

Realtime конфигурация Asterisk, использование базы данных MariaDB

Используемый на текущий момент способ хранения конфигурации Asterisk в текстовых файлах имеет ряд недостатков, основной из которых - необходимость при малейшем изменении конфигурации перезагружать соответствующий модуль или вообще Asterisk в целом. В высоконагруженных средах даже субсекундный простой, вызванный этим действием, абсолютно неприемлем, не говоря о невозможности сохранить текущие звонки и очередь после перезагрузки.

Проблему динамического изменения конфигурационных параметров Asterisk решает так называемая Realtime конфигурация, при которой конфигурационные параметры хранятся в таблицах базы данных (БД) вместо текстовых файлов. Изменение значения какого-либо параметра в таблице приводит к немедленному изменению в работе Asterisk (однако не всегда хранение параметров в БД означает realtime, можно сконфигурировать статическое хранение параметров, при котором после внесения изменений в базу потребуется перезагрузить Asterisk для принятия изменений).

#Установка СУБД MariaDB

MariaDB представляет собой систему управления реляционными базами данных, является свободным ответвлением (распространяется по лицензии GNU GPL) от СУБД MySQL, при этом сохраняя обратную совместимость для API.

В официальном репозитории AppStream доступна версия MariaDB 10.3, чтобы получить новее, необходимо скачать исходный код и собрать пакет самостоятельно, либо добавить репозиторий разработчика в список репозитория для `yum` и установить пакет стандартным способом, что и будет сделано далее.

Для добавления репозитория MariaDB версии 10.5 создайте файл `/etc/yum.repos.d/MariaDB.repo` и добавьте в него следующие строки:

```
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/10.5/centos8-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
enabled=1
```

** строка `enabled` отвечает за включение/отключение репозитория, изменяя значение в файле можно включать/отключать репозиторий при необходимости.*

Установите вспомогательные пакеты и саму СУБД MariaDB (для предотвращения возникновения конфликта между репозиториями придется на время установки выключить стандартный репозиторий AppStream)

```
sudo yum install boost-program-options -y
```

```
sudo yum --disablerepo=AppStream install MariaDB-server  
MariaDB-client
```

Запуск службы СУБД (и установка автозапуска на старте)

```
sudo systemctl enable mariadb && sudo systemctl start mariadb
```

После установки рекомендуется осуществить первичную настройку параметров безопасности MariaDB. По умолчанию пароль пользователя СУБД root не задан (требуется просто нажать Enter), в следующих шагах, кроме шага, где потребуется задать новый пароль пользователю root (это пользователь СУБД, НЕ системный root), просто нажимать Enter, поскольку подходящий ответ здесь выбран по умолчанию:

```
sudo mysql_secure_installation
```

Проверка возможности локального подключения к СУБД (Пароль вводить от пользователя root СУБД, заданный на предыдущем шаге)

```
mysql -u root -p
```



```
Welcome to the MariaDB  
Your MariaDB connecti  
Server version: 10.3  
  
Copyright (c) 2000, 2  
  
Type 'help;' or '\h'  
MariaDB [(none)]>
```

Выход в bash: \q

Теоретические сведения о реляционных БД и SQL

Реляционная БД - набор данных, организованных в виде таблиц, включающих в себя столбцы (поля/атрибуты) и строки (записи/кортежи). Строка таблицы по сути - это набор данных, относящихся к хранимому в БД объекту. Таблицы БД, также называемые отношениями (relation), имеют согласно реляционной модели следующие свойства:

- Столбцы имеют определенный порядок, заданный при создании таблицы, которая может не иметь строк, но обязана иметь хотя бы один столбец.
- Таблица не может хранить две одинаковые строки
- Каждый столбец имеет уникальное в пределах таблицы имя и хранит значения одного типа (дата, строка, число и тд)
- На пересечении строки и столбца (в ячейке) таблицы может быть только атомарное значение (т.е. единичное, не группа значений)

Согласно этой же модели, любой элемент данных (ячейка) в таблице может быть однозначно идентифицирован именем столбца и значением в ячейке столбца, выбранного в качестве первичного ключа.

