TOPAZ Maria DB

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Москва 2022

ОГЛАВЛЕНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ

1. АРМ Автоматизированное рабочее место
2. АСУ ТП, АСУТП Автоматизированная система управления технологическими процессами
3. БД База данных
4. Заказчик Лицо, заинтересованное в выполнении исполнителем работ, оказании им услуг или приобретении у продавца какого- либо продукта (в широком смысле).
5. ИБ Информационная безопасность
6. ИС Информационная система
7. ИТ Информационные технологии
8. Компания ООО ИК «СИБИНТЕК»
9. ЛВС Локальная вычислительная сеть
10. ОС Операционная система
11. ОУ Объект учета
12. ПАЗ Противоаварийная защита
14. ПО Программное обеспечение
15. СЗИ Средства защиты информации
16. СУБД Система управления базами данных
17. ТО Техническое обслуживание
18. УЗ Учетная запись
19. ФИО Фамилия Имя Отчество
20. ФР Функциональная роль
21. ЦДС Центральная диспетчерская служба Компании, осуществляющая первичный прием обращений
22. API Интерфейс прикладного программирования

1 ОБЩИЕ СВЕДЕНИЯ

1.2 НАИМЕНОВАНИЕ СИСТЕМЫ

Полное наименование системы: Система управления базами данных TOPAZ MARIADB.

Условное обозначение (краткое наименование): Система, СУБД.

1.3 НАЗНАЧЕНИЕ СИСТЕМЫ

СУБД TOPAZ MARIADB выполняет функцию управляющей программы по отношению к данным базы данных. Система предназначена для обеспечения набор программных модулей, обеспечивающих работоспособность комплекса.

СУБД обеспечивает сохранение и накопление собираемых и расчетных данных в виде массивов информации.

1.4 ФУНКЦИИ СИСТЕМЫ

Система обеспечивает выполнение следующих функций:

- предварительная обработка полученных данных брокером сообщений;
- запись данных, полученных от подсистемы взаимодействия со смежными системами;
- запись предварительно обработанных данных;
- запись данных в архив по событию или с заданной периодичностью в соответствии с настраиваемыми параметрами;
- долговременное хранение временных рядов, полученных данных;
- выборку архивных данных по запросам от подсистемы инженерных расчетов;
- сохранение показателей, рассчитанных в подсистеме инженерных расчетов.

2 ОБЩИЕ СВЕДЕНИЯ ОБ АДМИНИСТРИРОВАНИИ СИСТЕМЫ

2.2 ПОРЯДОК УПРАВЛЕНИЯ ДОСТУПАМИ ПОЛЬЗОВАТЕЛЕЙ

К работе с Системой допускаются пользователи, ознакомленные с Руководством пользователя, Руководством по предоставлению доступа на Систему и руководящими документами в области информационной безопасности.

Предоставление доступа привилегированным пользователям осуществляется согласно требованиям информационной безопасности Компании.

В случае необходимости внесения изменений в перечень привилегированных пользователей, Владелец Системы организует обновление соответствующего эксплуатационного документа на Систему, обновленная версия этого документа согласуется и утверждается установленным порядком. На основании данного утвержденного эксплуатационного документа осуществляется предоставление, изменение, прекращение доступа привилегированных пользователей к Системе. Процедура обновления этого эксплуатационного документа в обязательном порядке должна быть проведена в случае утраты оснований для доступа привилегированного пользователя к Системе.

Учетные записи привилегированных пользователей Системы должны удовлетворять требованиям информационной безопасности Компании.

2.3 ПРИВИЛЕГИРОВАННЫЕ РОЛИ

Для администрирования Системы рекомендуется определять следующие роли привилегированных пользователей:

- Системный администратор;
- Администратор СУБД;
- Администратор приложений;

Конкретный состав администраторов определяется политикой информационной безопасности Компании.

Основными обязанностями роли «Системный администратор» являются:

- установка, настройка, резервное копирование и администрирование программных и технических средств, применяемых в Системе, в том числе:
    - установка и обслуживание системного и вспомогательного ПО серверов, на которых функционирует Система;
    - регулярное и своевременное обновление компонент Системы;
    - администрирование системного ПО и аппаратного обеспечения серверов Системы;
    - обеспечение и настройка резервного копирования, хранения и восстановления системного ПО;
    - установка, настройка и мониторинг работоспособности комплекса технических средств Системы, а также серверов Системы;
    - оптимизация работы серверов Системы.

Основными обязанностями роли «Администратор СУБД» являются:

- обеспечение и настройка хранения и восстановления баз данных, входящих в состав Системы;
- настройка и мониторинг параметров программного обеспечения СУБД;
- мониторинг производительности и доступности СУБД;
- оптимизация функционирования СУБД;
- организация и контроль восстановления баз данных Системы из резервных копий.

Основными обязанностями роли «Администратор приложений» являются:

- мониторинг работоспособности Системы (Web-портал);
- разработка и модификация отчетных форм, мнемосхем;
- предоставление, изменение и изъятие прав доступа привилегированных пользователей к Системе;
- администрирование информационного обмена со смежными системами:
    - заведение учетных тегов в интерфейсе Системы (Web-портал);
    - настройка конфигураций на станциях сбора данных;
    - проверка корректности интеграции со смежными системами;
- консультирование пользователей по телефону и корпоративной электронной почте.

2.4 НЕПРИВИЛЕГИРОВАННЫЕ РОЛИ

Для эксплуатации Системы определены следующие роли непривилегированных пользователей:

Оператор (пользователю – обладателю данной ФР доступен ручной ввод данных, просмотр мнемосхем, отчётов по соответствующему структурному подразделению (цеху) предприятия).

Специалист (пользователю – обладателю данной ФР доступен, просмотр мнемосхем, отчётов, корректировка данных по соответствующему структурному подразделению (цеху) предприятия).

Руководитель (пользователю – обладателю данной ФР доступен только просмотр внесённых данных и формирование отчёта по соответствующему структурному подразделению (цеху) предприятия).

Обучение непривилегированных пользователей рассмотрено в разделе 2.8 документа

«Описание жизненного цикла».

2.5 ПРАВА ДОСТУПА К ИНФОРМАЦИОННЫМ РЕСУРСАМ СИСТЕМЫ

Доступ к Системе регламентируется руководящими документами Компании. Администратор приложений выполняет необходимые настройки по правам доступа пользователей к Системе (предоставление, изменение, изъятие прав

доступа для конкретных пользователей).

Аутентификация пользователей в Системе осуществляется по личным учетным записям.

3 РАБОТА С БАЗОЙ ДАННЫХ
3.2 Основы работы с TOPAZ MariaDB

3.2.1 Вход в TOPAZ MariaDB

Вход на сервер TOPAZ MariaDB из командной строки осуществляется следующим образом:

```
mysql -u //user_name// -p -h //ip_address// //db_name//
```

Заменить *user_name* на ваше имя пользователя пользователя базы данных. Заменить *ip_address* с именем хоста или адресом вашего сервера. Если вы получаете доступ к TOPAZ MariaDB с того же сервера, на который вы вошли, не включайте-часи *ip адрес*. Заменить *db_name* на имя базы данных, к которой вы хотите получить доступ (например *test*, которая иногда уже создается для целей тестирования, запустив TOPAZ MariaDB-secure-installation/mysql_secure_installation, в этом случае вы можете не казывать *db_name*).

Когда будет предложено ввести пароль, введите его. Если ваш вход в систему выполнен успешно, вы должны увидеть что-то похожее на это:

```
MariaDB <[test]>
```

Здесь вы будете вводить все свои операторы SQL. Подробнее о них позже. А пока давайте посмотрим на компоненты подсказки: Часть «TOPAZ MariaDB» означает, что вы подключены к серверу базы данных TOPAZ MariaDB. Слово в скобках — это имя вашей базы данных по умолчанию, База данных *test* в этом примере.

3.2.2 Подключение к TOPAZ MariaDB

TOPAZ MariaDB — это система баз данных, сервер баз данных. Для взаимодействия с сервером TOPAZ MariaDB можно использовать клиентскую программу, используя API (интерфейс прикладного программирования) для взаимодействия с сервером TOPAZ MariaDB. Для работы с TOPAZ MariaDB исползуется mysql.

В командной строке Linux следует ввести следующее, чтобы войти в систему как пользователь root и войти в режим монитора:

```
mysql -u root -p -h localhost
```

Опция -u предназначена для указания имени пользователя. Вместо root Вы можете указать другое имя пользователя. Это имя пользователя TOPAZ MariaDB, а не имя пользователя Linux.

Опция -p указывает mysql клиенту запросить у Вас пароль. Если пароль для root пользователь не установлен, то пароль будет пустым, и достаточно нажать [Enter] при появлении запроса.

Опция -h предназначена для указания имени хоста или IP-адреса сервера. Это необходимо, если клиент работает на другой машине, чем сервер. Если вы подключились к серверной машине, вам, вероятно, не понадобится использовать данную опцию.

3.2.3 Создание структуры

Чтобы иметь возможность добавлять данные и управлять ими, сначала необходимо

создать структуру базы данных. Команда создания базы данных из клиент mysql выглядит следующим образом:

```
CREATE DATABASE bookstore;

USE bookstore;
```

Этот оператор SQL создаст подкаталог с именем bookstore в файловой системе Linux в каталоге, в котором хранятся ваши файлы данных TOPAZ MariaDB. Он создаст место для добавления таблиц, которые будут хранить данные. Второй оператор SQL выше установит эту новую базу данных в качестве базы данных по умолчанию. Он останется вашим по умолчанию, пока вы не измените его на другой или пока не выйдете из TOPAZ MariaDB.

В качестве примера, чтобы создать таблицу, в которой будут храниться основные данные о книгах, можно ввести следующее:

```
CREATE TABLE books (
isbn CHAR(20) PRIMARY KEY,
title VARCHAR(50),
author_id INT,
publisher_id INT,
year_pub CHAR(4),
description TEXT );
```

Этот оператор SQL создает таблицы с шестью полями или, скорее, столбцами. Первый столбец (isbn) представляет собой идентификационный номер для каждой строки. Он имеет тип символов фиксированной ширины, равный 20 символам. Это будет столбец первичного ключа, по которому будут индексироваться данные. Тип данных столбца для названия книги — это столбец символов переменной ширины, содержащий не более пятидесяти символов. Третий и четвертый столбцы будут использоваться для идентификационных номеров автора и издателя. Это целочисленные типы данных. Пятая колонка используется для года публикации каждой книги. Последний столбец предназначен для ввода описания каждой книги. Это тип данных TEXT, что означает, что это столбец переменной ширины, и он может содержать до 65535 байт данных для каждой строки. Есть несколько других типов данных, которые можно использовать для столбцов, но это дает хорошую выборку.

