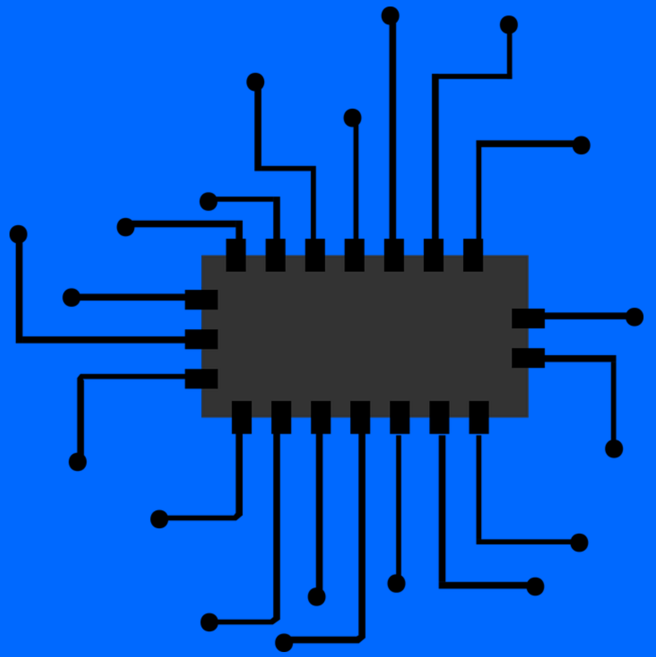
Opensource ICT Solutions

# Partitioning a Zabbix MySQL database

And why you should…



oicts.com/about

# Who am I?

Nathan Liefting

IT Consultant / Zabbix Trainer

@larcorba / larcorba.com

# Our presentation today

**1 – Why would you partition the database**

**2 – How to partition the database**

**3 – Manage the partitions with Perl**

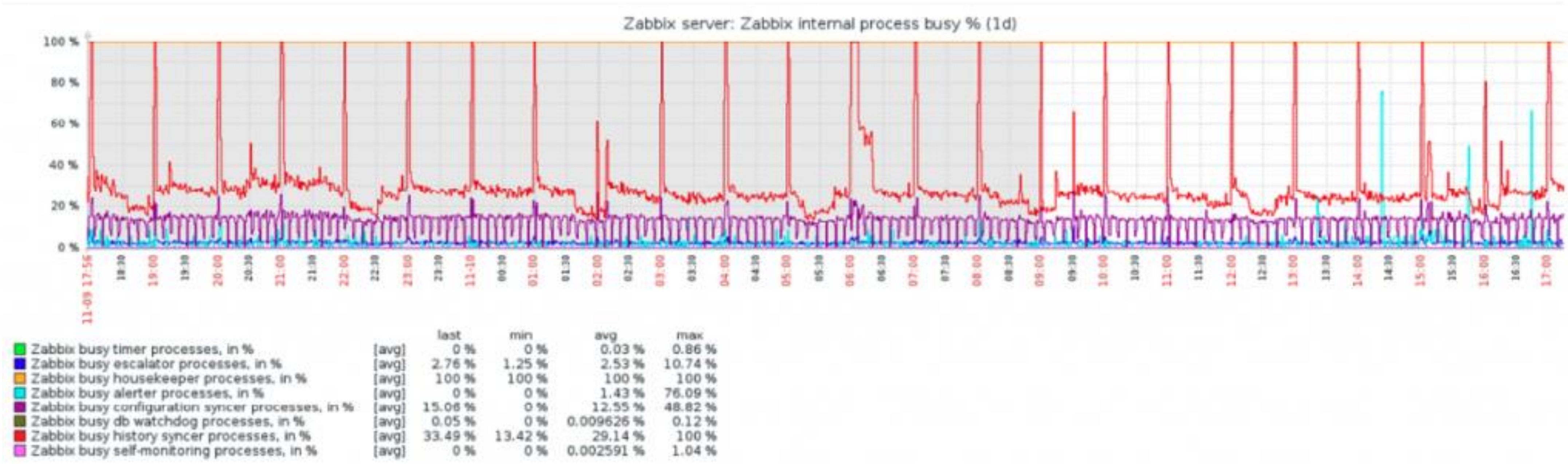
**4 – Manage the partitions with Stored Procedures**

**Let’s wrap things up**

**More content like this**

# 1 – Why would you partition the database

# Housekeeper can’t keep up



| | | | last | min | avg | max |
|---|---|---|---|---|---|---|
| 🟩 | Zabbix busy timer processes, in % | [avg] | 0 % | 0 % | 0.03 % | 0.86 % |
| 🟦 | Zabbix busy escalator processes, in % | [avg] | 2.76 % | 1.25 % | 2.53 % | 10.74 % |
| 🟧 | Zabbix busy housekeeper processes, in % | [avg] | 100 % | 100 % | 100 % | 100 % |
| ■ | Zabbix busy alerter processes, in % | [avg] | 0 % | 0 % | 1.43 % | 76.09 % |
| 🟪 | Zabbix busy configuration syncer processes, in % | [avg] | 15.06 % | 0 % | 12.55 % | 48.82 % |
| 🟫 | Zabbix busy db watchdog processes, in % | [avg] | 0.05 % | 0 % | 0.009626 % | 0.12 % |
| 🟥 | Zabbix busy history syncer processes, in % | [avg] | 33.49 % | 13.42 % | 29.14 % | 100 % |
| ■ | Zabbix busy self-monitoring processes, in % | [avg] | 0 % | 0 % | 0.002591 % | 1.04 % |

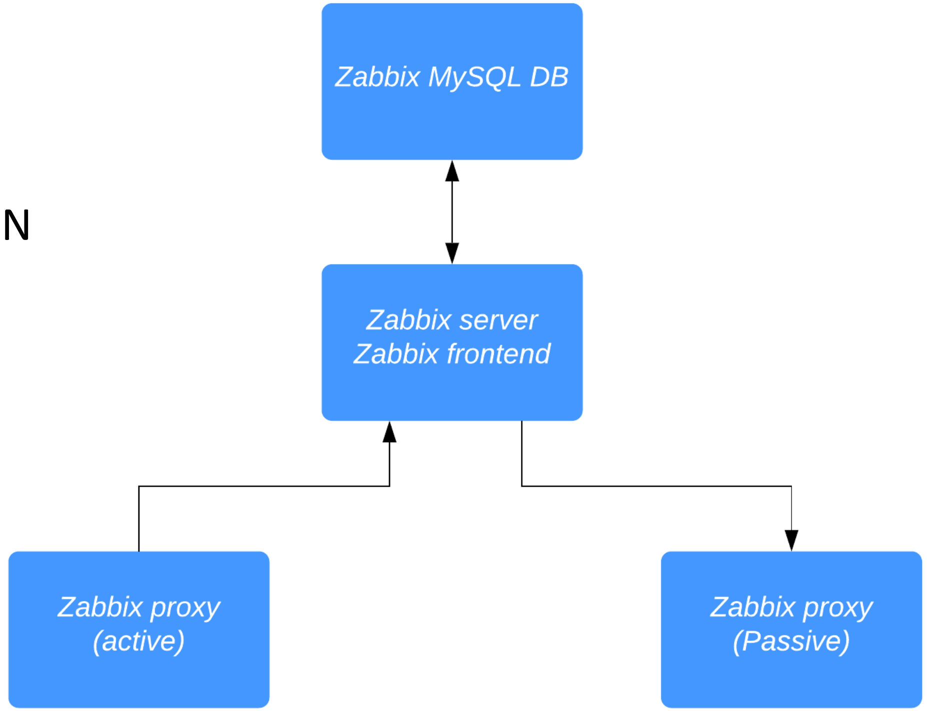
- Housekeeper internal process more than 75% busy all the time
- Looks up old data in the database and deletes this entry by entry
- Impossible to tune the housekeeper process at one point