Первичный ключ (Primary key)

Может быть логическим (естественным - на основе поля, содержащего реальные данные объекта) или суррогатным (искусственным - на базе дополнительного поля, например, уникального ID). Рекомендуется использовать суррогатные ключи, поскольку данные объекта могут измениться, что приведет к необходимости удалять существующую запись и вносить новую, т.к. первичные ключи изменять нельзя.

Внешний ключ (Foreign key)

Внешние ключи позволяют связать строки и отдельные элементы разных таблиц.

Аспекты реляционных БД

- SQL - (язык структурированных запросов) основной интерфейс для работы с реляционными БД, используется для добавления, обновления и удаления строк данных, а также управления работой БД в целом.
- Целостность данных (через определение первичных и внешних ключей, а также ограничений (атрибутов столбца, см. в справочнике запросов SQL))
- Транзакция - последовательность операций на языке SQL, представляющая собой единую логическую задачу. Она должна быть выполнена целиком, либо не должен быть выполнен ни один из ее компонентов, независимо от других транзакций.

Требования ACID к транзакциям

- Атомарность - транзакция может выполняться только целиком, если не выполнена любая из частей - вся транзакция полностью отменяется.
- Единообразие - все данные, записываемые в БД через транзакции должны соответствовать всем правилам и ограничениям БД
- Изолированность - обеспечения согласованности и гарантия независимости каждой транзакции
- Надежность - все внесенные транзакцией изменения в БД на момент успешного завершения этой транзакции считаются постоянными.

Далее приведен краткий список некоторых команд SQL (их НЕ нужно выполнять сейчас, но они могут быть полезны при выполнении нижеследующих заданий как примеры).

Краткий справочник запросов SQL.

Отобразить все созданные БД

```
SHOW DATABASES;
```

Создать базу; таблицу с описанием столбцов (имя, тип данных, указание атрибутов (опционально))

```
CREATE DATABASE databasename;
```

```
CREATE TABLE table_name (  
    column1 datatype attribute,
```

```
column2 datatype,  
column3 datatype,  
....  
);
```

Примеры атрибутов:

PRIMARY KEY - указание столбца в качестве первичного ключа (NOT NULL присваивается автоматически)

UNIQUE - указание, что столбец должен хранить только уникальные значения

NULL и NOT NULL - указание, может ли столбец хранить значение NULL

AUTO_INCREMENT - значение столбца столбца будет автоматически увеличиваться при добавлении новой строки.

Записать в таблицу новую строку

```
INSERT INTO table_name (column1, column2) VALUES  
(value1, value2);
```

Обновить существующую(ие) строку(и), которые подпадают под условие(я)

```
UPDATE table_name SET column1 = value1, column2 = value2, ...  
WHERE condition1 AND condition2;
```

Выбрать все строки из таблицы

```
SELECT * FROM table_name;
```

Удалить базу, таблицу, строку из таблицы

```
DROP DATABASE databasename;
```

```
DROP TABLE table_name;
```

```
DELETE FROM table_name WHERE condition;
```

#продолжение практической части

Далее приведены команды для создания БД и ее конфигурации на языке SQL, который традиционно используется для управления реляционными БД. Обратите внимание, запрос SQL ВСЕГДА заканчивается символом ; . Команды SQL выполняются в интерфейсе СУБД (вход `mysql -u root -p`).