Чтобы увидеть, как выглядит созданная таблица, введите следующую инструкцию SQL:

```
DESCRIBE books;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| isbn         | char(20)    | NO   | PRI | NULL    |       |
| title        | varchar(50) | YES  |     | NULL    |       |
| author_id    | int(11)     | YES  |     | NULL    |       |
| publisher_id | int(11)     | YES  |     | NULL    |       |
| year_pub     | char(4)     | YES  |     | NULL    |       |
| description  | text        | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

Чтобы изменить настройки таблицы, вы можете использовать оператор ALTER TABLE. Чтобы полностью удалить таблицу (включая ее данные), вы можете использовать оператор DROP TABLE, за которым следует имя таблицы.

3.2.4 **Ввод данных**

Основным методом ввода данных в таблицу является использование оператора INSERT.:

```
INSERT INTO books
(title, author_id, isbn, year_pub)
VALUES("The Trial", "1", "0805210407", "1995"),
("The Metamorphosis", "1", "0553213695", "1995"),
("America", "1", "0805210644", "1995");
```

Мы даем ключевое слово VALUES только один раз, за которым следует отдельный набор значений для каждой книги, каждое из которых заключено в круглые скобки и разделено запятыми.

3.2.5 Извлечение данных

Основной метод извлечения данных из таблиц заключается в использовании оператора SELECT. В качестве примера, давайте получим список названий книг из таблицы books:

```
SELECT title
FROM books;
```

Это отобразит все строки книг в таблице. Если в таблице тысячи строк, TOPAZ MariaDB отобразит тысячи. Чтобы ограничить количество извлекаемых строк, мы могли бы добавить LIMIT к SELECT:

```
SELECT title
FROM books
LIMIT 5;
```

Это ограничит количество отображаемых строк до пяти. Чтобы иметь возможность указать имя автора для каждой книги вместе с названием, вам нужно будет соединить таблицу books с таблицей авторов. Для этого мы можем использовать JOIN:

```
SELECT title, name_last
FROM books
JOIN authors USING (author_id);
```

Notice that the primary table from which we're drawing data is given in the FROM clause. The table to which we're joining is given in the JOIN clause along with the commonly named column (i.e., author_id) that we're using for the join.

To retrieve the titles of only books written by Kafka based on his name (not the author_id), we would use the WHERE
clause with the SELECT statement. This would be entered like the following:

```
SELECT title AS "Kafka Books"
FROM books
JOIN authors USING (author_id)
WHERE  name_last  =  "kafka";

+-------------------------------+
| Kafka Books         |
+-------------------------------+
| The Castle          |
| The Trial           |
| The Metamorphosis |
| America             |
+-------------------------------+
```

В этом операторе будут перечислены названия книг Кафки, хранящиеся в базе данных. Обратите внимание, что я добавилВ КАЧЕСТВЕпараметр рядом с заголовком имени столбца, чтобы изменить заголовок столбца в наборе результатов Kafka Books. Это известно как псевдоним.

3.2.6 Изменение и удаление данных

Чтобы изменить существующие данные, распространенный метод заключается в использовании оператора UPDATE. Однако при изменении данных мы должны быть уверены, что изменяем правильные строки. В нашем примере это может быть еще одна книга с названием *Америка* написано другим автором. Поскольку ключевой столбец isbn имеет только уникальные номера, и мы знаем номер ISBN книги, которую хотим изменить, мы можем использовать его для указания строки.

```
UPDATE books
SET title = "Amerika"
WHERE isbn = "0805210644";
```

Это изменит значение столбца заголовка для указанной строки.

Для удаления строки можно использовать утверждение DELETE. Сначала запустив SELECT мы определяем идентификационный номер автора как 2034. Используя этот идентификационный номер автора, мы можем ввести следующее:

```
DELETE FROM books
WHERE author_id = "2034";
```

Этот оператор удалит все строки из таблиц для данного author_id. Чтобы сделать это правильно, мы должны сделать то же самое для таблицы авторов. Мы просто заменили бы имя таблицы в операторе выше; все остальное было бы так же.

3.3 Получение данных из TOPAZ MariaDB

Самый простой способ получить данные из TOPAZ MariaDB — использовать оператор SELECT.

3.3.1 Основные элементы

Основные, минимальные элементы SELECT вызов оператора по ключевому слову SELECT. Для выбора столбцов можно использовать звездочку в качестве подстановочного знака для выбора всех столбцов в конкретной таблице. Используя базу данных из вымышленного книжного магазина, можно ввести следующую инструкцию SQL, чтобы получить список всех столбцов и строк в таблице, содержащей информацию о книгах:

```
SELECT * FROM books;
```

Это позволит получить все данные, содержащиеся вкнигистол. Если мы хотим получить только определенные столбцы, мы должны перечислить их вместо звездочки в списке, разделенном запятыми, например:

```
SELECT isbn, title, author_id
FROM books;
```

Это сужает ширину набора результатов, извлекая только три столбца, но по-прежнему извлекает все строки в таблице. Если таблица содержит тысячи строк данных, это может быть больше данных, чем нам нужно. Если мы хотим ограничить результаты несколькими книгами, скажем, пятью, нужно включить оператор LIMIT:

```
SELECT isbn, title, author_id
FROM books
LIMIT 5;
```

Это даст первые пять строк, найденных в таблице. Если мы хотим получить следующие десять найденных, мы должны добавить параметр начальной точки непосредственно перед количеством отображаемых строк, разделенных запятой:

```
SELECT isbn, title, author_id
FROM books
LIMIT 5, 10;
```

3.3.2 Избирательность и порядок

Предыдущие операторы сузили количество извлекаемых столбцов и строк, но они не были очень избирательными. Предположим, что нам нужны только книги, написанные определенным автором, скажем, Достоевским. Глядя в таблицу авторов, мы обнаруживаем, что его идентификационный номер автора — 4729. Используя WHERE можно получить список книг из базы данных для этого конкретного автора следующим образом:

```
SELECT isbn, title
FROM books
WHERE author_id = 4729
LIMIT 5;
```

Операторы SQL, которые мы рассмотрели до сих пор, отображают названия книг в том порядке, в котором они находятся в базе данных. Если мы хотим расположить результаты в алфавитно-цифровом порядке на основе значений столбца заголовка, например, мы должны добавит оператор ORDER BY:

```
SELECT isbn, title
FROM books
WHERE author_id = 4729
ORDER BY title ASC
LIMIT 5;
```

Обратите внимание, что ORDER BY идет после WHERE и перед LIMIT. Этот оператор не только отобразит строки по названию книги, но и извлечет только первые пять в зависимости от порядка. То есть TOPAZ MariaDB сначала извлечет все строки на основе WHERE, упорядочит данные на основе ORDER BY, а затем отобразит ограниченное количество строк на основе LIMIT.

3.4 Добавление и изменение данных в TOPAZ MariaDB

Существует несколько способов добавления и изменения данных в TOPAZ MariaDB. Вы можете использовать несколько операторов SQL, каждый из которых имеет несколько параметров.

3.4.1 Добавление данных

Чтобы добавить данные в таблицу в TOPAZ MariaDB, вам нужно будет использовать команду INSERT. Его базовый минимальный синтаксис — это команда INSERT за которым следует имя таблицы, а затем ключевое слово VALUES со списком значений, разделенных запятыми, содержащихся в круглых скобках:

```
INSERT table1
VALUES("text1","text2","text3");
```

В данном примере текст добавляется в таблицу с именем table1, которая содержит только три столбца — то же количество значений, которое мы вставляем. Количество столбцов должно совпадать. Однако, если вы не хотите вставлять данные во все столбцы таблицы, вы можете назвать нужные столбцы:

```
INSERT INTO table1
(col3, col1)
VALUES("text3","text1");
```

В этом примере мы не только называем столбцы, но и перечисляем их в другом порядке. Это приемлемо для TOPAZ MariaDB. Только не забудьте перечислить значения в том же порядке. Если вы собираетесь вставлять данные в таблицу и хотите указать все значения, кроме одного (скажем, ключевой столбец, так как он автоматически увеличивается), вы можете просто указать значение DEFAULT, чтобы не перечислять столбцы.

Если у вас есть много строк данных для вставки в одну и ту же таблицу, может быть эффективнее вставить все строки в одну инструкцию SQL. Вставка нескольких строк может быть выполнена следующим образом:

```
INSERT IGNORE
INTO table2
VALUES("id1","text","text"),
("id2","text","text"),
("id2","text","text");
```

Обратите внимание, что ключевое слово VALUES используется только один раз, и каждая строка содержится в собственном наборе круглых скобок, каждый набор разделен запятыми. Мы намеренно добавили ошибку в этот пример: мы пытаемся вставить три строки данных в таблицу2, для которых первый столбец является поле UNIQUE. Введенная здесь третья строка имеет тот же идентификационный номер ключевого столбца, что и вторая строка. Обычно это приводит к ошибке, и ни одна из трех строк не вставляется. Однако, поскольку заявление имеет флаг IGNORE, дубликаты будут проигнорированы и не вставлены, но другие строки все равно будут вставлены. Таким образом, первая и вторая строки выше будут вставлены, а третья — нет.

3.4.2 Приоритет

Оператор INSERT имеет приоритет над операторами чтения (т. е. операторами SELECT). Оператор INSERT заблокирует таблицу и заставит других клиентов ждать, пока она не будет завершена. На загруженном сервере TOPAZ MariaDB, который имеет много одновременных запросов данных, это может привести к задержкам пользователей при запуске скрипта, выполняющего ряд операторов INSERT. Если вы не хотите, чтобы пользовательские запросы были приостановлены, и вы можете подождать, чтобы вставить данные, вы можете использовать флаг LOW_PRIORITY:

```
INSERT LOW_PRIORITY
INTO table1
VALUES("text1","text2","text3");
```

Флаг LOW_PRIORITY поставит оператор INSERT в очереди, ожидая завершения всех текущих и ожидающих запросов, прежде чем он будет выполнен. Если новые запросы выполняются во время ожидания оператора с низким приоритетом, то они помещаются перед ним в очередь. TOPAZ MariaDB не начинает выполнять оператор с низким приоритетом, пока не закончатся другие ожидающие запросы. Однако, как только транзакция начинается, таблица блокируется, и любые другие запросы данных из таблицы, которые поступают после ее запуска, должны ждать, пока она не будет завершена. Поскольку он блокирует таблицу, операторы с низким приоритетом предотвратят одновременные вставки от других клиентов, даже если вы имеете дело с таблицей MyISAM. Кстати, обратите внимание, что флаг LOW_PRIORITY стоит перед INTO.

Одно потенциальное неудобство оператора INSERT LOW_PRIORITY заключается в том, что клиент будет вынужден ждать успешного завершения этого утверждения. Поэтому, если вы вставляете данные на загруженный сервер с настройкой низкого приоритета с помощью клиента mysql, ваш клиент может быть заблокирован на несколько минут, а может и часов, в зависимости от того, насколько занят ваш сервер в данный момент. В качестве альтернативы ожиданию других клиентов с запросами на чтение или ожидания вашего клиента вы можете использовать флаг DELAYED вместо флага LOW_PRIORITY:

```
INSERT DELAYED
INTO table1
VALUES("text1","text2","text3");
```

TOPAZ MariaDB примет запрос как запрос с низким приоритетом и поместит его в свой список задач для выполнения, когда у него будет перерыв. Однако он немедленно освобождает клиента, чтобы клиент мог продолжить ввод других операторов SQL или даже выйти. Еще одним преимуществом этого метода является то, что несколько INSERT DELAYED запросы объединяются для вставки блока, когда есть пробел, что делает процесс потенциально быстрее, чем INSERT LOW_PRIORITY. Недостаток этого выбора, однако, заключается в том, что клиенту никогда не сообщается, успешно ли выполнена отложенная вставка или нет. Клиент информируется о сообщениях об ошибках при вводе оператора — оператор должен быть действительным, прежде чем он будет поставлен в очередь, — но ему не сообщается о проблемах, которые возникают после его принятия.

3.4.3 Замена данных

Когда вы добавляете большие объемы данных в таблицу с ключевым полем, как упоминалось ранее, вы можете использовать флаг IGNORE, чтобы предотвратить вставку дубликатов, но по-прежнему разрешать ввод уникальных строк. Однако могут быть случаи, когда вы действительно хотите заменить строки с теми же ключевыми полями новыми. В такой ситуации вместо использования INSERT вы можете использовать оператор REPLACE:

```
REPLACE LOW_PRIORITY
INTO table2 (id, col1, col2)
VALUES("id1","text","text"),
("id2","text","text"),
("id3","text","text");
```

Обратите внимание, что синтаксис такой же, как у оператора INSERT. Эти флаги имеют одинаковый эффект. Кроме того, можно вставить несколько строк, но нет необходимости в флаге IGNORE, так как дубликатов не будет — оригиналы просто будут перезаписаны. На самом деле, когда строка заменяется, она сначала полностью удаляется, а затем вставляется новая строка. Любым столбцам без значений в новой строке будут присвоены значения по умолчанию для столбцов. Ни одно из значений старой строки не сохраняется. REPLACE также позволит вам комбинировать его с оператором SELECT, как мы видели с оператором INSERT заявление ранее.

3.4.4 Обновление данных

Если вы хотите изменить данные, содержащиеся в существующих записях, но только для определенных столбцов, вам нужно будет использовать UPDATE утверждение. Синтаксис для UPDATE несколько отличается от показанного ранее синтаксиса для INSERT и REPLACE заявления:

```
UPDATE LOW_PRIORITY table3
SET col1 = "text-a", col2="text-b"
WHERE id < 100;
```

В операторе SQL здесь мы изменяем значение двух столбцов, названных по отдельности, с помощью SET пункт. Пункт SET опционально может использоваться в операторах INSERT и REPLACE, но это исключает возможность использования нескольких строк. В заявлении выше мы также используем WHERE чтобы определить, какие записи были изменены: обновляются только строки с идентификатором, значение которого меньше 100.

Полезная функция оператора UPDATE заключается в том, что он позволяет использовать текущее значение столбца для обновления того же столбца. Например, предположим, что вы хотите добавить один день к значению столбца даты, где дата — воскресенье. Вы можете сделать следующее:

```
UPDATE table5
SET col_date = DATE_ADD(col_date, INTERVAL 1 DAY)
WHERE DAYOFWEEK(col_date) = 1;
```

Для строк, где днем недели является воскресенье, функция DATE_ADD() примет значение col_date перед его обновлением и добавит к нему один день. Затем TOPAZ MariaDB возьмет эту сумму и установит для нее col_date.

Есть еще пара поворотов, которые вы теперь можете сделать с UPDATE заявление: если вы хотите обновить строки в определенном порядке, вы можете добавить пункт ORDER BY. Вы также можете ограничить количество строк, которые обновляются с помощью пункт LIMIT. Ниже приведен пример обоих этих положений:

```
UPDATE LOW_PRIORITY table3
SET col1="text-a", col2="text-b"
WHERE id < 100
ORDER BY col3 DESC
LIMIT 10;
```

Порядок может быть нисходящим, как указано здесьDESCили по возрастанию либо с флагом ASC, либо просто опустив его, так как по возрастанию используется значение по умолчанию. LIMIT Предложение, конечно, ограничивает количество затрагиваемых строк до десяти.

Если вы хотите обратиться к нескольким таблицам в одном UPDATE заявление, вы можете сделать это следующим образом:

```
UPDATE table3, table4
SET table3.col1 = table4.col1
WHERE table3.id = table4.id;
```

Здесь мы видим соединение между двумя таблицами с именем. В таблице 3 значение col1 устанавливается равным значению того же столбца в таблице 4, где значения id из каждого совпадают. Мы не обновляем обе таблицы здесь; мы просто обращаемся к обоим. Мы должны указать имя таблицы для каждого столбца, чтобы предотвратить ошибку двусмысленности. Операторы ORDER BY и LIMIT не допускаются с несколькими обновлениями таблицы.

Есть еще одна комбинация, которую вы можете сделать с оператором INSERT. Это включает в себя оператор UPDATE. При вставке нескольких строк данных, если вы хотите отметить, в каких строках есть потенциально повторяющиеся записи, а какие являются новыми, вы можете добавить столбец с именем status и соответствующим образом изменить его значение с помощью оператора:

```
INSERT IGNORE INTO table1
(id, col1, col2, status)
VALUES("1012","text","text","new"),
("1025,"text","text","new"),
("1030,"text","text","new")
ON DUPLICATE KEY
UPDATE status = "l";
```

Поскольку флаг IGNORE, ошибки не будут генерироваться, дубликаты не будут вставлены или заменены, а остальное будет добавлено. Из-за ON DUPLICATE KEY , статус столбца исходной строки будет установлен на старый, если будут повторяющиеся попытки ввода. Остальные будут вставлены, и их статус будет установлен на новый.

3.5 Изменение таблиц в TOPAZ MariaDB

3.5.1 Перед началом

Для примеров в этой статье мы будем ссылаться на базу данных с именем db1 содержащий таблицу с названием clients. Таблица clients предназначена для отслеживания имен и адресов клиентов. Для того, что бы , чтобы увидеть, как выглядит таблица следует ввести команду DESCRIBE:

```
DESCRIBE clients;

+--------------------+--------------------+---------+--------+-------------+----------+
| Field              | Type               | Null    | Key    | Default     | Extra    |
+--------------------+--------------------+---------+--------+-------------+----------+
| cust_id            | int(11)            |         | PRI    | 0           |          |
| name               | varchar(25)        | YES     |        | NULL        |          |
| address            | varchar(25)        | YES     |        | NULL        |          |
| city               | varchar(25)        | YES     |        | NULL        |          |
| state              | char(2)            | YES     |        | NULL        |          |
| zip                | varchar(10)        | YES     |        | NULL        |          |
| client_type        | varchar(4)         | YES     |        | NULL        |          |
+--------------------+--------------------+---------+--------+-------------+----------+
```

Прежде чем вносить какие-либо структурные изменения в таблицу в TOPAZ MariaDB, особенно если она содержит данные, необходимо сделать резервную копию таблицы, подлежащей изменению. Есть несколько способов сделать это, но некоторые из них могут быть запрещены вашей веб-хостинговой компанией. Даже если ваша база данных находится на вашем собственном сервере, утилита mysqldump, как правило, является лучшим инструментом для создания и восстановления резервных копий в TOPAZ MariaDB, и обычно она разрешена веб-хостинговыми компаниями. Для резервного копирования таблицы клиентов с помощью mysqldump, нужно ввести следующее из командной строки:

```
mysqldump --user="username" --password="password" --add-locks db1 clients > clients.sql
```

В данной команде, имя пользователя и пароль указаны в первой строке. В следующей строке опция --add-locks используется для блокировки таблицы перед резервным копированием и для ее автоматического разблокирования после завершения резервного копирования. В третьей строке выше указано имя базы данных, за которым следует имя таблицы. Перенаправление (т.е.>) указывает оболочке отправить результаты дампа в текстовый файл с именемклиенты.sqlв текущем каталоге. Путь к каталогу может быть указан перед именем файла, чтобы создать файл в другом месте. Если таблицу необходимо восстановить, из оболочки можно запустить следующее:

```
mysql --user="username" --password="password" db1 < clients.sql
```

3.5.2 Renaming & Shifting Tables

Чтобы изменить название таблицы clients на client_addresses следует ввести следующую команду:

```
RENAME TABLE clients
TO client_addresses;
```

Оператор RENAME TABLE также позволяет переместить таблицу в другую базу данных, просто добавив имя принимающей базы данных перед именем новой таблицы, разделенное точкой. Чтобы переместить вновь названную таблицу client_addresses в базу данных db2 следует ввести:

```
RENAME TABLE client_addresses
TO db2.client_addresses;
```

3.6 Импорт данных в TOPAZ MariaDB

3.6.1 Загрузка данных

Оператор LOAD DATA INFILEпредназначен для импорта данных из обычного текстового файла в TOPAZ MariaDB. Ниже пример воода данных в поле клиента mysql для загрузки данных из файла с именем проспекты.txt в таблицу prospect_contact :

```
LOAD DATA INFILE "/tmp/prospects.txt"
INTO TABLE prospect_contact
FIELDS TERMINATED BY "|";
```

Перед вводом приведенного выше оператора сеанс TOPAZ MariaDB, конечно же, будет переключен на базу данных sales_dept с USE утверждение. Однако можно указать базу данных вместе с именем таблицы (например,sales_dept.prospect_contact). Например, предположим, что строки в текстовом файле начинаются с двойной кавычки и заканчиваются двойной кавычкой и жестким возвратом Windows (т. е. возвратом и переводом строки).Пример утверждения:

```
LOAD DATA INFILE "/tmp/prospects.txt"
INTO TABLE prospect_contact
FIELDS TERMINATED BY "|"
LINES STARTING BY ""
TERMINATED BY ""\r\n";
```

3.6.2 Дублирование строки

Если таблица prospect_contact уже содержит некоторые записи, которые будут импортированы из проспектов.txt, то следует принять решение о том, что делать с дубликатами в TOPAZ MariaDB. Оператор SQL, как указано выше, заставит TOPAZ MariaDB попытаться импортировать дубликаты записей и создать для них дубликаты строк в prospect_contact. Если свойства таблицы настроены так, чтобы не допускать дубликатов, то TOPAZ MariaDB выдаст ошибку. Чтобы заставить TOPAZ MariaDB заменить повторяющиеся существующие строки теми, которые импортируются, нужно добавить REPLACE перед INTO TABLE:

```
LOAD DATA INFILE "/tmp/prospects.txt"
REPLACE INTO TABLE prospect_contact
FIELDS TERMINATED BY "|"
LINES STARTING BY ""
TERMINATED BY ""\n";
```

3.7 Создание резервных копий с помощью mysqldump

Утилитой для создания резервной копии данных сервера TOPAZ MariaDB является mysqldump . Он извлекает данные и схемы из каждой базы данных и

таблицы и создает текстовый файл данных вне TOPAZ MariaDB. Этот текстовый файл данных (известный как файл дампа) будет содержать операторы SQL, необходимые для восстановления баз данных и данных.

3.7.1 Резервное копирование всего

Чтобы экспортировать все базы данных в TOPAZ MariaDB, используя mysqldump , из командной строки файловой системы следует ввести следующее:

```
mysqldump -u admin_backup -p -x -A > /data/backup/dbs.sql
```

Первый набор опций (--u *admin_backup* -p) сообщит TOPAZ MariaDB, что эта утилита должна выполняться пользователем *admin_backup* и что у пользователя должен быть запрошен пароль, который нужно будет ввести в следующей строке при запросе. Если дамп должен выполняться cron посредством сценария оболочки, этот параметр можно изменить на--p*mypwd*, где *mypwd* является паролем — между-p и паролем. Опция -x позволяет TOPAZ MariaDB заблокировать все таблицы перед выполнением резервного копирования. Блокировка не будет снята, пока процесс не будет завершен. Чтобы связать операторы INSERT для каждой таблицы следует добавить опцию--e. Этот расширенный параметр вставки уменьшит размер файла дампа и позволит быстрее выполнять любые возможные будущие восстановления. Опция -A указывает, что все базы данных должны быть экспортированы. Знак «больше» — это перенаправление оболочки стандартного вывода (STDOUT) на путь и файл, названный в его честь.

Пример, приведенный для резервного копирования всей базы данных, является сокращенной версией. Приведенное выше также можно ввести следующим образом, расписав опции полностью:

```
mysqldump --user=admin_backup --password --lock-tables --all-databases > /data/backup/dbs.sql
```

3.7.2 Одна база данных

Резервное копирование всех баз данных одновременно с помощью mysqldump может привести к созданию одного большого файла дампа. Это может занять больше времени для завершения резервного копирования и сделать восстановление позже немного громоздким. Поэтому может быть полезнее разносить резервные копии на основе баз данных, создавая несколько файлов меньшего размера. Вы также можете создавать резервные копии критически важных баз данных или тех, которые сильно изменяются в более медленное время дня, чтобы не мешать взаимодействию с пользователем.

Чтобы экспортировать только одну базу данных, а не все, следует ввести следующее:

```
mysqldump --user=admin_backup --password --lock-tables --databases db1 > /data/backup/db1.sql
```

Существенная разница в этой строке заключается в том, что опция-A заменена на-B и была указана база данных для экспорта. Чтобы экспортировать несколько баз данных, просто введите их после опции -B, разделив пробелами (например, -B db1 db2).

3.7.3 Резервное копирование таблиц

Для больших баз данных может потребоваться резервное копирование данных на

основе таблиц, а не всей базы данных. Чтобы сделать резервную копию только одной таблицы, из командной строки можно ввести следующее:

```
mysqldump --user=admin_backup --password --lock-tables db1 table1 > /data/backup/db1_table1.sql
```

Первое замечание, что --databases опция не была включена в строку выше. Утилита предполагает, что первое заданное имя является базой данных, а второе имя является именем таблицы, а не другой базы данных. Чтобы создать резервную копию нескольких таблиц из базы данных следует перечислить их после имени базы данных, разделенных пробелами (например, db1 table1 table2).

3.8 Строковые функции TOPAZ MariaDB

В TOPAZ MariaDB есть множество встроенных функций, которые можно использовать для управления строками данных.

3.8.1 Форматирование

Есть несколько строковых функций, которые используются для форматирования текста и чисел для более удобного отображения. Полезной функцией для объединения содержимого полей данных с текстом является функция CONCAT(). В качестве примера предположим, что в таблице с именем contacts есть столбец для имени каждого контактного лица по продажам и столбец для фамилии. Следующая инструкция SQL объединит их:

```
SELECT CONCAT(name_first, ' ', name_last)
AS Name
FROM contacts;
```

Этот оператор отобразит имя, пробел, а затем фамилию вместе в одном столбце. Предложение AS изменит заголовок столбца результатов на Имя.

Другая конкатенирующая функция CONCAT_WS(). Он объединит столбцы с разделителем между ними. Это может быть полезно при предоставлении данных другим программам. Предположим, есть программа, которая импортирует данные, но требует, чтобы поля были разделены вертикальными чертами. Можно экспортировать данные или использовать оператор SELECT, подобный следующему, в сочетании с интерфейсом, написанным на языке API, таком как Perl:

```
SELECT CONCAT_WS("|", col1, col2, col3)
FROM table1;
```

Первый элемент выше — это разделитель. Остальные элементы — это колонки, которые нужно связать вместе.

Если требуется отформатировать длинное число с запятыми через каждые три цифры и точкой для десятичной точки (например, 100 000,00), можно использовать функцию FORMAT():

```
SELECT CONCAT("$", FORMAT(col5, 2))
FROM table3;
```

3.8.2 Извлечение данных

В случае, если необходимо извлечь определенные элементы из столбца, в TOPAZ MariaDB есть несколько функций, которые могут помочь. Предположим, столбец в таблице contacts содержит номера телефонов контактных лиц отдела продаж, включая коды городов, но без дефисов и круглых скобок. Код города каждого из них может быть извлечен для сортировки с помощью LEFT() и номер телефона с RIGHT().

```
SELECT LEFT(telephone, 3) AS area_code,
RIGHT(telephone, 7) AS tel_nbr
FROM contacts
ORDER BY area_code;
```

в LEFT() выше, в колонке telephone указывается вместе с количеством символов для извлечения, начиная с первого символа слева в колонке. RIGHT() аналогична, но она начинается с последнего символа справа, считая влево, чтобы захватить в этом операторе последние семь символов. В операторе SQL, приведенном выше, area_code повторно используется для упорядочения набора результатов. Для переформатирования номера телефона следует использовать функцию SUBSTRING().

```
SELECT CONCAT("(", LEFT(telephone, 3), ") ",
SUBSTRING(telephone, 4, 3), "-",
MID(telephone, 7)) AS "Telephone Number"
FROM contacts
ORDER BY LEFT(telephone, 3);
```

3.8.3 Изменение текта

В TOPAZ MariaDB есть несколько функций, которые могут помочь в работе с текстом. Одной из таких функций является REPLACE(). С его помощью каждое вхождение параметра поиска в строке может быть заменено. Например, чтобы заменить *Mrs.* на *Ms.* в столбце, содержащем титул человека, но только в выводе, следует ввести:

```
SELECT CONCAT(REPLACE(title, "Mrs.", "Ms."),
" ", name_first, " ", name_last) AS Name
FROM contacts;
```

Если нужно вставить или заменить определенный текст из столбца (но не все его содержимое), можно использовать INSERT()функционировать совместно с LOCATE()функция. Например, если в другой таблице контактов есть заголовок и полное имя контакта в одном столбце. Чтобы изменить вхождения Mrs. на Ms., нельзя использовать REPLACE()так как заголовок встроен в этот пример. Вместо этого следует ввести следующее:

```
SELECT INSERT(name, LOCATE(name, "Mrs."), 4, "Ms.")
FROM contacts;
```

3.9 Восстановление данных из файлов дампа

Если данные TOPAZ MariaDB были потеряны, но в ходе работы использовалось утилита mysqldump, то можно использовать файлы дампа для восстановления ваших данных. В конце концов, в этом суть резервных копий. Чтобы восстановить файл дампа, достаточно, чтобы клиент mysql выполнил все операторы SQL, содержащиеся в файле. Возможный способ восстановления из файла дампа — ввести следующее:

```
mysql --user admin_restore --password < /data/backup/db1.sql
```

3.9.1 Восстановление таблицы

Простым и легким методом ограничения восстановления было бы временное

```
GRANT SELECT
ON db1.* TO "admin_restore_temp"@"localhost"
IDENTIFIED BY "its_pwd";

GRANT ALL ON db1.table1
TO      "admin_restore_temp"@"localhost";
```

создание пользователя, имеющего права только для таблицы, которую вы требуется восстановить. Для этого следует использовать оператор GRANT:

3.10 Подключение к TOPAZ MariaDB

Чтобы подключиться к серверу TOPAZ MariaDB, клиентское программное обеспечение должно предоставить правильные параметры подключения. Клиентское программное обеспечение чаще всего клиент mysql , используется для ввода операторов из командной строки, но те же принципы применимы к любому клиенту.

Если параметр подключения не указан, он вернется к значению по умолчанию.

3.10.1 Параметры подключения

host

```
--host=name
-h name
```

Подключитесь к серверу TOPAZ MariaDB на данном хосте. Хост по умолчанию localhost.

password

```
--password[=passwd]
-p[passwd]
```

Пароль учетной записи TOPAZ MariaDB. Если вы использовали -p или --password, но не введете пароль, вам будет предложено ввести его.

pipe

```
--pipe
-W
```

В системах Windows, запущенных с опцией --enable-named-pipe, используйте этот параметр для подключения к серверу с использованием именованного канала.

port