# Partition before you Zabbix

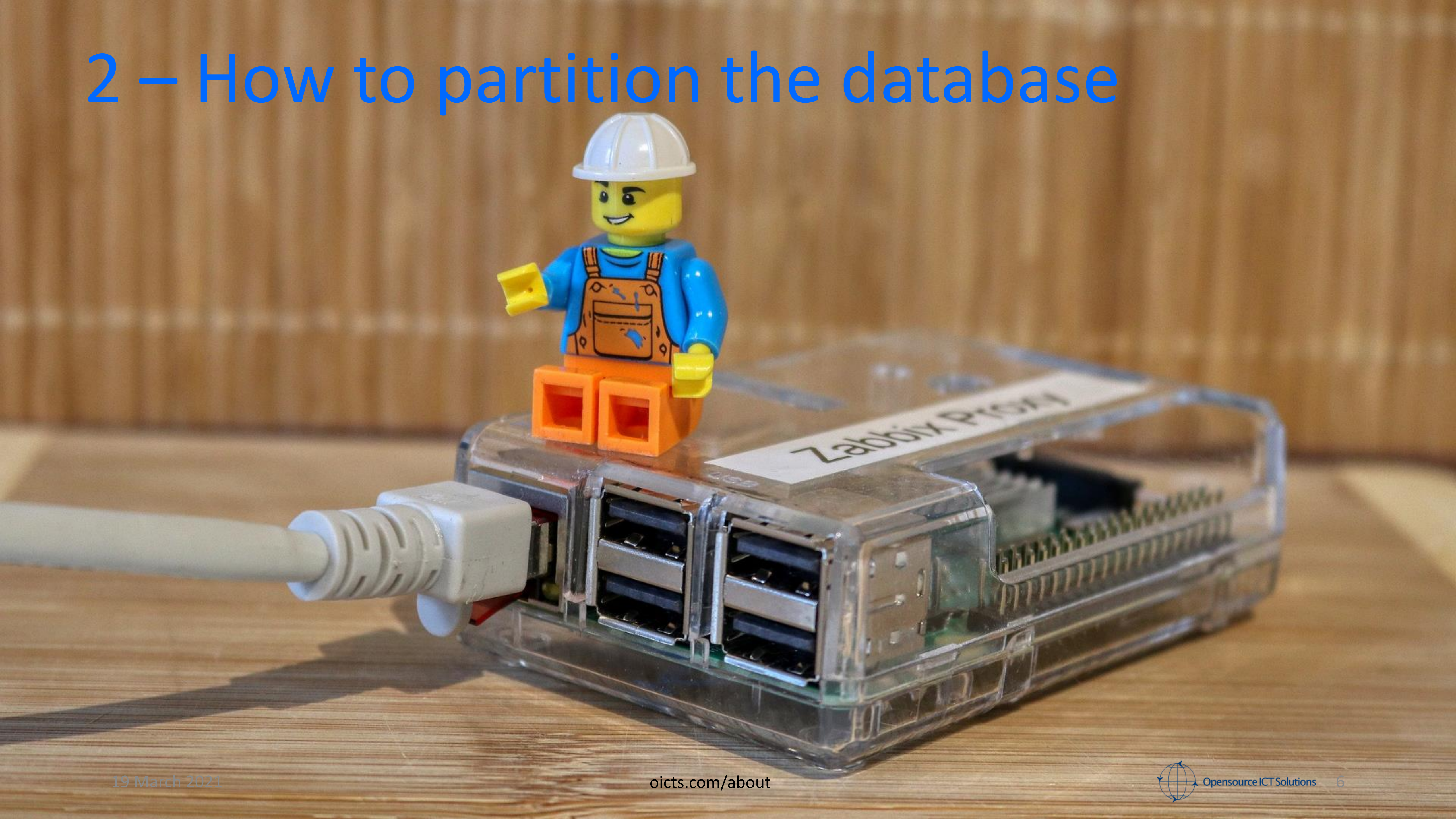
- When building a good Zabbix setup, design is where it all starts
- Partitioning is part of your design

- Expect a server to grow? PARTITION
- Only a proof of concept that might go into production? PARTITION

