменения, произведенные в файле опций .my.cnf, были сохранены, вы с помощью следующей команды завершили работу сервера MySQL:

```
mysqladmin -u root -p shutdown
```

Для закрытия сервера MySQL в этой команде используется утилита mysqladmin с параметром shutdown. В этой команде указываются также учетная запись пользователя root и запрос на ввод пароля. После закрытия сервера вы с помощью следующей команды сразу же запустили его заново:

```
mysqld_safe -u mysql &
```

В этой команде для запуска сервера и наблюдения за ним используется сценарий mysqld_safe. В качестве пользователя в этой команде указан mysql; соответствующая учетная запись была создана во время инсталляции MySQL. (Более подробную информацию об этой учетной записи можно найти в главе 2.) Присутствие символа амперсанда (&) в конце команды является специфичным для Linux и указывает на то, что этот сценарий должен работать в фоновом режиме, обеспечивая таким образом поддержку функций мониторинга этого сценария.

Перезапустив сервер MySQL, вы просмотрели оглавление каталога данных и убедились, что в нем появились три новых журнальных файла. Затем с помощью утилиты mysql вы выполнили несколько SQL-операторов, после чего в текстовом редакторе Vim проверили содержимое файла <хост>.log. Этот журнальный файл должен был содержать четыре выполненных вами SQL-оператора, а также дату и время их выполнения.

Потом вы с помощью следующей команды проверили содержимое файла <хост>-bin.000001:

```
mysqlbinlog <хост>-bin.000001
```

Команда mysqlbinlog предназначена для просмотра содержимого файлов бинарного журнала и индексного файла. Этот журнальный файл должен содержать SQL-операторы, которые изменяют данные (CREATE TABLE, INSERT и DROP TABLE), но не оператор SELECT.

Практическое занятие

Активизация ведения журнала запросов и бинарного журнала в Windows

В приведенных ниже шагах описано, как включить регистрацию работы сервера в журнале запросов и бинарном журнале обновлений в Windows.

1. Поскольку по умолчанию в Windows служба MySQL использует другой файл опций, нежели тот, с которым работают клиентские утилиты, вы можете удалить эту службу, а затем создать ее вновь таким образом, чтобы в качестве файла опций она использовала файл my.ini. Прежде чем удалить службу MySQL, ее необходимо сначала остановить. Откройте окно командной строки операционной системы и в ответ на приглашение введите следующую команду:

   ```
   net stop mysql
   ```

   Вы должны получить сообщение, свидетельствующее о том, что служба MySQL была остановлена.

2. Затем удалите из системы службу MySQL, выполнив следующую команду:

   ```
   mysqld-nt --remove
   ```

   Должно появиться сообщение, свидетельствующее о том, что эта служба удалена.

3. После этого службу MySQL необходимо обратно добавить в Windows, только теперь в качестве конфигурационного эта служба должна использовать файл `my.ini`. Выполните следующую команду:

   ```
   "c:\program files\mysql\mysql server 5.0\bin\mysqld-nt" --install MySQL
   --defaults-file="c:\windows\my.ini"
   ```

   Должно появиться сообщение, свидетельствующее об успешной установке службы.

4. Прежде чем начать работу с MySQL, эту службу необходимо запустить. Для этого введите следующую команду:

   ```
   net start mysql
   ```

   Вы должны получить сообщение, которое указывает, что служба MySQL стартовала.

5. Следующий шаг состоит в активизации ведения журналов в файле опций `my.ini`. Откройте файл `C:\WINDOWS\my.ini` в текстовом редакторе, таком как Notepad. Перейдите в раздел, начинающийся с `[mysqld]`, и добавьте пустую строку под заголовком `[mysqld]`.

6. Добавьте под заголовком `[mysqld]` следующие строки:

   ```
   log
   log-bin
   ```

7. Сохраните и закройте файл `my.ini`.

8. Чтобы сделанные в файле `my.ini` изменения вступили в силу, службу MySQL потребуется перезапустить. Остановите службу, выполнив следующую команду:

   ```
   net stop mysql
   ```

   Должно появиться сообщение о том, что служба MySQL остановлена.

   Можно было бы не запускать эту службу до тех пор, пока не внесены изменения в файл опций, пропустив таким образом шаги с 4 по 8. Однако запуск службы перед изменениями файла опций гарантирует, что служба MySQL добавлена в Windows должным образом и запускается без ошибок. Если вы отложите запуск этой службы до того момента, пока не будут внесены изменения в файл опций, и при ее запуске возникнут проблемы, вы не будете знать причину этой проблемы — возникла она из-за неправильной установки этой службы либо из-за изменений, внесенных в файл опций.