```
--port=num
-P num
```

Номер порта TCP/IP, используемый для соединения. По умолчанию 3306.

protocol

```
--protocol=name
```

Указывает протокол, который будет использоваться для соединения для соединения. Протокол может быть один из следующих: TCP , SOCKET , PIPE или MEMORY (без учета регистра). Как правило это значение следует оставить по умолчанию..

- TCP: TCP/IP-соединение с сервером (локальным или удаленным). Доступно во всех операционных системах.
- SOCKET: соединение с файлом сокета Unix, доступное для локального сервера только в системах Unix.
- PIPE. Соединение по именованному каналу (локальное или удаленное). Доступно только в Windows.
- MEMORY. Соединение с общей памятью с локальным сервером только в системах Windows.

shared-memory-base-name

```
--shared-memory-base-name=name
```

Доступно только в системах Windows, в которых сервер был запущен с опцией--shared-memory, данная опция указывает имя общей памяти, используемое для подключения к локальному серверу. Значение чувствительно к регистру и по умолчанию равно MySQL.

socket

```
--socket=name
-S name
```

Для соединений с локальным хостом либо используемый файл сокета Unix (по умолчанию/tmp/ mysql .sock) или в Windows, где сервер был запущен с опцией--enable-named-pipe, имя (без учета регистра) именованного канала для использования (по умолчанию MySQL).

user

```
--user=name
-u name
```

Имя пользователя TOPAZ MariaDB для использования при подключении к серверу. По умолчанию используется либо ваше имя пользователя Unix, либо ODBC в Windows. См. GRANT Команда для получения подробной информации о создании учетных записей пользователей TOPAZ MariaDB.

3.11 Конфигурирование TOPAZ MariaDB для удалённого доступа клиентов

Некоторые пакеты TOPAZ MariaDB привязывают TOPAZ MariaDB к 127.0.0.1 (IP-адрес с обратной связью) по умолчанию в качестве меры безопасности, используя директиву конфигурации bind-address. Старые пакеты MySQL иногда полностью отключали сеть TCP/IP, используя директиву skip-networking. Прежде чем перейти к их настройке, ознакомьтесь с информацией, что делает каждый из них:

- skip-networking. Сообщает TOPAZ MariaDB о запуске без каких-либо сетевых опций TCP / IP.
- bind-address требуется для понимания дополнительной информации. Данный сервер обычно имеет два или более сетевых интерфейса. Два наиболее распространённых их них – это сетевое устройство Loopback и Network Interface Card (NIC), которые позволяют взаимодействовать с сетью. TOPAZ MariaDB привязан к интерфейсу обратной связи по умолчанию, поскольку это делает невозможным подключение к TCP-порту на сервере с удаленного хоста (адрес привязки должен относиться к локальному IP-адресу, иначе выйдет фатальная ошибка, и TOPAZ MariaDB не запустится). Это нежелательно, если Вы хотите использовать TCP-порт с удаленного хоста, поэтому необходимо удалить эту директиву bind-address (TOPAZ MariaDB поддерживает только один bind-адрес, но привязывается к 0.0.0.0 или :: (каждый IP), если директива bind-address не указана).

Если bind-адрес привязан к 127.0.0.1 (localhost), невозможно подключиться к серверу TOPAZ MariaDB с других хостов или с того же хоста по протоколу TCP/IP через интерфейс, отличный от loopback (127.0.0.1). Это, например, не сработает (подключение с именем хоста, которое указывает на локальный IP-адрес хоста):

```
(/my/maria-10.4) ./client/mysql --host=myhost --protocol=tcp --port=3306 test
ERROR 2002 (HY000): Can't connect to MySQL server on 'myhost' (115)
(/my/maria-10.4) telnet myhost 3306
Trying 192.168.0.11...
telnet: connect to address 192.168.0.11: Connection refused
```

Использование 'localhost' работает при соединении с bind_address:

```
(my/maria-10.4) ./client/mysql --host=localhost --protocol=tcp --port=3306 test
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor.  Commands end with ; or \g.
...
```

Поиск файла DEFAULTS

Чтобы включить TOPAZ MariaDB для прослушивания удаленных подключений, вам необходимо отредактировать свой файл по умолчанию.

Типовые расположения файлов по умолчанию:

```
  * /etc/my.cnf                          (*nix/BSD)
  * $MYSQL_HOME/my.cnf                   (*nix/BSD) *Most Notably /etc/mysql/my.cnf
  * SYSCONFDIR/my.cnf                    (*nix/BSD)
  * DATADIR\my.ini                       (Windows)
```

Вы можете увидеть, какие файлы DEFAULTS считываются и в каком порядке, выполнив:

```
shell> mysqld --help --verbose
./sql/mysqld  Ver 10.4.2-MariaDB-valgrind-max-debug for Linux on x86_64 (Source distribution)
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Starts the MariaDB database server.

Usage: ./sql/mysqld [OPTIONS]

Default options are read from the following files in the given order:
/etc/my.cnf /etc/mysql/my.cnf ~/.my.cnf
```

Последняя строка показывает, какие файлы DEFAULTS считываются.

3.11.1 Редактирование файлов DEFAULTS

Как только вы найдете файл defaults, откройте файл с помощью текстового редактора и попробуйте найти строки, подобные этой, в разделе [mysqld].

```
[mysqld]
   ...
   skip-networking
   ...
   bind-address = <some ip-address>
   ...
```

(Строки могут располагаться не в таком порядке, и порядок не имеет значения).

Если вы можете найти эти строки, убедитесь, что они обе имеют комментарии (предваряются символами хэша (#)):

```
[mysqld]
   ...
   #skip-networking
   ...
   #bind-address = <some ip-address>
   ...
```

В качестве альтернативы, просто добавьте следующие строки **в конец** Вашего файла .my.cnf (обратите внимание, что имя файла начинается с точки) в вашем домашнем каталоге или альтернативно **последней** в вашем файле /etc/my.cnf.

```
[mysqld]
skip-networking=0
skip-bind-address
```

Это работает, поскольку можно иметь любое количество разделов mysqld.

Сохраните файл и перезапустите mysqld демон или службу (см. раздел Запуск и остановка

TOPAZ MariaDB).

Вы можете проверить параметры, используемые mysqld, выполнив:

```
shell> ./sql/mysqld --print-defaults
./sql/mysqld would have been started with the following arguments:
--bind-address=127.0.0.1 --innodb_file_per_table=ON --server-id=1 --skip-bind-address ...
```

Не имеет значения, остался ли у вас исходный --bind-адрес, так как более поздний --skip-bind-адрес перезапишет его.

3.11.2 Предоставление пользовательских соединений с удалённых хостов

После установки сервера TOPAZ MariaDB настроена на прием подключений с удаленных хостов необходимо добавить пользователя, которому разрешено подключаться с чего-то другого, кроме 'localhost' (пользователи в TOPAZ MariaDB определяются как 'user'@'host', поэтому 'chadmaynard' @'localhost' и 'chadmaynard'@'1.1.1.1' (или 'chadmaynard'@'server.domain.local') - это разные пользователи, которые могут иметь совершенно разные разрешения и/или пароли.

Создание нового пользователя:

- войдите в mysql command line client (или предпочитаемый Вами графический клиент)

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 36
Server version: 5.5.28-MariaDB-mariadb1~lucid mariadb.org binary distribution

Copyright (c) 2000, 2012, Oracle, Monty Program Ab and others.

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

MariaDB [(none)]>
```

- для просмотра любых существующих удаленных пользователей, выполните следующую инструкцию SQL в таблице mysql.user

```
SELECT User, Host FROM mysql.user WHERE Host <> "localhost";
+-------------+------------------+
| User    | Host       |
+-------------+------------------+
| daniel | %          |
| root    | 127.0.0.1 |
| root    | ::1          |
| root    | gandalf    |
+-------------+------------------+
4 rows in set (0.00 sec)
```

Необходимо сделать несколько решений на основе следующих факторов:

- список допустимых привилегий
- к какой базе данных/таблицам применяются эти привилегии
- имя пользователя
- хост, с которого данных пользователь может подключиться
- пароль (опционально)

В примере ниже приведен процесс создания "корневого" пользователя, который может подключаться из любого места. В примере создается корневой пользователь, который может подключаться из любого места в моей локальной сети (LAN), который имеет адреса в подсети 192.168.100.0/24. Это улучшение, потому что подключение сервера TOPAZ MariaDB к Интернету и предоставление доступа ко всем хостам является не рекомендуется.

```
GRANT ALL PRIVILEGES ON *.* TO "root"@"192.168.100.%"
  IDENTIFIED BY "my-new-password" WITH GRANT OPTION;
```

(% - это wildcard)

Для получения дополнительной информации о том, как использовать GRANT, пожалуйста, смотрите страницу GRANT.

На данный момент мы достигли нашей цели, и у нас есть пользователь root, который может подключаться из любой точки локальной сети 192.168.100.0/24.

3.11.3 Порт 3306 конфигурирован в межсетевой экран

Еще один момент, который следует учитывать, настроен ли межсетевой экран на разрешение входящих запросов от удаленных клиентов:

На RHEL и CentOS 7 может потребоваться настроить межсетевой экран, чтобы разрешить TCP-доступ к MySQL с удаленных хостов. Для этого выполните обе эти команды:

```
firewall-cmd --add-port=3306/tcp
firewall-cmd --permanent --add-port=3306/tcp
```

3.11.4 Предупреждения

Если в вашей системе установлен программный межсетевой экран (или аппаратным межсетевой экран или NAT), вы должны разрешить подключения, предназначенные для TCP-порта, на котором работает MariaDB (по умолчанию и почти всегда 3306).

Чтобы отменить это изменение и больше не разрешать удаленный доступ, просто удалите строку skip-bind-address или раскомментируйте bind-address строку в вашем файле по умолчанию. Конечным результатом должно быть то, что вы должны иметь в выходных данных производных от ./sql/mysqld --print-defaults с опцией --bind-address=127.0.0.1, и без опции --skip-bind-address .