Создайте БД с именем `asteriskdb` с символьной кодировкой UTF8 по умолчанию:

```
CREATE DATABASE asteriskdb DEFAULT CHARACTER SET utf8 DEFAULT  
COLLATE utf8_general_ci;
```

На следующем шаге создайте пользователя СУБД, под которым Asterisk будет подключаться к СУБД, имя и пароль выберите самостоятельно.

```
CREATE USER 'имя_пользователя'@'localhost' IDENTIFIED BY 'пароль';
```

Выдайте права полного доступа на базу asteriskdb созданному на предыдущем шаге пользователю, обновите действующие права.

```
GRANT ALL PRIVILEGES ON asteriskdb.* TO 'имя_пользователя'@'localhost';
```

```
FLUSH PRIVILEGES;
```

Выберите базу для работы (по умолчанию после входа в СУБД база не выбрана)

```
USE asteriskdb;
```

Создайте в выбранной базе таблицы, необходимые Asterisk для хранения своих данных с помощью готовых SQL-скриптов.

```
source ~/asterisk-16.12.0/contrib/realtime/mysql/mysql_config.sql;
```

**Если СУБД возвращает ошибку, убедитесь, что путь до файла существует, проверьте, что версия установленного Asterisk (16.12.0 в примере) правильная, замените ~ на /home/USERNAME где USERNAME - имя вашего пользователя CentOS.*

Выход из СУБД: \q

#Сборка и установка коннектора ODBC

Коннектор Open Database Connectivity предоставляет API, с помощью которого сторонние сервисы (Asterisk в данном случае) могут получить доступ к БД.

Для работы коннектора требуются пакеты unixODBC unixODBC-devel, они были установлены во время выполнения предыдущей части работы среди зависимостей Asterisk.

Укажите прокси для Git:

```
git config --global http.proxy http://login:pass@proxy:port
```

Клонировать репозиторий с исходным кодом в текущую директорию и перейти в него

```
cd ~
```

```
git clone https://github.com/MariaDB/mariadb-connector-odbc.git
```

```
cd mariadb-connector-odbc
```

Выберите версию кода 3.1.5

```
git checkout 3.1.5
```

Инициализация и обновление вложенных репозиториев

```
git submodule init && git submodule update
```

Конфигурация параметров компиляции

```
cmake -G "Unix Makefiles" -DCMAKE_BUILD_TYPE=RelWithDebInfo -  
DCONC_WITH_UNIT_TESTS=Off -DWITH_OPENSSL=true -  
DCMAKE_INSTALL_PREFIX=/usr
```

Конец вывода в терминал в случае успешного завершения сборки выглядит так:

```
-- Configuring done  
-- Generating done  
-- Build files have been written to:
```

Сборка и установка

```
make
```

```
sudo make install
```

В конец файла /etc/odbcinst.ini добавьте описание нового драйвера подключения к СУБД (если уже существует с именем [MariaDB], приведите к представленному ниже виду):

```
[MariaDB]  
Description=ODBC for MariaDB  
Driver=/usr/lib64/libmaodbc.so  
Setup=/usr/lib64/libodbcmyS.so  
UsageCount=1
```

В файл /etc/odbc.ini добавьте параметры подключения к базе

```
[Asterisk-odbc]  
Description=MariaDB connection to 'asteriskdb' database  
driver=MariaDB  
server=localhost  
database=asteriskdb  
Port=3306  
Socket=/var/lib/mysql/mysql.sock  
option=3  
Charset=utf8
```

Проверка работоспособности коннектора (подключение к базе asteriskdb от имени созданного для Asterisk пользователя СУБД)

```
isql -v Asterisk-odbc имя_пользователя пароль
```

```

+-----+
| Connected! |
|           |
| sql-statement |
| help [tablename] |
| quit      |
|           |
+-----+
SQL> quit

```

В начало (перед строкой [asterisk]) файла /etc/asterisk/res_odbc.conf добавьте конфигурацию для ODBC коннектора:

```

[asteriskdb]

enabled => yes

dsn => Asterisk-odbc

username => имя_пользователя

password => пароль

pre-connect => yes

```

Перезапустите службу Asterisk

```
sudo systemctl restart asterisk
```

Проверка состояния подключения Asterisk к БД:

```
sudo asterisk -rx 'odbc show'
```

```

ODBC DSN Settings
-----

Name:    asteriskdb
DSN:     Asterisk-db
Number of active connections: 1 (out of 1)
Logging: Disabled

```

#Перенос конфигурации PJSIP в БД.