- Partitioning a big database? It's not fun



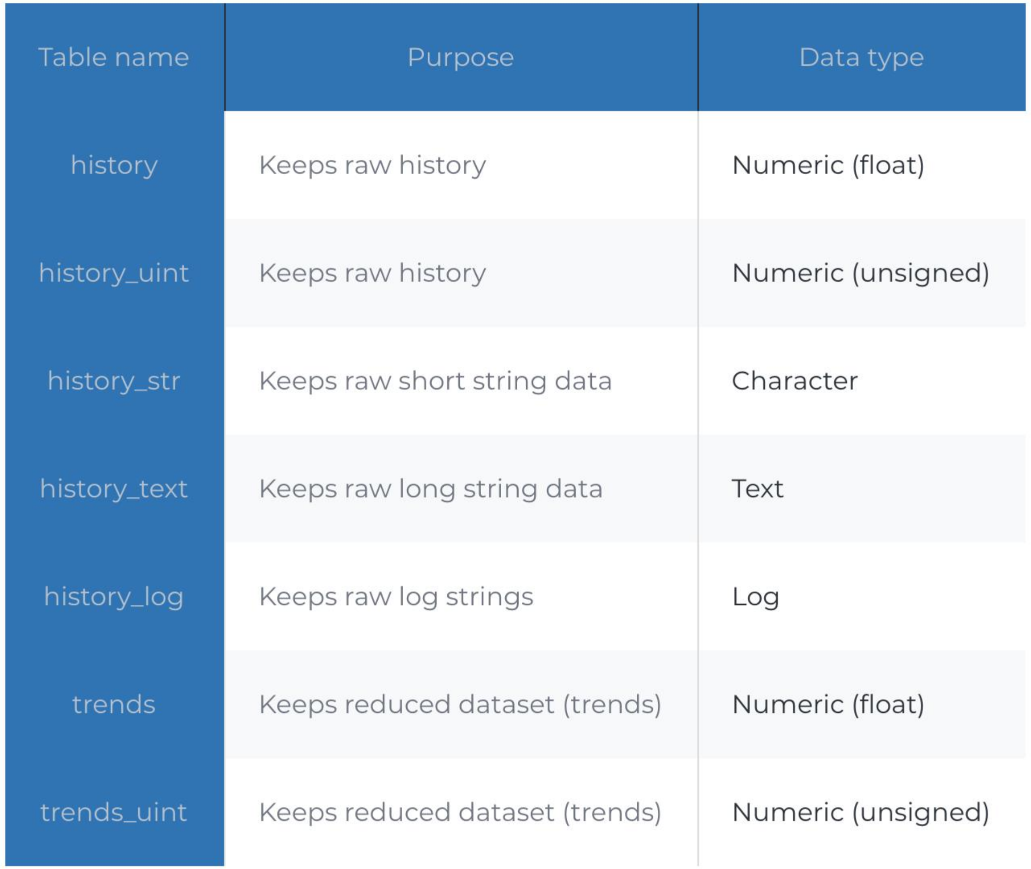
# 2 – How to partition the database



# Let’s prepare to partition

- Partitioning can take a long time
- Use the Linux **screen** command!

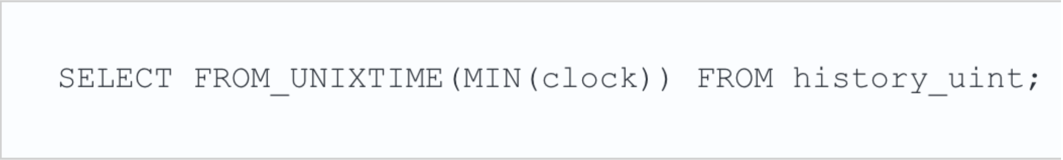
- Partition History tables -> by day
- Partition Trends tables -> by month



| Table name | Purpose | Data type |
|---|---|---|
| history | Keeps raw history | Numeric (float) |
| history_uint | Keeps raw history | Numeric (unsigned) |
| history_str | Keeps raw short string data | Character |
| history_text | Keeps raw long string data | Text |
| history_log | Keeps raw log strings | Log |
| trends | Keeps reduced dataset (trends) | Numeric (float) |
| trends_uint | Keeps reduced dataset (trends) | Numeric (unsigned) |

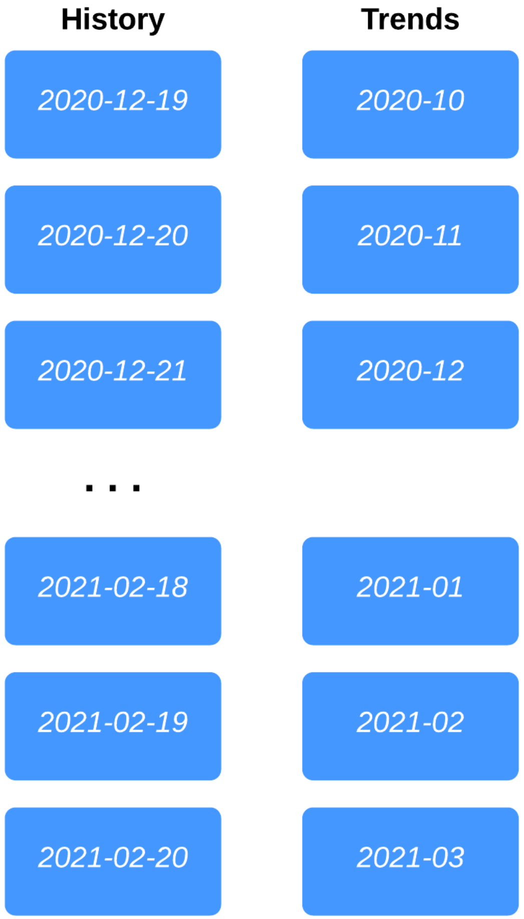
# Prepare your partitioning commands - time

- Get your current timestamp



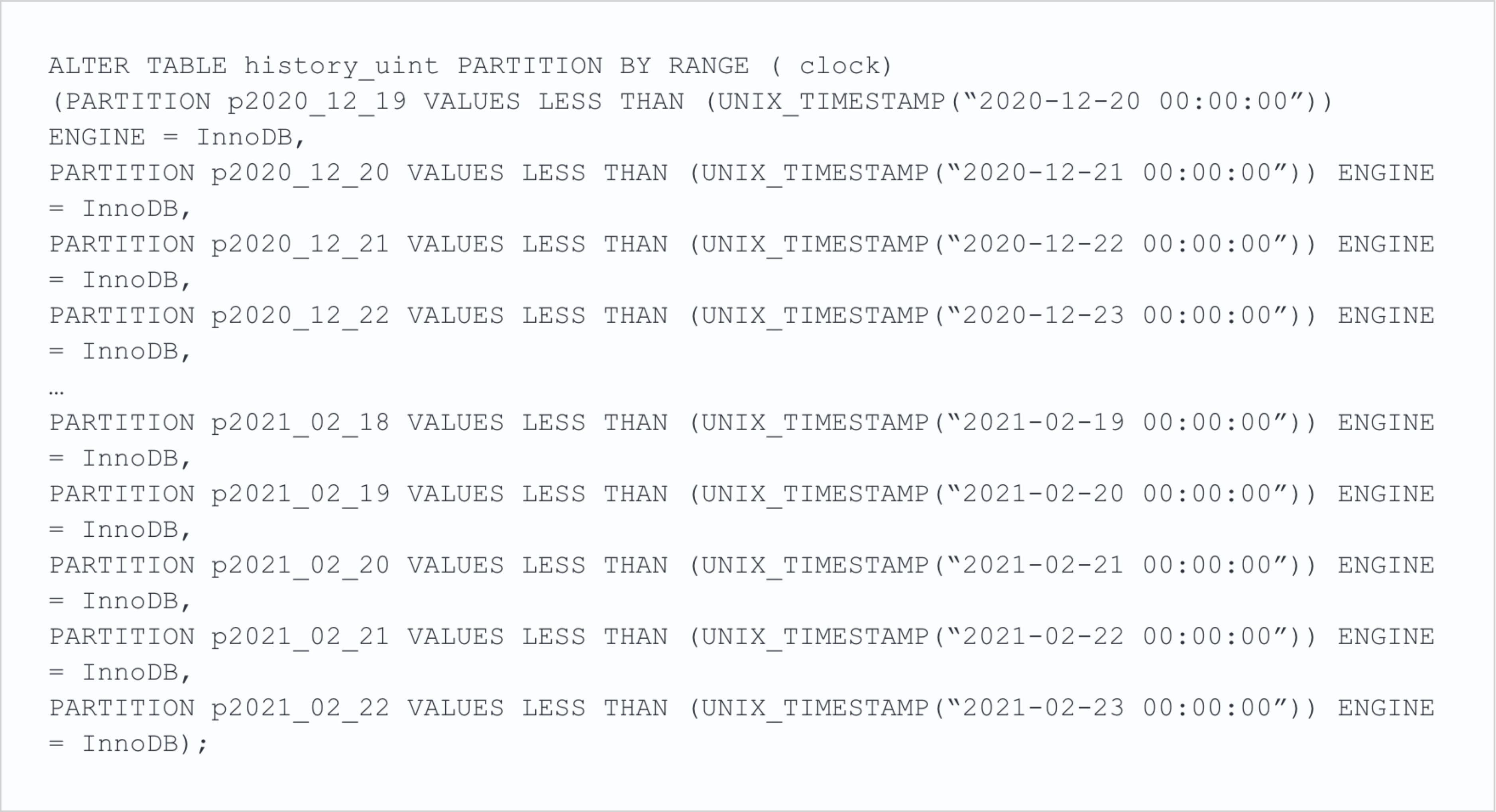
```
SELECT FROM_UNIXTIME(MIN(clock)) FROM history_uint;
```

