Role DATA - All Variants

Functional Components of the Role

The DATA role can be split on one or more servers. A setup with only one DATA server is only recommended for very small installations. The division corresponds to the following functional components:

Function	Description	Access	Quantity	Redundancy strategy
Primary	All write accesses occur here. Furthermore, all functional routines that change data are always executed here. Especially the call distribution is done here.	Write and Read	1	Can be designed redundantly through an active/passive configuration.
Reporting	Creation of complex data evaluations. These operations typically require a high level of storage, computation and I/O resources.	Read	0 - n	In very large systems, different groups of web servers can be distributed to different reporting slaves. Furthermore, several Reporting Slaves can be combined as an Active/Active Cluster
Realtime Statistics	Calculation of real-time statistics at short intervals for logged in users or supervisors and wallboards.	Read	0 - n	In very large systems, different groups of web servers can be distributed to different statistics slaves. Furthermore, several statistic slaves can be combined as an Active/Active Cluster
Customer queries	Creation of customer-specific data evaluations. The outsourcing of this function in a separate system serves primarily to protect the core system.	Read	0 - n	

From the installation point of view, these functional components have no effect because a complete local (synchronized) version of the database is stored on each server in the network. The distribution of the functions results more from the point of view of the "consumers" in which it is possible to configure which server can be accessed for which tasks. In the Web application server, for example, it is possible to specify the database connection for the primary, reporting, and real-time areas separately, so that it is possible to distribute these roles among different servers.

The only aspect of the installation that is affected by the function distribution is that if you are distributing the functions to different servers, you will need to build and configure an appropriate MySQL replication setup. This means that the "Primary" function is forced to run on a Replication Master, while all other functions can run on Replication Slaves.

Another very special type of configuration is a special setup in which two servers are connected in a master-master replication (which in turn can be used as masters for other slaves). In such a highly available configuration, only one of the masters is used as "primary". The other would normally serve as a passive reserve of the HA cluster. However, it is advisable to use this resource more sensibly by having the passive master perform either the "reporting" or the "statistics" function. The function-related IP addresses are managed by the HA manager. Such a configuration has the advantage of offering a high degree of availability without being too wasteful with resources.

Common installation steps

Independent of the function that a DATA server is to take over, the following installation steps must first be carried out on both master and slave.

Linking the data area

Connect data area as described on the page Connection STORE (All Linux except STORE)

Installing the Software

To include the official MySQL software repositories and install the MySQL server, use the following commands:

MySQL 8.x

MySQL 8.x

```
yum -y install libaio
yum -y install https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.rpm
yum -y install mysql-community-server
```

MySQL 5.6

MySQL 5.6

```
yum -y install http://dev.mysql.com/get/mysql-community-release-el6-5.noarch.rpm
yum -y install mysql-community-server
```

Both variants

The MySQL Server service is added to the list of automatically starting services and started with the following command:

MySQL service autostart

```
chkconfig mysqld on
service mysqld start
```

Next, the port shares for the MySQL server service must be entered and permanently stored in the firewall:

Configure firewall

```
firewall-cmd --zone=public --add-port=3306/tcp --permanent
firewall-cmd --reload
```

To simplify the configuration of the MySQL server, a directory is now created where modular configuration files can be stored. In order for these to be loaded from MySQL Server, an entry must be made in the main configuration file. This is done by entering the following commands:

Configure MySQL Server

```
mkdir /etc/my.cnf.d
cat <<EOFF >> /etc/my.cnf
#
# * IMPORTANT: Additional settings that can override those from this file!
# The files must end with '.cnf', otherwise they'll be ignored.
#
!includedir /etc/my.cnf.d/
EOFF
semanage fcontext -a -t mysqld_etc_t "/etc/my\.cnf\.d(/.*)?"
restorecon -R -v /etc/my.cnf.d
```

These commands create the directory, add the load instruction for the modular configuration files to the main configuration file, create a SELINUX security share for the new configuration directory and generate the corresponding security labels.

Next, a modular configuration file with some commented relevant optimization settings is imported.

MySQL 8.x

Load the basic settings

```
wget -P /etc/my.cnf.d http://cdn.jtel.de/downloads/configs/jtel-enhanced-8.cnf
```

The File `/etc/mycnf.d/jtel-enhanced-8.cnf` contains a number of well-commented configuration statements that can be used to optimize the functionality of the MySQL Server. Most of these instructions are commented upon. If necessary, these parameters should be adjusted with caution. However, the default values should be fine for most installations.

MySQL 5.6

Load the basic settings

```
wget -P /etc/my.cnf.d http://cdn.jtel.de/downloads/configs/jtel-enhanced.cnf
```

The File `/etc/mycnf.d/jtel-enhanced.cnf` contains a number of well-commented configuration statements that can be used to optimize the functionality of the MySQL Server. Most of these instructions are commented upon. If necessary, these parameters should be adjusted with caution. However, the default values should be fine for most installations.