В начало файла /etc/asterisk/sorcery.conf вставьте следующие строки для перевода PJSIP в режим конфигурации Realtime:

```

[res_pjsip]

endpoint=realtime,ps_endpoints

auth=realtime,ps_auths

aor=realtime,ps_aors

domain_alias=realtime,ps_domain_aliases

contact=realtime,ps_contacts

```

```
[res_pjsip_endpoint_identifier_ip]
identify=realtime,ps_endpoint_id_ips
```

В файл /etc/asterisk/extconfig.conf добавьте после строки [settings] сведения, где Asterisk должен искать конфигурацию для PJSIP:

```
ps_endpoints => odb,asteriskdb
ps_auths => odb,asteriskdb
ps_aors => odb,asteriskdb
ps_domain_aliases => odb,asteriskdb
ps_endpoint_id_ips => odb,asteriskdb
ps_contacts => odb,asteriskdb
```

Пример

Для вставки новых строк в таблицу используется запрос INSERT, параметр INTO указывает в какую таблицу вставить (указывается полный путь в формате БД. Таблица или просто имя таблицы при условии, что БД была выбрана ранее через USE), далее в скобках следует перечисление имен столбцов через запятую, в которые нужно вставить данные (это не обязательно все столбцы таблицы), после VALUES в скобках указываются в соответствующем перечисленным в первой скобке именам столбцов порядке значения.

Далее в качестве примера приведен процесс создания клиента PJSIP с номером 101, путем заполнения значениями трех ключевых таблиц:

- ps_aors

```
INSERT INTO asteriskdb.ps_aors (id, max_contacts) VALUES (101, 1);
```

- ps_auths

```
INSERT INTO asteriskdb.ps_auths (id, auth_type, password, username) VALUES (101, 'userpass', 'secret', 101);
```

- ps_endpoints

```
INSERT INTO asteriskdb.ps_endpoints (id, transport, aors, auth, callerid, context, disallow, allow, direct_media) VALUES (101, 'udp-transport', '101', '101', 'first<101>', 'internal', 'all', 'opus,alaw', 'no');
```

Это аналог следующей конфигурации из файла pjsip.conf:

```
[101]
type=endpoint
```



```
transport=udp-transport
context=internal
disallow=all
allow=alaw,opus
callerid=first<101>
auth=101
aors=101
direct_media=no
```

```
[101]
type=auth
auth_type=userpass
password=secret
username=101
```

```
[101]
type=aor
max_contacts=1
```

Задание. Войдите в СУБД и аналогично примеру создайте записи в таблицах `ps_aors` `ps_auths` `ps_endpoints` БД `asteriskdb`, описывающие конфигурацию для всех описанных в файле `pjsip.conf` клиентов. В файле `/etc/asterisk/pjsip.conf` удалите все секции, принадлежащие клиентам, оставьте ТОЛЬКО секцию `[udp-transport]` и все параметры в ней.

Проверьте, что SIP-клиент может зарегистрироваться на Asterisk после перезапуска службы.

#Установка веб-интерфейса phpMyAdmin для MariaDB

Для удобства управления MariaDB можно использовать веб-интерфейс для MySQL (MariaDB) - phpMyAdmin, представляющий собой готовое веб-приложение, написанное на языке PHP. Может быть работать на базе LAMP (Linux, Apache, MariaDB, PHP) или LEMP (Linux, NGINX, MariaDB, PHP) стека. В рамках данной работы примем за основу LEMP. Поскольку MariaDB уже присутствует в системе, остаётся установка веб-сервера NGINX и интерпретатора PHP с некоторыми дополнительными пакетами.

Системные репозитории CentOS содержат модифицированную версию NGINX в плане расположения и содержания конфигурационных файлов. Для использования стандартной конфигурации можно установить NGINX из официального репозитория или собрать из исходного кода. Воспользуемся первым из указанных вариантов.