3.12 Создание и использование представлений

3.12.1 База данных сотрудников

Создание таблицы, в которой будут содержаться все сотрудники и их контактная информация:

```
CREATE TABLE `Employees` (
  `ID` TINYINT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
  `First_Name` VARCHAR(25) NOT NULL,
  `Last_Name` VARCHAR(25) NOT NULL,
  `Position` VARCHAR(25) NOT NULL,
  `Home_Address` VARCHAR(50) NOT NULL,
  `Home_Phone` VARCHAR(12) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM;
```

Добавление в таблицу сотрудников:

```
INSERT INTO `Employees` (`First_Name`, `Last_Name`, `Position`, `Home_Address`, `Home_Phone`)
VALUES
  ("Mustapha", "Mond", "Chief Executive Officer", "692 Promiscuous Plaza", "326-555-3492"),
  ("Henry", "Foster", "Store Manager", "314 Savage Circle", "326-555-3847"),
  ("Bernard", "Marx", "Cashier", "1240 Ambient Avenue", "326-555-8456"),
  ("Lenina", "Crowne", "Cashier", "281 Bumblepuppy Boulevard", "328-555-2349"),
  ("Fanny", "Crowne", "Restocker", "1023 Bokanovsky Lane", "326-555-6329"),
  ("Helmholtz", "Watson", "Janitor", "944 Soma Court", "329-555-2478");
```

Создание второй таблицы, содержащей часы, которые каждый сотрудник отработал в течение недели:

```
CREATE TABLE `Hours` (
  `ID` TINYINT(3) UNSIGNED NOT NULL,
  `Clock_In` DATETIME NOT NULL,
  `Clock_Out` DATETIME NOT NULL
) ENGINE=MyISAM;
```

Добавление полной недели часов для каждого из сотрудников во вторую таблицу:

```
INSERT INTO `Hours`
VALUES ("1", "2005-08-08 07:00:42", "2005-08-08 17:01:36"),
  ("1", "2005-08-09 07:01:34", "2005-08-09 17:10:11"),
  ("1", "2005-08-10 06:59:56", "2005-08-10 17:09:29"),
  ("1", "2005-08-11 07:00:17", "2005-08-11 17:00:47"),
  ("1", "2005-08-12 07:02:29", "2005-08-12 16:59:12"),
  ("2", "2005-08-08 07:00:25", "2005-08-08 17:03:13"),
  ("2", "2005-08-09 07:00:57", "2005-08-09 17:05:09"),
  ("2", "2005-08-10 06:58:43", "2005-08-10 16:58:24"),
  ("2", "2005-08-11 07:01:58", "2005-08-11 17:00:45"),
  ("2", "2005-08-12 07:02:12", "2005-08-12 16:58:57"),
  ("3", "2005-08-08 07:00:12", "2005-08-08 17:01:32"),
  ("3", "2005-08-09 07:01:10", "2005-08-09 17:00:26"),
  ("3", "2005-08-10 06:59:53", "2005-08-10 17:02:53"),
  ("3", "2005-08-11 07:01:15", "2005-08-11 17:04:23"),
  ("3", "2005-08-12 07:00:51", "2005-08-12 16:57:52"),
  ("4", "2005-08-08 06:54:37", "2005-08-08 17:01:23"),
  ("4", "2005-08-09 06:58:23", "2005-08-09 17:00:54"),
  ("4", "2005-08-10 06:59:14", "2005-08-10 17:00:12"),
  ("4", "2005-08-11 07:00:49", "2005-08-11 17:00:34"),
  ("4", "2005-08-12 07:01:09", "2005-08-12 16:58:29"),
  ("5", "2005-08-08 07:00:04", "2005-08-08 17:01:43"),
  ("5", "2005-08-09 07:02:12", "2005-08-09 17:02:13"),
  ("5", "2005-08-10 06:59:39", "2005-08-10 17:03:37"),
  ("5", "2005-08-11 07:01:26", "2005-08-11 17:00:03"),
  ("5", "2005-08-12 07:02:15", "2005-08-12 16:59:02"),
  ("6", "2005-08-08 07:00:12", "2005-08-08 17:01:02"),
  ("6", "2005-08-09 07:03:44", "2005-08-09 17:00:00"),
  ("6", "2005-08-10 06:54:19", "2005-08-10 17:03:31"),
  ("6", "2005-08-11 07:00:05", "2005-08-11 17:02:57"),
  ("6", "2005-08-12 07:02:07", "2005-08-12 16:58:23");
```

3.12.2 Работа с базой данных сотрудников

В этом примере мы собираемся помочь отделу кадров, упростив запросы, которые должны выполнять их приложения. В то же время это позволит нам абстрагировать их запросы от базы данных, что позволит нам более гибко ее поддерживать.

3.12.3 Фильтр по имени, дате и времени

В предыдущем руководстве был рассмотрен запрос JOIN, который отображал все случаи опоздания для конкретного сотрудника. В этом руководстве данный запрос будет абстрагирован, для предоставления всех случаев опозданий для всех сотрудников. Затем данный запрос будет стандартизирован в виде представления.

Предыдущий запрос выглядел следующим образом:

```
SELECT
  `Employees`.`First_Name`,
  `Employees`.`Last_Name`,
  `Hours`.`Clock_In`,
  `Hours`.`Clock_Out`
FROM `Employees`
INNER JOIN `Hours` ON `Employees`.`ID` = `Hours`.`ID`
WHERE  `Employees`.`First_Name`  =  "Helmholtz"
AND DATE_FORMAT(`Hours`.`Clock_In`, "%Y-%m-%d") >= "2005-08-08"
AND DATE_FORMAT(`Hours`.`Clock_In`, "%Y-%m-%d") <= "2005-08-12"
AND DATE_FORMAT(`Hours`.`Clock_In`, "%H:%i:%S") > "07:00:59";
```

Результат:

```
+------------+-----------+---------------------+---------------------+
| First_Name | Last_Name | Clock_In            | Clock_Out           |
+------------+-----------+---------------------+---------------------+
| Helmholtz  | Watson    | 2005-08-09 07:03:44 | 2005-08-09 17:00:00 |
| Helmholtz  | Watson    | 2005-08-12 07:02:07 | 2005-08-12 16:58:23 |
+------------+-----------+---------------------+---------------------+
```

3.12.4 Уточнение зарпоса

Предыдущий пример показывает все время, когда Helmholtz приходил после семи утра. Здесь видно, что Helmholtz дважды опаздывал в течение этого отчетного периода, и также можно видеть, что в обоих случаях он либо ушел точно вовремя, либо ушел рано. Чтобы исключить из отчета всех, чье время на работе было больше 10 часов и одной минуты с времени прихода до времени ухода с работы.

```
SELECT
  `Employees`.`First_Name`,
  `Employees`.`Last_Name`,
  `Hours`.`Clock_In`,
  `Hours`.`Clock_Out`,
  (TIMESTAMPDIFF(MINUTE,`Hours`.`Clock_Out`,`Hours`.`Clock_In`) + 601) as Difference
FROM `Employees`
INNER JOIN `Hours` USING (`ID`)
WHERE DATE_FORMAT(`Hours`.`Clock_In`, "%Y-%m-%d") >= "2005-08-08"
AND DATE_FORMAT(`Hours`.`Clock_In`, "%Y-%m-%d") <= "2005-08-12"
AND DATE_FORMAT(`Hours`.`Clock_In`, "%H:%i:%S") > "07:00:59"
AND TIMESTAMPDIFF(MINUTE,`Hours`.`Clock_Out`,`Hours`.`Clock_In`) > -601;
```

Это дает следующий список людей, которые нарушили нашу политику посещаемости:

```
+------------+-----------+---------------------+---------------------+------------+
| First_Name | Last_Name | Clock_In            | Clock_Out           | Difference |
+------------+-----------+---------------------+---------------------+------------+
| Mustapha   | Mond      | 2005-08-12 07:02:29 | 2005-08-12 16:59:12 |          4 |
| Henry      | Foster    | 2005-08-11 07:01:58 | 2005-08-11 17:00:45 |          2 |
| Henry      | Foster    | 2005-08-12 07:02:12 | 2005-08-12 16:58:57 |          4 |
| Bernard    | Marx      | 2005-08-09 07:01:10 | 2005-08-09 17:00:26 |          1 |
| Lenina     | Crowne    | 2005-08-12 07:01:09 | 2005-08-12 16:58:29 |          3 |
| Fanny      | Crowne    | 2005-08-11 07:01:26 | 2005-08-11 17:00:03 |          2 |
| Fanny      | Crowne    | 2005-08-12 07:02:15 | 2005-08-12 16:59:02 |          4 |
| Helmholtz  | Watson    | 2005-08-09 07:03:44 | 2005-08-09 17:00:00 |          4 |
| Helmholtz  | Watson    | 2005-08-12 07:02:07 | 2005-08-12 16:58:23 |          4 |
+------------+-----------+---------------------+---------------------+------------+
```

3.12.5 Утилита представлений

В предыдущем примере мы видим, что было несколько случаев, когда сотрудники приходили поздно и уходили рано. К сожалению, мы также можем видеть, что этот запрос становится излишне сложным. Наличие всего этого SQL в нашем приложении не только создает более сложный код приложения, но также означает, что если мы когда-либо изменим структуру этой таблицы, нам придется изменить то, что становится несколько запутанным запросом. Именно здесь представления становятся полезны.

3.12.6 Создание представлений опозданий сотрудников

Создание представления почти точно такого же, как создание инструкции SELECT, поэтому мы можем использовать нашу предыдущую инструкцию SELECT при создании нашего нового представления:

```
CREATE SQL SECURITY INVOKER VIEW Employee_Tardiness AS
SELECT
  `Employees`.`First_Name`,
  `Employees`.`Last_Name`,
  `Hours`.`Clock_In`,
  `Hours`.`Clock_Out`,
(TIMESTAMPDIFF(MINUTE,`Hours`.`Clock_Out`,`Hours`.`Clock_In`) + 601) as Difference
FROM `Employees`
INNER JOIN `Hours` USING (`ID`)
WHERE DATE_FORMAT(`Hours`.`Clock_In`, "%Y-%m-%d") >= "2005-08-08"
AND DATE_FORMAT(`Hours`.`Clock_In`, "%Y-%m-%d") <= "2005-08-12"
AND DATE_FORMAT(`Hours`.`Clock_In`, "%H:%i:%S") > "07:00:59"
AND TIMESTAMPDIFF(MINUTE,`Hours`.`Clock_Out`,`Hours`.`Clock_In`) > -601;
```

Следует обратить внимание, что первая строка запроса содержит инструкцию 'SQL SECURITY INVOKER' - это означает, что при доступе к представлению, оно выполняется с теми же привилегиями, что и у пользователя, получающего доступ к представлению. Таким образом, если кто-то, не имеющий доступа к нашей таблице сотрудников, попытается получить доступ к этому представлению, он получит сообщение об ошибке.