Both Variants

Now the MySQL server must be restarted:

Start MySQL Server

```
service mysqld restart
```

After the first start of the MySQL server, the access data for the root user must now be defined.

Since in MySQL a user account consists not only of a username but also of an origin address of the connection, another root user must be created to connect from any origin address.

MySQL 8.x

MySQL 8.x stores a generated password for the root user in the file **/var/log/mysqld.log**

This password must be extracted first. Since it often contains special characters that cannot easily be entered in the command line, the first adjustment is made by entering the password manually.

MySQL 8.x - Create and configure server users

```
mysqladmin -u root -p password '<password>'
```

Then the following command chain is entered to create the additional user:

ATTENTION: replace <password> with the corresponding password.

MySQL 8.x - Create and configure server users

```
mysql -u root -p<password> -v -e"CREATE USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY '<password>'"
mysql -u root -p<password> -v -e"GRANT ALL ON *.* TO 'root'@'%' WITH GRANT OPTION"
mysql -u root -p<password> -v -e"FLUSH PRIVILEGES"
```

MySQL 5.6

ATTENTION: replace <password> with the corresponding password.

MySQL 5.6 - Create and configure server users

```
mysqladmin -u root password '<password>'
mysql -u root -p<password> -v -e"CREATE USER 'root'@'%' IDENTIFIED BY '<password>'"
mysql -u root -p<password> -v -e"GRANT ALL ON *.* TO 'root'@'%' WITH GRANT OPTION"
mysql -u root -p<password> -v -e"FLUSH PRIVILEGES"
```

Both Variants

Next, an additional plugin module is added to MySQL Server. This module is required from jtel software version 3.06 for communication with other software components. For new installations, it should also be installed if it is planned to install older revisions of the software, so that nothing hinders a later update. This is done with the following commands:

Install UDP Send Plugin - MASTER

```
cp /home/jtel/shared/JTELCarrierPortal/Libraries/jtel_udf_udpsend/jtel_udf_udpsend.so /usr/lib64/mysql/plugin/
chown root:root /usr/lib64/mysql/plugin/jtel_udf_udpsend.so
chmod 755 /usr/lib64/mysql/plugin/jtel_udf_udpsend.so
chcon system_u:object_r:lib_t:s0  /usr/lib64/mysql/plugin/jtel_udf_udpsend.so
```

Install UDP Send Plugin - SLAVE

```
cp /home/jtel/shared/JTELCarrierPortal/Libraries/jtel_udf_udpsend/dummy/jtel_udf_udpsend.so /usr/lib64/mysql/plugin/
chown root:root /usr/lib64/mysql/plugin/jtel_udf_udpsend.so
chmod 755 /usr/lib64/mysql/plugin/jtel_udf_udpsend.so
chcon system_u:object_r:lib_t:s0  /usr/lib64/mysql/plugin/jtel_udf_udpsend.so
```

To make the additional function available to SQL procedures, the following command must be executed (replace <password> with the corresponding password):

Register the UDP send command

```
mysql -u root -p<password> -v -e"DROP FUNCTION IF EXISTS udpsend"
mysql -u root -p<password> -v -e"CREATE FUNCTION udpsend RETURNS STRING SONAME 'jtel_udf_udpsend.so'"
```

Important Note

The SQL commands listed above must be executed on a database server **before** it becomes part of a replication group. If the UDP plugin is to be retrofitted on existing DATA servers, a different procedure must be selected:

1. The module must be copied to the plugin directory on **all** servers of the group (both master and slaves) (see code block "Install UDP Send Plugin")
2. The plugin may **only** be registered on the master server. Since the command is also executed on the slaves by replication, it is not necessary to execute the command there as well.

ATTENTION: If the command is executed **without** the UDP plugin being present on all servers in the network, the replication is aborted and can only be repaired by manual intervention.

Adaptation of my.cnf to server RAM

In order for the server to make full use of the provided RAM, a configuration must be adjusted with vi.

This setting should be about 3/4 of the server's RAM, but leave 3-4 GB for mysql and other processes.

vi /etc/my.cnf.d/jtel-enhanced.cnf

```
# For 8 GB RAM
innodb_buffer_pool_size = 5120M

# For 12 GB RAM
innodb_buffer_pool_size = 8192M

# For 16 GB RAM
innodb_buffer_pool_size = 12288M


...
# From 16 GB simply take 3/4 of the RAM
```

MySQL Restart

Finally, the MySQL server is restarted so that all settings are applied:

Restart the MySQL server

```
service mysqld restart
```