Создайте файл `/etc/yum.repos.d/nginx.repo` и поместите в него следующие строки:

```
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
```

Далее необходимо установить веб-сервер NGINX:

```
sudo yum install nginx
```

Основная конфигурация NGINX содержится в файле `/etc/nginx/nginx.conf`, его структура представлена ниже:

```
global options

events {
    .....
}

http{
    .....
    include /etc/nginx/conf.d/*.conf;
}
```

Сначала идут глобальные параметры (**global options**), которые задают основные параметры работы NGINX, например, от какого пользователя будет запущен процесс, а также количество одновременно запущенных процессов. В секции **events** описывается, как NGINX должен реагировать на входящие подключения, ниже расположена секция **http**, объединяющая все параметры, связанные с работой протокола HTTP. Последняя директива говорит, что дополнительно в конфигурацию секции `http` включаются все

файлы с расширением `.conf` из директории `/etc/nginx/conf.d`, эти файлы могут как содержать дополнительные параметры `http`, так и секции `server` с вложенными секциями `location`. По умолчанию существует файл `/etc/nginx/conf.d/default.conf` который ответственен за настройки сайта по умолчанию, отображение проверочной страницы NGINX после установки, его структура представлена ниже.

```
server {  
  
    .....  
  
    location ...{  
  
        .....  
    }  
  
}
```

Обычно в секции **http** находится одна или несколько секций **server**, каждая секция отвечает за отдельный домен (доменное имя сайта), в секции **server** размещаются секции **location**, каждая из которых отвечает за обработку запроса с определенным URL.

Для оптимизации производительности отредактируйте и добавьте недостающие параметры в основной файл конфигурации NGINX `/etc/nginx/nginx.conf`:

1. Глобальные параметры. Исправьте/добавьте под параметром `user`:

```
worker_processes  auto;  
  
worker_cpu_affinity  auto;
```

Эти параметры оптимизируют производительность рабочих процессов NGINX и их количество под число доступных ядер CPU.

2. Настройка обработки соединений, добавьте следующие параметры внутри блока `events` :

```
multi_accept on;  
  
use epoll;
```

Параметр `use epoll` — устанавливает оптимальный метод обработки соединений для Linux. Когда `multi_accept` выключен, рабочий процесс NGINX за один раз будет принимать только одно новое соединение. В противном случае рабочий процесс за один раз будет принимать сразу все новые соединения.

3. Параметры HTTP. Добавьте/скорректируйте следующие параметры

```
access_log      off;
server_tokens   off;
tcp_nopush      on;
tcp_nodelay     on;
reset_timedout_connection on;
gzip            on;
gzip_static     on;
gzip_min_length 1024;
gzip_comp_level 3;
```

Для ускорения работы и снижения нагрузки на диск в текущей установке можно отключить ведение лога доступа к веб-серверу через директиву `access_log`. Директива `server_tokens` при выставлении параметра в `off` предписывает NGINX не отсылать информацию о своей версии клиентам. Это может быть полезно, если вы хотите усложнить эксплуатацию ошибок и уязвимостей конкретной версии NGINX. Директивы `tcp` отвечают за ускорение работы транспортного протокола TCP, `keepalive_timeout` и `reset_timedout_connection` отвечают за таймер `keepalive` для сетевого соединения (время, в течение которого соединение остается открытым с момента последнего ответа клиента на сообщение `keepalive`) и разрыв соединений с истекшим таймером. Директивы `gzip` включают сжатие передаваемых данных, предварительное сжатие статических файлов и минимальный размер файла, подлежащего сжатию.

Сохраните файл конфигурации и проверьте его корректность:

```
sudo nginx -t
```

```
[vasya@voip-vasya ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[vasya@voip-vasya ~]$
```

Если на предыдущем шаге проверка пройдена, запустите службу NGINX с автозапуском на старте ОС:

```
sudo systemctl enable nginx && sudo systemctl start nginx
```

Откройте в фаерволе 80 порт для доступа к веб-интерфейсу.

```
sudo firewall-cmd --add-service=http --permanent
```

```
sudo firewall-cmd --reload
```

Проверьте, что по URL [http://IP CentOS](http://IP_CentOS) в браузере отображается тестовая страница Nginx.