- Command output example: **2020-12-19 08:09:17**
- Today’s date: **2021-02-19 09:00:00**



| History | Trends |
|---|---|
| *2020-12-19* | *2020-10* |
| *2020-12-20* | *2020-11* |
| *2020-12-21* | *2020-12* |
| . . . | |
| *2021-02-18* | *2021-01* |
| *2021-02-19* | *2021-02* |
| *2021-02-20* | *2021-03* |

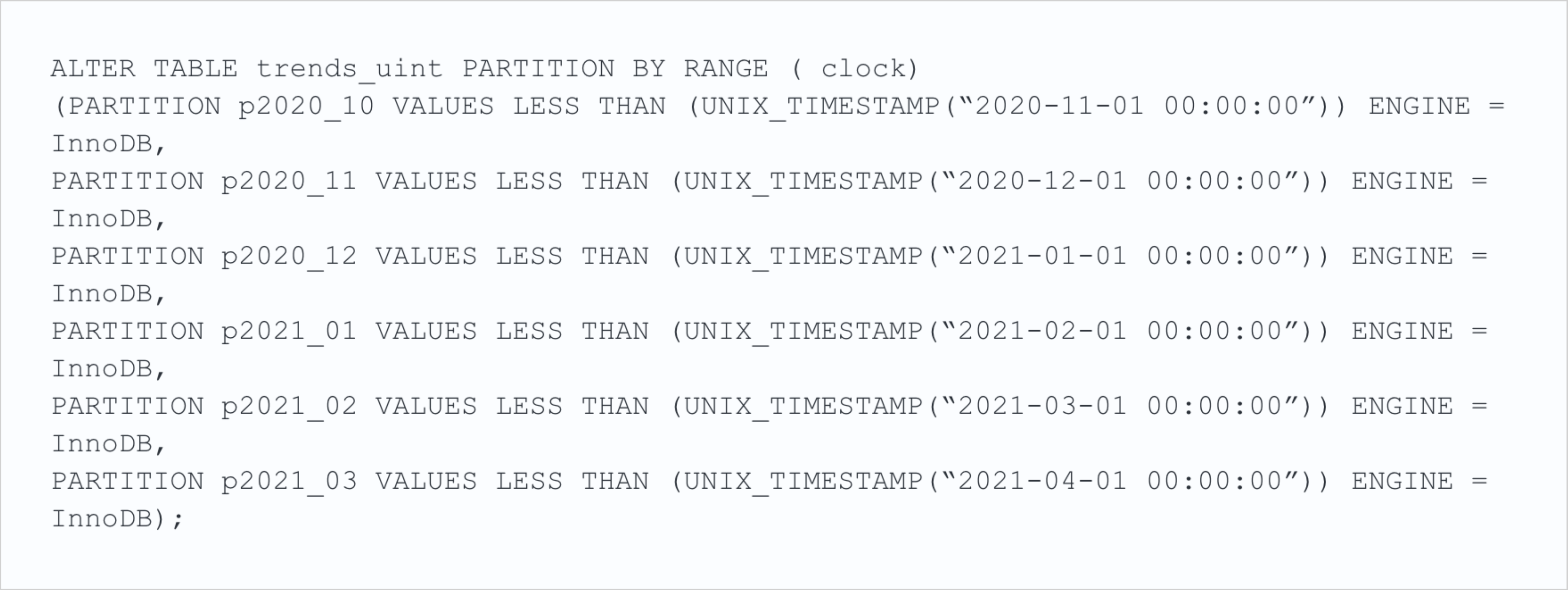
# Prepare your partitioning commands - history



```
ALTER TABLE history_uint PARTITION BY RANGE ( clock)
(PARTITION p2020_12_19 VALUES LESS THAN (UNIX_TIMESTAMP(“2020-12-20 00:00:00”))
ENGINE = InnoDB,
PARTITION p2020_12_20 VALUES LESS THAN (UNIX_TIMESTAMP(“2020-12-21 00:00:00”)) ENGINE
= InnoDB,
PARTITION p2020_12_21 VALUES LESS THAN (UNIX_TIMESTAMP(“2020-12-22 00:00:00”)) ENGINE
= InnoDB,
PARTITION p2020_12_22 VALUES LESS THAN (UNIX_TIMESTAMP(“2020-12-23 00:00:00”)) ENGINE
= InnoDB,
…
PARTITION p2021_02_18 VALUES LESS THAN (UNIX_TIMESTAMP(“2021-02-19 00:00:00”)) ENGINE
= InnoDB,
PARTITION p2021_02_19 VALUES LESS THAN (UNIX_TIMESTAMP(“2021-02-20 00:00:00”)) ENGINE
= InnoDB,
PARTITION p2021_02_20 VALUES LESS THAN (UNIX_TIMESTAMP(“2021-02-21 00:00:00”)) ENGINE
= InnoDB,
PARTITION p2021_02_21 VALUES LESS THAN (UNIX_TIMESTAMP(“2021-02-22 00:00:00”)) ENGINE
= InnoDB,
PARTITION p2021_02_22 VALUES LESS THAN (UNIX_TIMESTAMP(“2021-02-23 00:00:00”)) ENGINE
= InnoDB);
```

- We’ll prepare the same for the **history**, **history_str**, **history_text**, **history_log** tables