9. Запустите службу MySQL, выполнив следующую команду:

   ```
   net start mysql
   ```

   Вы должны получить сообщение, которое указывает, что служба MySQL стартовала.

10. Перейдите в каталог `C:\Program Files\MySQL\MySQL Server 5.0\data` и убедитесь, что в каталоге данных появились три следующих файла:

    ```
    <хост>.log
    <хост>-bin.000001
    <хост>-bin.index
    ```

11. Воспользуйтесь утилитой `mysql` для инициализации журнальных файлов. Введите следующую команду:

    ```
    mysql test
    ```

    Вы должны подключиться к MySQL, о чем свидетельствует появившаяся командная строка `mysql`.

12. Затем создайте таблицу, добавьте в нее строку, удалите таблицу и выйдите из утилиты `mysql`. Для этого выполните следующие SQL-операторы:

    ```
    CREATE TABLE t1 (c1 INT);
    INSERT INTO t1 VALUES (12);
    SELECT * FROM t1;
    DROP TABLE t1;
    exit
    ```

    Каждый выполненный оператор должен вернуть соответствующее сообщение. После выполнения последнего оператора вы должны вернуться в командную строку оболочки (в каталог данных).

13. С помощью текстового редактора, такого как Notepad, проверьте содержимое файла `<хост>.log`. Кроме информации о версии, номере порта и соединении, этот файл должен содержать следующие данные:

    ```
    Time                Id  Command        Argument
    061001 4:33:09      1   Connect        root@localhost on test
    061001 4:33:49      1   Query          CREATE TABLE t1 (c1 INT)
                        1   Query          INSERT INTO t1 VALUES (12)
                        1   Query          SELECT * FROM t1
                        1   Query          DROP TABLE t1
                        1   Quit
    ```

14. Завершив просмотр файла журнала, закройте его.

15. С помощью утилиты `mysqlbinlog` просмотрите файл `<хост>-bin.000001`. Для этого введите следующую команду:

    ```
    mysqlbinlog <хост>-bin.000001
    ```

    Помимо информации о версии и соединении, этот файл должен содержать следующие данные:

    ```
    # at 79
    #061005 18:40:08 server id 1  log_pos 79  Query  thread_id=1  exec_time=0  error_
    use test;
    SET TIMESTAMP=1097026808;
    CREATE TABLE t1 (c1 INT);
    # at 138
    #061005 18:40:08 server id 1  log_pos 138 Query  thread_id=1  exec_time=0  error_
    SET TIMESTAMP=1097026808;
    INSERT INTO t1 VALUES (12);
    # at 199
    #061005 18:40:08 server id 1  log_pos 199 Query  thread_id=1  exec_time=0  error_
    SET TIMESTAMP=1097026808;
    DROP TABLE t1;
    ```

Описание полученных результатов

Чтобы объединить параметры конфигурации в одном файле, необходимо удалить службу MySQL, а затем создать ее заново. Поэтому на первом шаге вы с помощью следующей команды остановили службу MySQL:

```
net stop mysql
```

Эта команда останавливает службу. Это значит, что теперь к серверу MySQL доступа нет. Прежде чем удалить службу, ее необходимо остановить. После того как служба была остановлена, вы выполнили следующую команду:

```
mysqld-nt --remove
```

Для удаления службы MySQL из Windows используется команда `mysqld-nt` с опцией `--remove`. После того как служба была удалена, вы с помощью следующей команды добавили новую службу:

```
"c:\program files\mysql\mysql server 5.0\bin\mysqld-nt" --install MySQL
--defaults-file="c:\windows\my.ini"
```

В этой команде заданы путь и имя исполняемого файла сервера `mysqld-nt` и опции `--install` и `--defaults-file`. Опция `-install` указывает, что новая служба создается с именем MySQL, а опция `--defaults-file` — на то, что при старте эта служба будет использовать файл опций `my.ini`, расположенный в каталоге `C:\WINDOWS`.

Создав службу MySQL, вы затем запустили ее с помощью следующей команды:

```
net start mysql
```

Это позволяет убедиться, что служба MySQL установлена без ошибок. Проверить работу сервера можно с помощью утилиты `mysql`, которая работает только тогда, когда можно установить соединение с сервером MySQL (что означает, что служба MySQL установлена и работает нормально).