Для работы последних версий phpMyAdmin требуется веб-сервер с поддержкой PHP 7.1 или новее; пакет PHP-FPM для NGINX, отвечающий за обработку php можно установить из репозитория Remi:

```
sudo yum install http://rpms.remirepo.net/enterprise/remi-release-8.rpm
```

Проверьте, что репозиторий Remi добавлен в список разрешенных:

```
sudo yum repolist
```

Включение модуля репозитория PHP версии 7.4 и установка PHP-FPM для Nginx и дополнительных пакетов PHP для phpMyAdmin:

```
sudo yum module enable php:remi-7.4 -y
```

```
sudo yum install php-fpm php-pear php-gettext php-bcmath php-zip php-mysqlnd php-gd php-pdo php-json php-xml php-opcache php-mbstring php-soap php-curl
```

В файле `/etc/php-fpm.d/www.conf` измените параметры пользователя `user` и `group`, под которым будет работать `php-fpm` с `apache` на `nginx` и проверьте, что взаимодействие между `php-fpm` и сторонними приложениями осуществляется через UNIX сокет:

```
listen = /run/php-fpm/www.sock;
```

```
user = nginx
```

```
group = nginx
```

В файле `/etc/php.ini` найдите, раскомментируйте и отредактируйте следующие параметры (отключение автоисправления URL)

```
cgi.fix_pathinfo=0
```

Выдайте права `php-fpm`, на директорию для хранения данных сессий.

```
sudo chown -R nginx.nginx /var/lib/php/session/
```

Разрешите автозагрузку на старте и запустите службу `php-fpm`

```
sudo systemctl enable php-fpm && sudo systemctl start php-fpm
```

Установка phpMyAdmin

Архив с файлами phpMyAdmin находится в директории с софтом для лабораторной работы. Перенесите его с помощью WinSCP в домашнюю директорию своего пользователя CentOS и распакуйте

```
cd ~
```

```
tar -xzf phpMyAdmin-*
```

```
sudo mkdir /var/www/phpmyadmin
```

```
sudo cp -r ./phpMyAdmin-5.0.2-english/* /var/www/phpmyadmin
```

```
sudo mkdir /var/www/phpmyadmin/tmp
```

```
sudo chown -R nginx.nginx /var/www/phpmyadmin/tmp
```

В файле /etc/nginx/conf.d/default.conf найдите строку `#error_page 404` и приведите конфигурацию, находящуюся ДО этой строки к следующему виду:

```
server {
    listen      80;
    server_name _;
    root /var/www/phpmyadmin;
    charset utf-8;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~ /\.php$ {
        try_files $uri =404;
        fastcgi_pass unix:/run/php-fpm/www.sock;
        fastcgi_index index.php;
```

```

fastcgi_param                                SCRIPT_FILENAME
$document_root$fastcgi_script_name;

        fastcgi_param QUERY_STRING           $query_string;
        include fastcgi_params;
    }

```

/ — корень сайта, перехватывает все запросы к домену, если не найдено более точное совпадение.

~\.php\$ — перехватывает запросы к файлам с расширением php. То есть если в запросе указан файл .php, то запрос отправляется на обработку к php-fpm.

Сгенерируйте произвольную последовательность, чтобы использовать в качестве криптографического ключа для алгоритма Blowfish, используемого в phpMyAdmin.

```
head /dev/urandom | tr -dc A-Za-z0-9 | head -c 32 ; echo ''
```

Скопируйте полученную последовательность символов и вставьте ее в конфигурацию phpMyAdmin, находящуюся в файле /var/www/phpmyadmin/config.inc.php

```

<?php

$cfg['blowfish_secret'] = 'generated_symbols';

?>`

```

Перезапустите службы для принятия изменений конфигурации


```
sudo systemctl restart php-fpm
```

```
sudo systemctl restart nginx
```