# Prepare your partitioning commands - trends

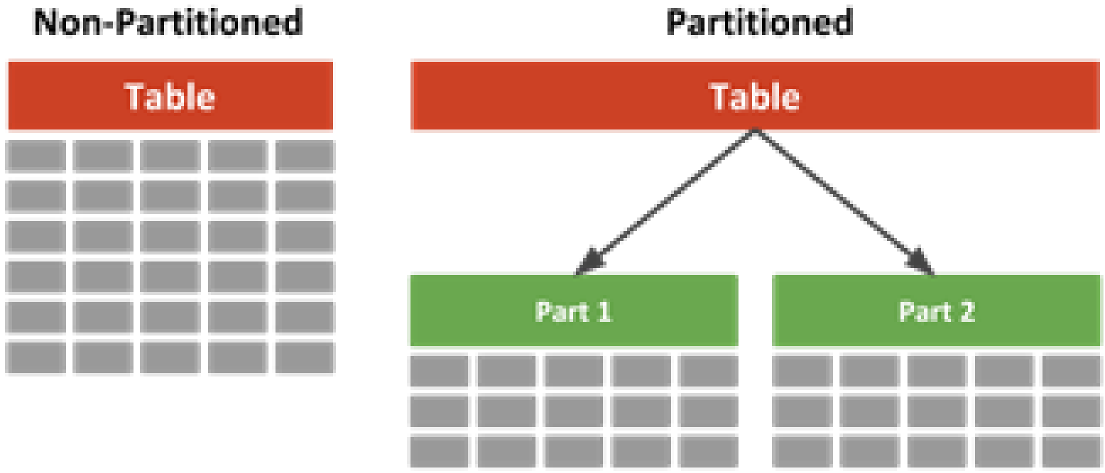


```
ALTER TABLE trends_uint PARTITION BY RANGE ( clock)
(PARTITION p2020_10 VALUES LESS THAN (UNIX_TIMESTAMP(“2020-11-01 00:00:00”)) ENGINE =
InnoDB,
PARTITION p2020_11 VALUES LESS THAN (UNIX_TIMESTAMP(“2020-12-01 00:00:00”)) ENGINE =
InnoDB,
PARTITION p2020_12 VALUES LESS THAN (UNIX_TIMESTAMP(“2021-01-01 00:00:00”)) ENGINE =
InnoDB,
PARTITION p2021_01 VALUES LESS THAN (UNIX_TIMESTAMP(“2021-02-01 00:00:00”)) ENGINE =
InnoDB,
PARTITION p2021_02 VALUES LESS THAN (UNIX_TIMESTAMP(“2021-03-01 00:00:00”)) ENGINE =
InnoDB,
PARTITION p2021_03 VALUES LESS THAN (UNIX_TIMESTAMP(“2021-04-01 00:00:00”)) ENGINE =
InnoDB);
```

- We’ll prepare the same for the **trends** table

# Now it’s time to partition

- Again do not forget to open a **screen**
- Login to MySQL
- Execute the prepared command table by table
- Have patience!



# 3 – Manage the partitions with Perl

# Getting the script

- Download the script here: https://github.com/OpensourceICTSolutions/zabbix-mysql-partitioning-perl

- Save it to:

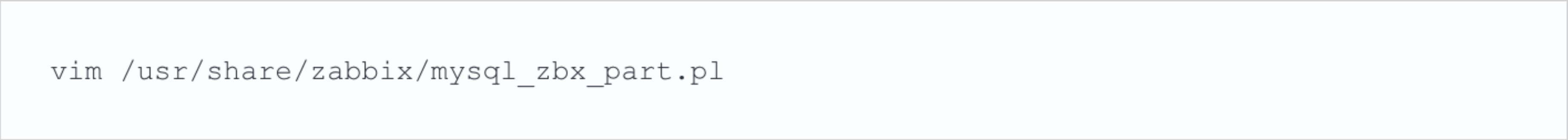
```
/usr/share/zabbix/
```

- Make it executable:

```
chmod +x /usr/share/zabbix/mysql_zbx_part.pl
```

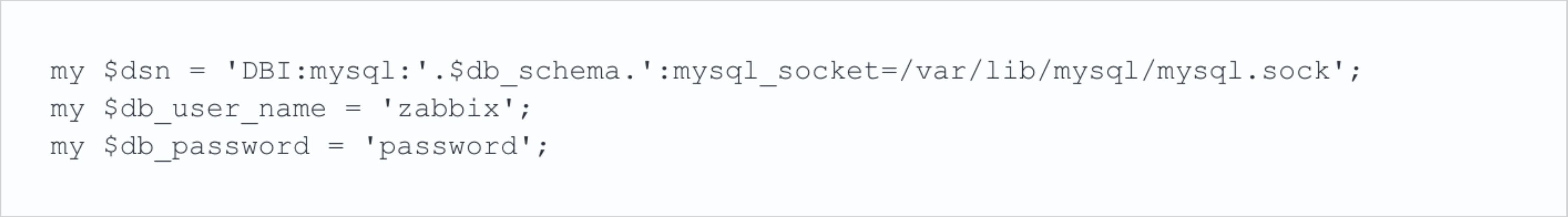
# Edit the script

- Make sure to open the script with your editor:



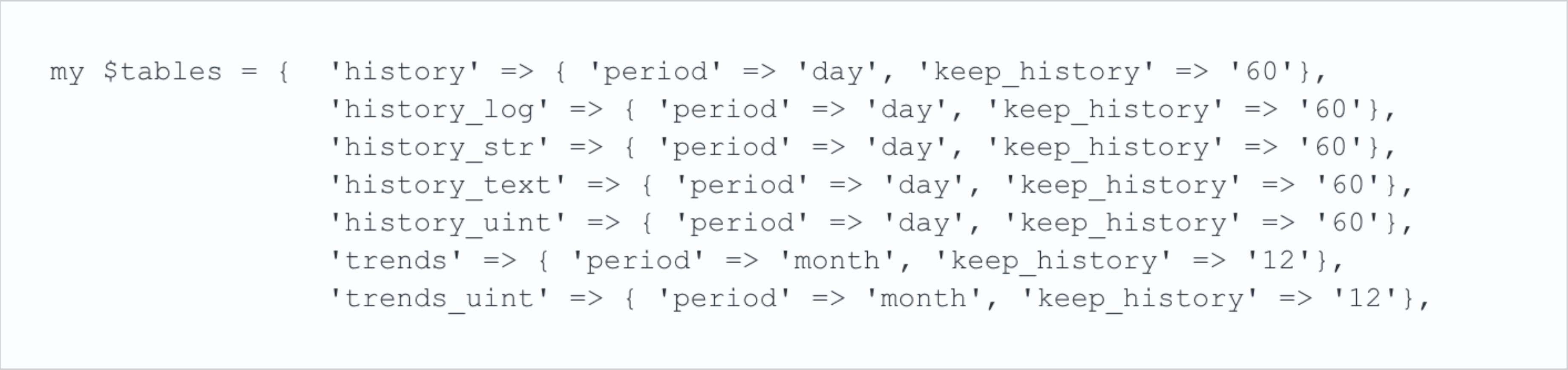
```
vim /usr/share/zabbix/mysql_zbx_part.pl
```

- Make sure to setup your MySQL login details:



```
my $dsn = 'DBI:mysql:'.$db_schema.':mysql_socket=/var/lib/mysql/mysql.sock';
my $db_user_name = 'zabbix';
my $db_password = 'password';
```