Следующий шаг после добавления новой службы MySQL — включение регистрации работы сервера. Для этого сначала нужно отредактировать файл опций, расположенный в каталоге `C:\WINDOWS`. (Файл опций `my.ini` был вами создан в главе 3. За подробным описанием этого файла обращайтесь к этой главе.) Вы отредактировали файл опций, добавив в раздел `[mysqld]` следующие строки:

```
log
log-bin
```

Команда `log` активизирует регистрацию запросов в журнале запросов и создает файл `<хост>.log`. Команда `log-bin` включает регистрацию изменений в бинарном журнале и создает файлы `<хост>-bin.000001` и `<хост>-bin.index`. Если файл бинарного журнала уже существует, то у нового файла расширение будет на 1 больше номера последнего из существующих файлов этого журнала.

После модификации файла опций вы опять остановили и перезапустили службу MySQL. Затем вы просмотрели оглавление каталога данных и убедились, что в каталог были добавлены три новых журнальных файла. После этого вы, используя утилиту `mysql`, выполнили несколько SQL-операторов. После выхода из этой утилиты вы с помощью текстового редактора просмотрели содержимое файла `<хост>.log`. Этот журнальный файл должен был содержать записи о четырех выполненных вами SQL-операторах, а также дату и время их выполнения.

Затем с помощью следующей команды вы просмотрели содержимое файла <хост>-bin.000001:

```
mysqlbinlog <host>-bin.000001
```

Команда mysqlbinlog предназначена для просмотра файлов бинарного журнала обновлений и индексного файла. Этот журнальный файл должен содержать SQL-операторы, которые изменяют данные (CREATE TABLE, INSERT и DROP TABLE), но не оператор SELECT.

Резюме

В этой главе было рассмотрено несколько тем, связанных с администрированием сервера MySQL. Вы научились просматривать и модифицировать системные настройки, предпринимать конкретные действия, влияющие на работу сервера MySQL, и работать с различными журналами. Ниже перечислены административные задачи, которые вы научились решать.

- С помощью утилиты mysqladmin просматривать системную информацию и корректировать работу сервера.
- С помощью оператора SHOW VARIABLES просматривать серверные системные переменные.
- С помощью оператора SELECT просматривать значения динамических системных переменных.
- С помощью оператора SHOW STATUS получать значения серверных переменных состояния.
- Задавать системные настройки при запуске сервера.
- Задавать системные настройки в файле опций.
- Задавать системные настройки в процессе работы сервера.
- Просматривать файл журнала ошибок.
- Активизировать ведение журнала запросов и просматривать файлы этого журнала.
- Активизировать ведение бинарного журнала обновлений и просматривать файлы этого журнала.

Теперь, зная о способах выполнения в MySQL основных административных функций, вы готовы перейти к следующей теме – безопасности. Понимание того, как обеспечивается безопасность системы, может помочь защитить информацию, хранящуюся в ваших базах данных, и разрешать доступ к ней только определенному кругу лиц, предотвращая тем самым просмотр и изменение данных другими пользователями системы. После этого вы перейдете к оптимизации производительности сервера, реализации репликации, резервного копирования баз данных и, при необходимости, их восстановления из резервных копий. Большинство из того, что вы узнали в этой главе, составляет основу для выполнения административных функций, описываемых в последующих главах.

Упражнения

В этой главе вы научились выполнять целый ряд административных операций. Для закрепления полученных навыков предлагается выполнить приведенные ниже упражнения. Ответы вы сможете найти в приложении А.

1. Напишите команду, которая записывает на диск открытые таблицы из кэша, закрывает, а затем повторно открывает журнальные файлы и перечитывает таблицы привилегий. Кроме того, она должна возвращать информацию о состоянии сервера MySQL. Эта команда должна выполняться от имени пользователя `myadmin` и должна выдавать подсказку на ввод пароля.

2. Составьте SQL-оператор, который выводит все глобальные серверные системные переменные, содержащие в своем имени подстроку `max`, а также значения этих переменных.

3. Составьте SQL-оператор, который выводит значение динамической системной переменной `query_cache_limit` для текущего сеанса.

4. Составьте SQL-оператор, который выводит на экран все серверные переменные состояния, содержащие в своем имени подстроку `cache`, а также значения этих переменных.

5. Составьте SQL-оператор, который присваивает динамической системной переменной `max_tmp_tables` текущего сеанса значение 24.

6. Добавьте в файл опций команды, которые активизируют регистрацию обновлений в бинарном журнале и создают требуемый файл бинарного журнала и индексный файл. Создаваемые файлы должны располагаться в каталоге данных.

7. Напишите команду, которая отображает содержимое файла бинарного журнала `Server21-bin.000327`.