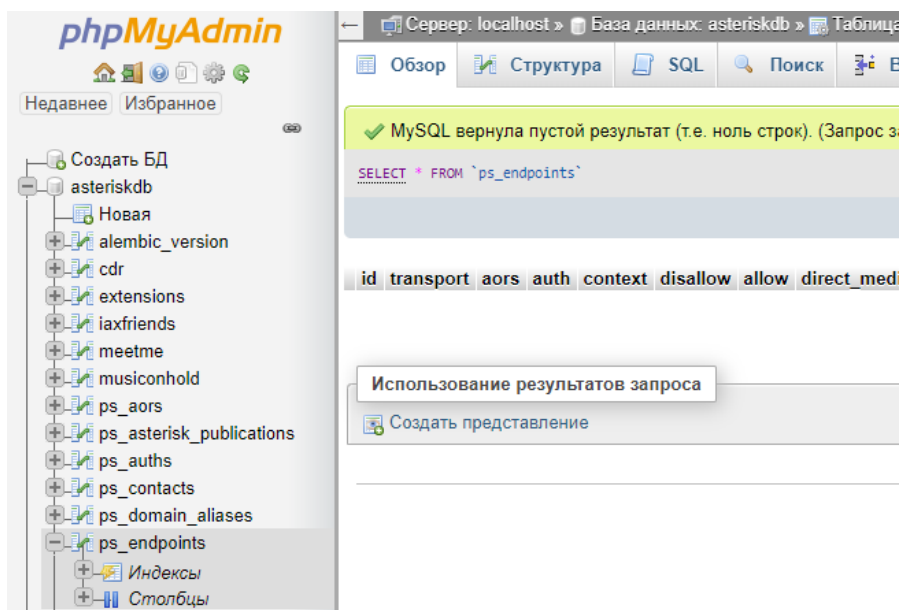
#Работа в интерфейсе phpMyAdmin

Войдите в веб-интерфейс phpMyAdmin, используя URL [http://IP CentOS](http://IP_CentOS) в браузере. В качестве учетных данных для входа используйте пользователя root СУБД.

Для сохранения конфигурации phpMyAdmin требуется отдельная БД, чтобы создать ее, найдите внизу страницы соответствующее сообщение (приведено ниже), нажмите на ссылку Find out why и на следующей странице нажмите Create.

 The phpMyAdmin configuration storage is not completely configured, some extended features have been deactivated. [Find out why.](#)
Or alternately go to 'Operations' tab of any database to set it up there.

Выберите слева базу asteriskdb, в ней таблицу ps_endpoints. Нажмите на Создать представление (Create view).



В появившемся окне укажите название `ps_endpoints_view`, в названия столбцов вставьте укажите `id`, `callerid`, `aors`, `auth`, `context`, `transport`, `disallow`, `allow`, `direct_media`

Создать представление

Детали

OR REPLACE ☐

ALGORITHM UNDEFINED

Определитель

SQL SECURITY

VIEW название ps_endpoints

Названия столбцов id, callerid, aors, auth, context, transport, disallow, allow

AS

```

1 SELECT id, callerid, aors, auth, context, transport,
disallow, allow, direct_media FROM `ps_endpoints`

```

Format

WITH CHECK OPTION

Вперёд

Заккрыть

Ниже вставьте `SELECT id, callerid, aors, auth, context, transport, disallow, allow, direct_media FROM `ps_endpoints`` Нажмите *Format*, для форматирования кода (так выглядит эталонная запись запросов на SQL).

Нажмите Вперед, после этого слева, в новом разделе для asteriskdb - Представления (Views) выберите представление `ps_endpoints_view`. Убедитесь,

что в таблице отображается запись со всеми настроенными параметрами из раздела `endpoints`.