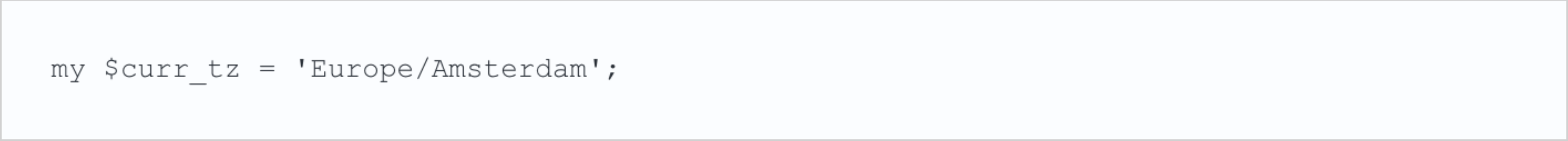
- Then add how long you want to keep data:



```
my $tables = {  'history' => { 'period' => 'day', 'keep_history' => '60'},
                'history_log' => { 'period' => 'day', 'keep_history' => '60'},
                'history_str' => { 'period' => 'day', 'keep_history' => '60'},
                'history_text' => { 'period' => 'day', 'keep_history' => '60'},
                'history_uint' => { 'period' => 'day', 'keep_history' => '60'},
                'trends' => { 'period' => 'month', 'keep_history' => '12'},
                'trends_uint' => { 'period' => 'month', 'keep_history' => '12'},
```

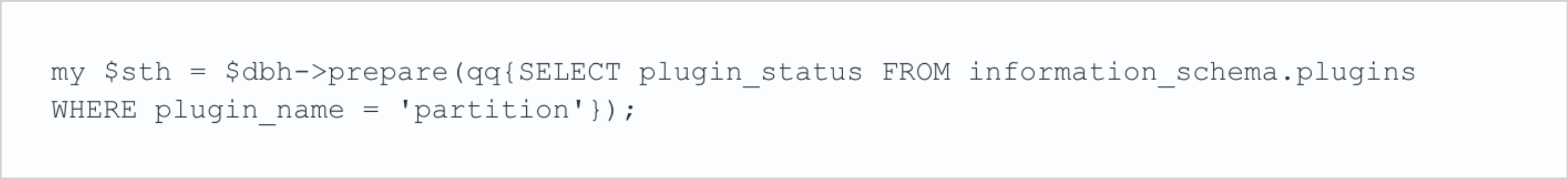
# Edit the script

- Add the correct timezone:

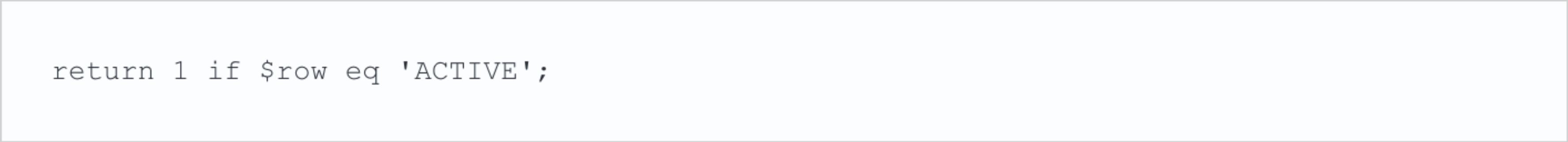


```
my $curr_tz = 'Europe/Amsterdam';
```

- You’re done, unless you use MySQL 5.5 (or earlier). Comment the following:



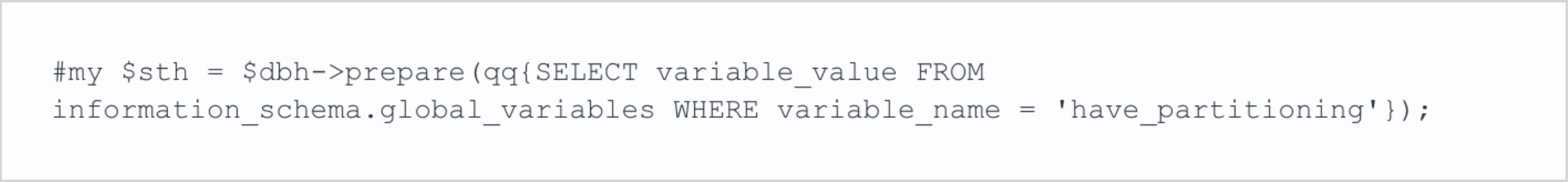
```
my $sth = $dbh->prepare(qq{SELECT plugin_status FROM information_schema.plugins
WHERE plugin_name = 'partition'});
```



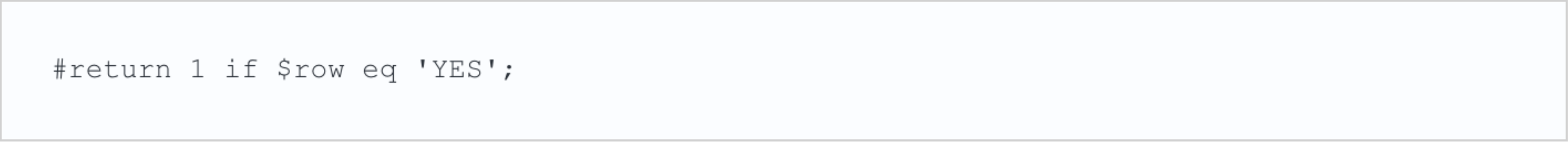
```
return 1 if $row eq 'ACTIVE';
```

# Edit the script

- And uncomment:



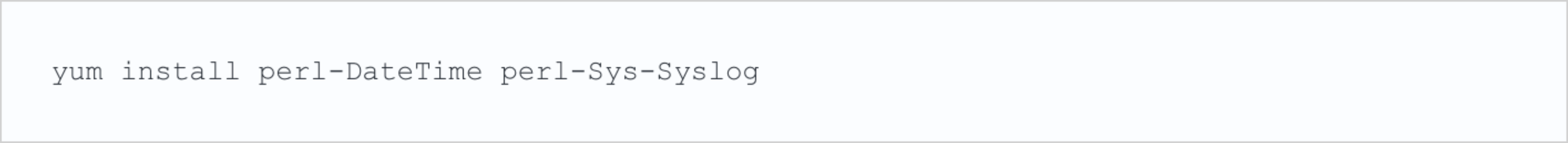
```
#my $sth = $dbh->prepare(qq{SELECT variable_value FROM
information_schema.global_variables WHERE variable_name = 'have_partitioning'});
```