За исключением параметра безопасности, остальная часть запроса достаточно понятна сама по себе. Мы просто запускаем 'CREATE VIEW <view- name> AS', а затем добавляем любую допустимую инструкцию SELECT, и наше представление создается. Теперь, если мы запускаем SELECT из представления, мы увидим, что получаем те же результаты, что и раньше, с гораздо меньшим количеством SQL:

```
SELECT * FROM Employee_Tardiness;
```

```
+------------+-----------+---------------------+---------------------+------------+
| First_Name | Last_Name | Clock_In            | Clock_Out           | Difference |
+------------+-----------+---------------------+---------------------+------------+
| Mustapha   | Mond      | 2005-08-12 07:02:29 | 2005-08-12 16:59:12 |          5 |
| Henry      | Foster    | 2005-08-11 07:01:58 | 2005-08-11 17:00:45 |          3 |
| Henry      | Foster    | 2005-08-12 07:02:12 | 2005-08-12 16:58:57 |          5 |
| Bernard    | Marx      | 2005-08-09 07:01:10 | 2005-08-09 17:00:26 |          2 |
| Lenina     | Crowne    | 2005-08-12 07:01:09 | 2005-08-12 16:58:29 |          4 |
| Fanny      | Crowne    | 2005-08-09 07:02:12 | 2005-08-09 17:02:13 |          1 |
| Fanny      | Crowne    | 2005-08-11 07:01:26 | 2005-08-11 17:00:03 |          3 |
| Fanny      | Crowne    | 2005-08-12 07:02:15 | 2005-08-12 16:59:02 |          5 |
| Helmholtz  | Watson    | 2005-08-09 07:03:44 | 2005-08-09 17:00:00 |          5 |
| Helmholtz  | Watson    | 2005-08-12 07:02:07 | 2005-08-12 16:58:23 |          5 |
+------------+-----------+---------------------+---------------------+------------+
```

Теперь мы можем даже выполнять операции над таблицей, например, ограничивать наши результаты только теми, разница в которых составляет не менее пяти минут:

```
SELECT * FROM Employee_Tardiness WHERE Difference >=5;
```

```
+------------+-----------+---------------------+---------------------+------------+
| First_Name | Last_Name | Clock_In            | Clock_Out           | Difference |
+------------+-----------+---------------------+---------------------+------------+
| Mustapha   | Mond      | 2005-08-12 07:02:29 | 2005-08-12 16:59:12 |          5 |
| Henry      | Foster    | 2005-08-12 07:02:12 | 2005-08-12 16:58:57 |          5 |
| Fanny      | Crowne    | 2005-08-12 07:02:15 | 2005-08-12 16:59:02 |          5 |
| Helmholtz  | Watson    | 2005-08-09 07:03:44 | 2005-08-09 17:00:00 |          5 |
| Helmholtz  | Watson    | 2005-08-12 07:02:07 | 2005-08-12 16:58:23 |          5 |
+------------+-----------+---------------------+---------------------+------------+
```

3.12.7 Другие применения представлений

Помимо простого упрощения SQL-запросов нашего приложения, представления

могут предоставить и другие преимущества, некоторые из которых возможны только при их использовании.

3.12.8 Ограничение доступа к данным

Например, несмотря на то, что наша база данных сотрудников содержит поля для должности, домашнего адреса и домашнего телефона, наш запрос не позволяет отображать эти поля. Это означает, что в случае проблем с безопасностью в приложении (например, атаки с использованием внедрения SQL кода или даже с участием программиста-злоумышленника) нет риска раскрытия личной информации сотрудника.

3.12.9 Безопасность уровня строк

Также можно определить отдельные представления, чтобы включить конкретный раздел WHERE для обеспечения безопасности; например, если мы хотим ограничить доступ руководителя отдела только к сотрудникам, которые ему подчиняются, мы могли бы указать его личность в инструкции представления CREATE, и тогда он не сможет видеть сотрудников никаких других отделов. Если это представление доступно для записи, и оно определено с помощью раздела CASCADE, это ограничение также будет применяться к записям. На самом деле это единственный способ реализовать безопасность на уровне строк в MySQL, поэтому представления также играют важную роль в этой области.

3.12.10 Упреждающая оптимизация

Также можно определить наши представления таким образом, чтобы принудительно использовать индексы, чтобы другие, менее опытные разработчики не рисковали запускать неоптимизированные запросы или операторы JOIN, которые приводят к сканированию всей таблицы и расширенным блокировкам. Дорогостоящие запросы, запросы с оператором SELECT* и плохо продуманные операторы JOIN могут не только полностью замедлить работу базы данных, но и привести к сбою INSERT, истечению подключения клиентов и ошибкам в отчетах. Создавая представление, которое уже оптимизировано, и позволяя пользователям выполнять свои запросы к нему, вы можете гарантировать, что они не приведут к значительному снижению производительности без необходимости.

3.12.11 Сводные таблицы

Когда мы проводим реинжиниринг нашего приложения, нам иногда приходится изменять базу данных для оптимизации или размещения новых или удаленных функций.

Используя представления можно изменить формат базовой таблицы, сохраняя при этом тот же формат таблицы для устаревшего приложения. Таким образом, приложение, которому требуются имя пользователя, имя хоста и время доступа в строковом формате, может получить доступ к тем же данным, что и приложение, которому требуются имя, фамилия, user@host и время доступа в формате временной метки Unix.

3.13 Объединение таблиц с помощью JOIN

Ниже приведен пример использования трёх базовых типов JOIN, с которыми возможно экспериментировать для того чтоб увидеть реализацию различных JOIN:

```
CREATE TABLE t1 ( a INT );
CREATE TABLE t2 ( b INT );
INSERT INTO t1 VALUES (1), (2), (3);
INSERT INTO t2 VALUES (2), (4);
SELECT * FROM t1 INNER JOIN t2 ON t1.a = t2.b;
SELECT * FROM t1 CROSS JOIN t2;
SELECT * FROM t1 LEFT JOIN t2 ON t1.a = t2.b;
SELECT * FROM t2 LEFT JOIN t1 ON t1.a = t2.b;
```

Первые два SELECT зачастую пишутся с устаревшей формой:

```
SELECT * FROM t1, t2 WHERE t1.a = t2.b;
SELECT * FROM t1, t2;
```

Из этого Вы можете видеть, что **INNER JOIN** создает результирующий набор, содержащий только строки, которые имеют совпадение в обеих таблицах (t1 и t2) для указанного join условия(ий).

Процедуры **CROSS JOIN** создают результирующий набор, в котором каждая строка в каждой таблице соединяется с каждой строкой в другой таблице; это также называется **декартовым произведением**. В TOPAZ MariaDB ключевое слово CROSS можно опустить, так как оно ничего не делает. Любое JOIN без раздела ON является CROSS JOIN.

LEFT JOIN - это внешнее соединение, которое создает результирующий набор со всеми строками из таблицы "слева" (t1); значения для столбцов в другой таблице (t2) зависят от того, было ли найдено совпадение или нет. Если совпадение не найдено, всем столбцам из этой таблицы присваивается значение 0 для этой строки.

RIGHT JOIN аналогично **LEFT JOIN**, хотя его результирующий набор содержит все строки из правой таблицы, а столбцы левой таблицы при необходимости будут заполнены нулями.

Операторы JOIN могут быть объединены для считывания результатов

из трех или более таблиц.

Вот результирующие данные различных операторов SELECT,

перечисленных выше:

```
SELECT * FROM t1 INNER JOIN t2 ON t1.a = t2.b;
------ ------
| a    | b    |
------ ------
|    2 |    2 |
------ ------
1 row in set (0.00 sec)


SELECT * FROM t1 CROSS JOIN t2;
------ ------
| a    | b    |
------ ------
|    1 |    2 |
|    2 |    2 |
|    3 |    2 |
|    1 |    4 |
|    2 |    4 |
|    3 |    4 |
------ ------
6 rows in set (0.00 sec)


SELECT * FROM t1 LEFT JOIN t2 ON t1.a = t2.b;
------ ------
| a    | b    |
------ ------
|    1 | NULL |
|    2 |    2 |
|    3 | NULL |
------ ------
3 rows in set (0.00 sec)


SELECT * FROM t2 LEFT JOIN t1 ON t1.a = t2.b;
------ ------
| b    | a    |
------ ------
|    2 |    2 |
|    4 | NULL |
------ ------
2 rows in set (0.00 sec)
```

3.14 Работа с индексами

Наример, имеется таблица со следующими строками.

```
+------+------------+-----------+---------------------------+-----------------------------------+--------------+
| ID | First_Name | Last_Name | Position                  | Home_Address                      | Home_Phone   |
+------+------------+-----------+---------------------------+-----------------------------------+--------------+
|  1 | Mustapha   | Mond      | Chief Executive Officer | 692 Promiscuous Plaza             | 326-555-3492 |
|  2 | Henry      | Foster    | Store Manager             | 314 Savage Circle                 | 326-555-3847 |
|  3 | Bernard    | Marx      | Cashier                   | 1240 Ambient Avenue               | 326-555-8456 |
|  4 | Lenina     | Crowne    | Cashier                   | 281 Bumblepuppy Boulevard | 328-555-2349 |
|  5 | Fanny      | Crowne    | Restocker                 | 1023 Bokanovsky Lane              | 326-555-6329 |
|  6 | Helmholtz  | Watson    | Janitor                   | 944 Soma Court                    | 329-555-2478 |
+------+------------+-----------+---------------------------+-----------------------------------+--------------+
```

Теперь представьте, что вас попросили вернуть домашний телефон Fanny Crowne. Без индексов единственный способ сделать это - просмотреть каждую строку, пока не найдете совпадающие имя и фамилию. Теперь представьте, что есть миллионы записей, и Вы можете видеть, что даже для быстрого сервера базы данных это крайне неэффективно.

Ответ заключается в сортировке записей. Если бы они были сохранены в алфавитном порядке по фамилиям, даже человек мог бы быстро найти запись среди большого количества. Но мы не можем отсортировать всю запись по фамилии. Что, если мы хотим также просмотреть запись по идентификатору или по имени? Ответ заключается в создании отдельных индексов для каждого столбца, по которому мы хотим выполнить сортировку. Индекс просто содержит отсортированные данные (например, фамилию) и ссылку на исходную запись.