Представление (View) делает возможным отобразить выбранные данные из таблиц БД в виде настраиваемой таблицы (в отображении можно выбрать конкретные столбцы, переименовать их, соединить несколько). Данное представление `ps_endpoints_view` было создано для удобства отображения нужных параметров, поскольку если посмотреть, какие столбцы содержит таблица `ps_endpoints`, можно убедиться, что их число достаточно велико, причем большинство пустует ввиду отсутствия нужды задавать те параметры, которые эти столбцы хранят, а нужные столбцы разбросаны по всей ширине таблицы, что затрудняет читабельность. Для оставшихся таблиц `ps_aors` и `ps_auths` можно не создавать представление как можете самостоятельно убедиться, число столбцов в них невелико.

Во вкладке *Структура* можно настраивать столбцы, которые содержит текущая выбранная таблица, во вкладке *SQL* - писать SQL-запросы, в *Поиск* находить нужную строку по значениям полей. Вкладка *Вставить* предлагает возможность добавления строк в таблицу с помощью заполнения графической формы.

Для ускорения поиска данных и их сортировки, в таблице используются **Индексы**, которые назначаются из числа столбцов таблицы (отображены слева в *древовидной структуре таблицы*). Индексы могут быть уникальными (все соответствующие значения столбцов во всех строках должны быть различны), составными (содержать более одного столбца - в этом случае уникальность проверяется по составной строке из всех указанных столбцов индекса), кластерными (хранятся данные записей таблицы целиком) и некластерными (хранятся только ссылки на записи таблицы). Каждая таблица должна иметь **Первичный ключ**, который обязан быть УНИКАЛЬНЫМ (среди всех значений данного столбца) и ЕДИНСТВЕННЫМ (в одной таблице), первичные ключи таблиц InnoDB являются кластерными индексами. Первичный ключ может быть назначен вручную (в Структуре таблицы в phpMyAdmin будет помечен символом золотого ключа), или создан автоматически (скрыт в таком случае).

#Перенос конфигурации плана набора (extensions) в БД

В файле `/etc/asterisk/extconfig.conf` добавьте в секцию `[settings]` строку:

```
extensions => odbc,asteriskdb
```

В `/etc/asterisk/extensions.conf` в начало каждого контекста добавьте строку

```
switch => Realtime/ИМЯ_КОНТЕКСТА@extensions
```

Пример переноса контекста `outcall` на `node1` в БД:

4. В `/etc/asterisk/extensions.conf` добавить в начало переносимого контекста строку

```
[outcall]
```

```
switch => Realtime/outcall@extensions
```

5. Добавление данных об экстеншенах в таблицу extensions:

```
INSERT INTO extensions VALUES (1, 'outcall', '_1XXX', '1',  
'DIAL', 'PJSIP/${EXTEN}');
```

** не забудьте выбрать базу asteriskdb, в интерфейсе СУБД.*

6. Перезапустите Asterisk, проверьте работоспособность (вызовы совершаются успешно).

Задание. Аналогично приведенному примеру (используя SQL-запросы) или через графическую форму во вкладке *Вставить (Insert)* в phpMyAdmin, перенесите всю имеющуюся конфигурацию плана набора в таблицу extensions, удалите все записи об экстеншенах из файла extensions.conf (кроме параметров switch).

#Перенос конфигурации IAX в БД

В файл /etc/asterisk/extconfig.conf добавьте строку

```
iaxfriends => odbc,asteriskdb
```

Пример

7. В файле /etc/asterisk/iax.conf

добавьте в начало секции [general] строку

```
rtcachefriends=yes
```

8. Перенести конфигурацию пользователя IAX в БД

```
INSERT INTO `iaxfriends` (`id`, `name`, `type`, `secret`,  
`context`, `host`, `trunk`, `auth`, `qualify`) VALUES ('1',  
'iaxuserX', 'friend', 'secretX', 'outcall', 'dynamic', 'yes',  
'md5', 'yes')
```

9. Удалить всю секцию пользователя из iax.conf

10. Перезапустить Asterisk, проверить работоспособность IAX.

Задание. Аналогично приведенному примеру (используя SQL-запросы) или через графическую форму во вкладке *Вставить* в phpMyAdmin, перенесите всю имеющуюся конфигурацию IAX в таблицу iaxfriends. Проверьте работоспособность после переноса конфигурации в Realtime.