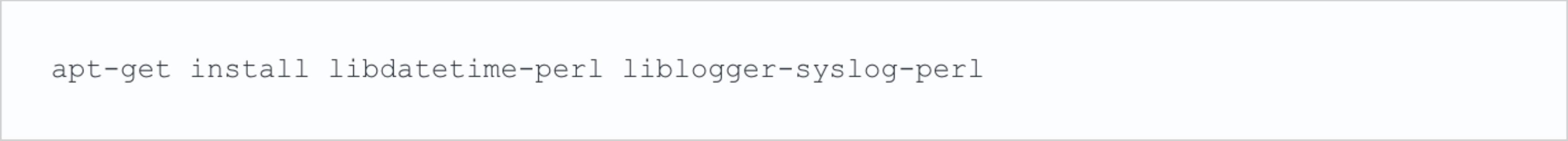
```
#return 1 if $row eq 'YES';
```

# Edit the script

- Last, but not least install the Perl script dependencies:

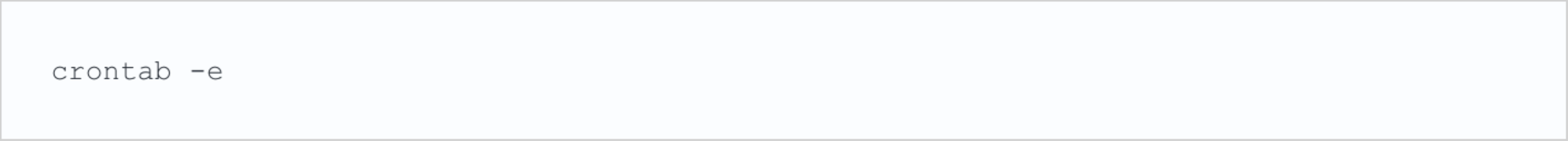


```
yum install perl-DateTime perl-Sys-Syslog
```



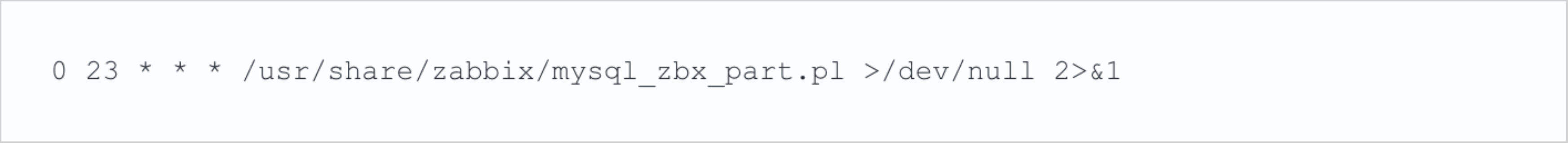
```
apt-get install libdatetime-perl liblogger-syslog-perl
```

- Then add a Cronjob to execute the script with:



```
crontab -e
```

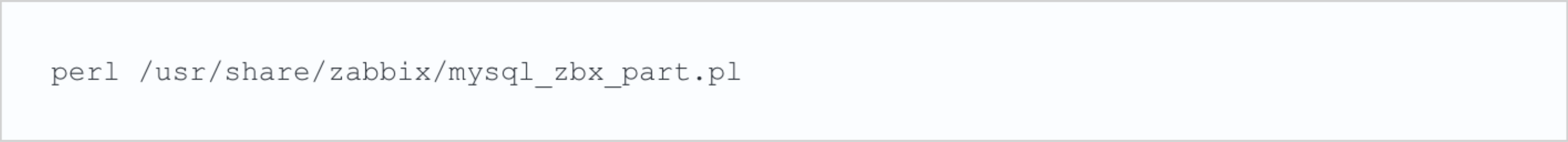
- Add the following line:



```
0 23 * * * /usr/share/zabbix/mysql_zbx_part.pl >/dev/null 2>&1
```

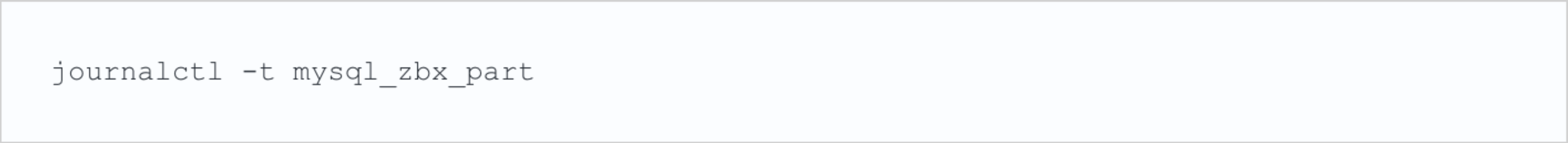
# Execute / monitor the script

- We can execute the script with:



```
perl /usr/share/zabbix/mysql_zbx_part.pl
```

- To check if the script ran use:



```
journalctl -t mysql_zbx_part
```

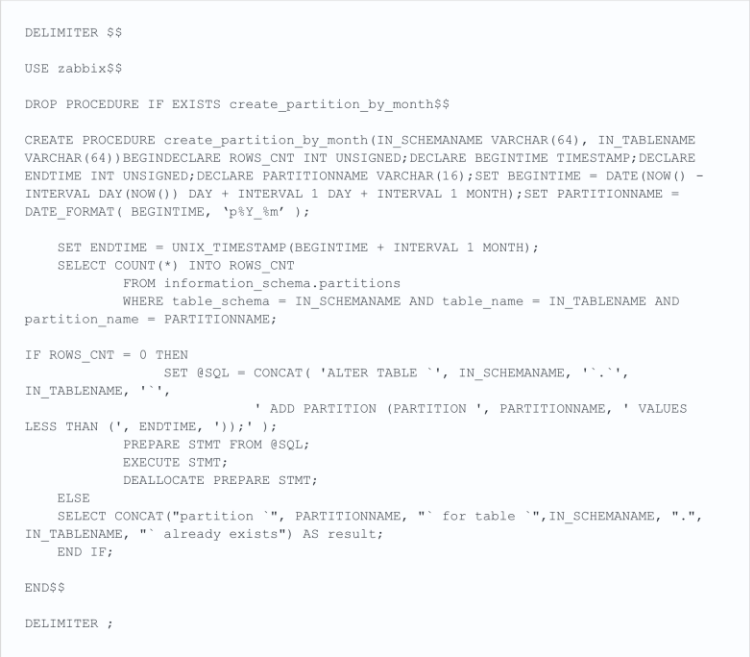
Congratulations, your database is partitioned and will remain so.