Например, индекс по фамилии (Last_Name):

```
+-------------------+------+
| Last_Name | ID |
+-------------------+------+
| Crowne    |  4 |
| Crowne    |  5 |
| Foster    |  2 |
| Marx      |  3 |
| Mond      |  1 |
| Watson    |  6 |
+-------------------+------+
```

И индекс по должности (Position)

```
+---------------------------------------------+------+
| Position                  | ID |
+---------------------------------------------+------+
| Cashier                   |  3 |
| Cashier                   |  4 |
| Chief Executive Officer |  1 |
| Janitor                   |  6 |
| Restocker                 |  5 |
| Store Manager             |  2 |
+---------------------------------------------+------+
```

позволяет быстро найти номера телефонов всех кассиров или номер телефона сотрудника с фамилией Маркс.

Там, где это возможно, Вам следует создать индекс для каждого столбца, по которому вы выполняете поиск записей, чтобы сервер не считывал каждую строку таблицы.

3.15 Устранение неполадок с подключением

Существует ряд распространенных проблем, которые могут возникнуть при

подключении к TOPAZ MariaDB.

3.15.1 Сервер не запускается в определённом местоположении

Если получаемая ошибка представляет собой что-то вроде:

```
mysql -uname -p
ERROR 2002 (HY000): Can't connect to local MySQL server through
  socket "/var/run/mysqld/mysqld.sock" (2 "No such file or directory")
```

или

```
mysql -uname -p --port=3307 --protocol=tcp
ERROR 2003 (HY000): Can't connect to MySQL server on  "localhost"
  (111 "Connection refused")
```

сервер либо не запущен, либо не работает на указанном порту, сокете или канале. Убедитесь, что вы используете правильные параметры хоста, порта, канала, сокета и протокола.

Файл сокета может находиться по нестандартному пути. В этом случае параметр сокета, вероятно, записан в файле my.cnf. Убедитесь, что его значение идентично в разделах [mysqld] и [client]; если нет, клиент будет искать сокет в неправильном месте.

Если вы не уверены, где запущен файл сокета Unix, это можно выяснить, например:

```
netstat -ln | grep mysqld
unix  2      [ ACC ]     STREAM     LISTENING     33209505 /var/run/mysqld/mysqld.sock
```

3.15.2 Не удается подключиться из удаленного местоположения

Обычно сервер TOPAZ MariaDB по умолчанию не принимает подключения от удаленного клиента или подключение по протоколу tcp и имени хоста и должен быть настроен для их разрешения.

```
(/my/maria-10.4) ./client/mysql --host=myhost --protocol=tcp --port=3306 test
ERROR 2002 (HY000): Can't connect to MySQL server on 'myhost' (115)
(/my/maria-10.4) telnet myhost 3306
Trying 192.168.0.11...
telnet: connect to address 192.168.0.11: Connection refused
(/my/maria-10.4) perror 115
OS error code 115:  Operation now in progress
```

Чтобы решить эту проблему, см. раздел Конфигурирование TOPAZ MariaDB для удаленного доступа клиентов

3.15.3 Проблемы с аутентификацией

В TOPAZ MariaDB плагин аутентификации unix_socket включен по умолчанию в Unix-подобных системах. При этом используются учетные данные операционной системы при подключении к TOPAZ MariaDB через локальный файл сокета Unix.

Аутентификация предоставляется определенной комбинации имени пользователя и хоста. user1'@'localhost' , например, не совпадает с user1'@'166.78.144.191' . Смотрите статью Предоставление разрешений для получения подробной информации о предоставлении разрешений.

Пароли хэшируются с помощью функции PASSWORD. Если вы установили пароль с помощью инструкции SET PASSWORD или использовали INSERT или UPDATE

для непосредственного обновления таблицы разрешений, функция PASSWORD должна использоваться одновременно. Например, пишите SET PASSWORD FOR 'bob'@'%.loc.gov' = PASSWORD('newpass') вместо того, чтобы писать SET PASSWORD FOR 'bob'@'%.loc.gov' = 'newpass' ;

Если таблицы предоставления были изменены напрямую, новые пароли или данные аутентификации не будут немедленно активированы. Для того, чтобы изменения вступили в силу, необходимо выполнить инструкцию FLUSH PRIVILEGES или опцию flush-privileges mysqladmin.

3.15.4 Проблемы с экспортом результатов запроса

Если вы можете выполнять обычные запросы, но получаете ошибку аутентификации при запуске операторов SELECT ... INTO OUTFILE, SELECT ... INTO DUMPFILE или LOAD DATA INFILE, то у вас нет разрешения на запись файлов на сервер. Для этого требуются права доступа к FILE.

3.15.5 Доступ к серверу, но не к базе данных

Если вы можете подключиться к серверу, но не к базе данных, например:

```
USE test;
ERROR 1044 (42000): Access denied for user "ian"@"localhost" to database "test"
```

или можете подключиться к определенной базе данных, но не к другой. Например, mysql -u name db1 работает, но mysql -u name db2 не запускается, вам не было предоставлено разрешение для конкретной базы данных.

3.15.6 Файлы опций и переменные среды

Возможно, что файлы опций или переменные среды могут указывать неправильные параметры подключения. Проверьте значения, указанные в любых файлах опций, считываемых клиентом, который Вы используете (см. Конфигурационные файлы и группы mysqld и документацию для конкретного используемого вами клиента - см. Клиенты и утилиты).

Файлы опций обычно могут быть скрыты с помощью функции no-defaults, например:

```
mysqlimport --no-defaults ...
```

3.15.7 Не удаётся подключиться к работающему серверу / Утерян пароль root

Если вы не можете подключиться к серверу, например, из-за потери пароля root, вы можете запустить сервер без использования таблиц привилегий, выполнив опцию --skip-grant-tables, которая предоставляет пользователям полный доступ ко всем таблицам. Затем вы можете запустить СБРОС ПРИВИЛЕГИЙ, чтобы возобновить использование таблиц предоставления, а затем УСТАНОВИТЬ ПАРОЛЬ, чтобы изменить пароль для учетной записи.

3.15.8 localhost и %

Возможно, вы создали пользователя следующим образом:

```
CREATE USER melisa identified by "password";
```

Это создает пользователя с подстановочным знаком '%'.

```
select user,host from mysql.user where user="melisa";
+---------+------+
| user    | host |
+---------+------+
| melisa | %    |
+---------+------+
```

Однако, возможно, вам все еще не удается войти в систему с localhost. Некоторые настройки создают анонимных пользователей, включая localhost. Итак, в пользовательской таблице существуют следующие записи:

```
select user,host from mysql.user where user="melisa" or user="";
+---------+-----------+
| user    | host      |
+---------+-----------+
| melisa | %         |
|         | localhost |
+---------+-----------+
```

Поскольку вы подключаетесь с localhost, используются анонимные учетные данные, а не учетные данные пользователя 'melisa'. Решение заключается либо в добавлении нового пользователя, специфичного для localhost, либо в удалении анонимного пользователя localhost.

4 ПРОЦЕДУРЫ ТЕХНИЧЕСКОГО ОБСЛУЖИВАНИЯ СИСТЕМЫ

4.2 МЕРОПРИЯТИЯ ПО КОНТРОЛЮ

Эксплуатационный контроль Системы заключается в постоянном наблюдении (мониторинге) и периодическом анализе параметров технического состояния Системы, с отслеживанием динамики происходящих с ними изменений.

Задачи эксплуатационного контроля:

- Мониторинг работоспособности, доступности и производительности серверного оборудования Системы.
- Мониторинг работоспособности Системы, включая анализ дискового пространства и памяти, загрузку процессора и процессов и т.д.
- Мониторинг операционных систем.
- Мониторинг проблем производительности БД, входящих в состав Системы.
- Мониторинг инцидентов, связанных с Системой.
- Анализ журналов компонентов Системы.
- Обнаружение и идентификация событий, в том числе: отказов в обслуживании, сбоев в работе оборудования (перезагрузок и т.п.) и программного обеспечения Системы.
- Анализ полученных данных и событий, определение технического состояния Системы.
- Анализ событий информационной безопасности, в т.ч. по журналам событий в Системе.
- Своевременное оповещение руководства, заинтересованных служб и подразделений об аварийном состоянии Системы.

Для мониторинга используются стандартные (встроенные) средства. Проверка состояния системы осуществляется Системными администраторами.

Реализация эксплуатационного контроля предусматривает установление контролируемых значений параметров, определяющих техническое состояние компонент Системы. При достижении параметром этого значения компонент Системы считается неисправным и требующим проведения операций ТО или ремонта. В этом случае, Система должна переводиться в режим обслуживания и обновления или аварийный режим работы.

Данные, полученные в результате эксплуатационного контроля (диагностирования), являются основой для принятия решений о необходимости выполнения ТО или ремонта, времени его проведения и объеме, а также о времени проведения очередного контроля технического состояния.

Непрерывный контроль (мониторинг) параметров технического состояния Системы регламентируется политикой Компании и осуществляется с помощью инструментов мониторинга, принятых в Компании.

4.3 ПРОЦЕДУРЫ РЕГЛАМЕНТИРОВАННОГО ОБСЛУЖИВАНИЯ

К основным видам технического обслуживания (ТО) относятся:

- плановое ТО (другие отраслевые названия: профилактическое, регламентированное) – техническое обслуживание, постановка на которое осуществляется в соответствии с требованиями документации;
- внеплановое ТО (другие отраслевые названия: корректирующее, нерегламентированное) – техническое обслуживание, постановка на которое осуществляется без предварительного назначения по техническому состоянию.

На регулярной основе (ежегодно, ежеквартально) формируется и утверждается План регламентированного обслуживания Системы, включающий комплекс профилактических организационных и технических мероприятий, направленных на предупреждение инцидентов и обеспечение бесперебойной работы Системы.

Установка и учет системных обновлений и обновлений ПО выполняется в

соответствии с регламентирующими документами Компании.

5 РЕЗЕРВНОЕ КОПИРОВАНИЕ

Требования по режиму функционирования и нормативы времени восстановления Системы определяются Заказчиком.

Время простоя при выполнении плановых работ по обслуживанию и обновлению Системы согласуется с Заказчиком.

План резервного копирования определяется проектной документацией на Систему и согласуется Заказчиком на основании политик Компании.

6 ВОССТАНОВЛЕНИЕ СИСТЕМЫ

Требования по режиму восстановления Системы определяются Заказчиком.

Время простоя при выполнении работ по восстановлению Системы согласуется с Заказчиком.

План восстановления определяется проектной документацией на Систему и согласуется Заказчиком на основании политик Компании при наличии плана резервного копирования.

Рекомендуется проводить регулярное тестирование плана восстановления Системы при условии создания необходимых архивов Системы.