# 4 – Manage the partitions with Stored Procedures

# Stored procedures, don’t use them (for this)!

**Cons**

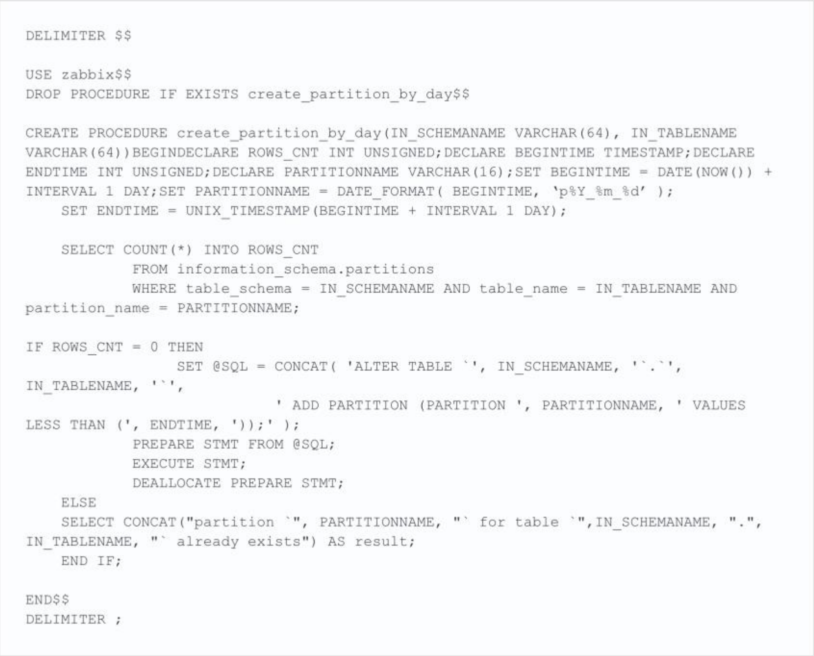
- Hard to troubleshoot
- Messy to setup
- ‘Hidden’ inside your database
- Requires additional knowledge



```
DELIMITER $$

USE zabbix$$

DROP PROCEDURE IF EXISTS create_partition_by_month$$

CREATE PROCEDURE create_partition_by_month(IN_SCHEMANAME VARCHAR(64), IN_TABLENAME
VARCHAR(64))BEGINDECLARE ROWS_CNT INT UNSIGNED;DECLARE BEGINTIME TIMESTAMP;DECLARE
ENDTIME INT UNSIGNED;DECLARE PARTITIONNAME VARCHAR(16);SET BEGINTIME = DATE(NOW() -
INTERVAL DAY(NOW()) DAY + INTERVAL 1 DAY + INTERVAL 1 MONTH);SET PARTITIONNAME =
DATE_FORMAT( BEGINTIME, ‘p%Y_%m’ );

    SET ENDTIME = UNIX_TIMESTAMP(BEGINTIME + INTERVAL 1 MONTH);
    SELECT COUNT(*) INTO ROWS_CNT
            FROM information_schema.partitions
            WHERE table_schema = IN_SCHEMANAME AND table_name = IN_TABLENAME AND
partition_name = PARTITIONNAME;

IF ROWS_CNT = 0 THEN
                 SET @SQL = CONCAT( 'ALTER TABLE `', IN_SCHEMANAME, '`.`',
IN_TABLENAME, '`',
                            ' ADD PARTITION (PARTITION ', PARTITIONNAME, ' VALUES
LESS THAN (', ENDTIME, '));' );
            PREPARE STMT FROM @SQL;
            EXECUTE STMT;
            DEALLOCATE PREPARE STMT;
    ELSE
    SELECT CONCAT("partition `", PARTITIONNAME, "` for table `",IN_SCHEMANAME, ".",
IN_TABLENAME, "` already exists") AS result;
    END IF;

END$$

DELIMITER ;
```

**Pros**

- Some organisations don’t allow external scripts
- We included a guide in the blog post



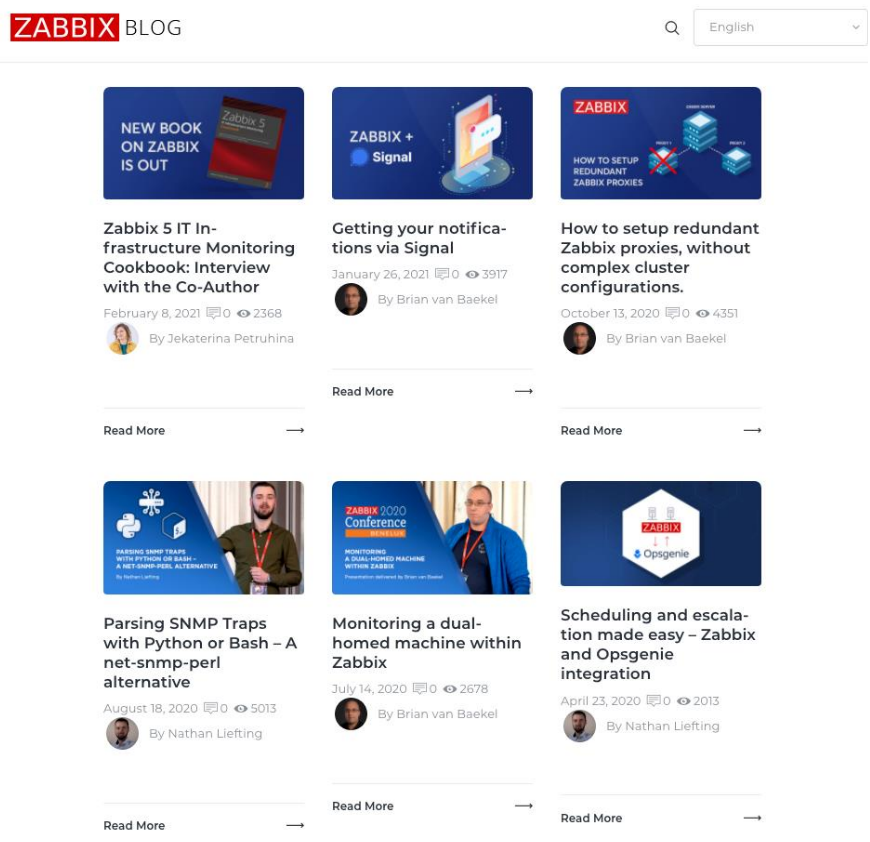
```
DELIMITER $$

USE zabbix$$
DROP PROCEDURE IF EXISTS create_partition_by_day$$

CREATE PROCEDURE create_partition_by_day(IN_SCHEMANAME VARCHAR(64), IN_TABLENAME
VARCHAR(64))BEGINDECLARE ROWS_CNT INT UNSIGNED;DECLARE BEGINTIME TIMESTAMP;DECLARE
ENDTIME INT UNSIGNED;DECLARE PARTITIONNAME VARCHAR(16);SET BEGINTIME = DATE(NOW()) +
INTERVAL 1 DAY;SET PARTITIONNAME = DATE_FORMAT( BEGINTIME, ‘p%Y_%m_%d’ );
    SET ENDTIME = UNIX_TIMESTAMP(BEGINTIME + INTERVAL 1 DAY);

    SELECT COUNT(*) INTO ROWS_CNT
            FROM information_schema.partitions
            WHERE table_schema = IN_SCHEMANAME AND table_name = IN_TABLENAME AND
partition_name = PARTITIONNAME;

IF ROWS_CNT = 0 THEN
                 SET @SQL = CONCAT( 'ALTER TABLE `', IN_SCHEMANAME, '`.`',
IN_TABLENAME, '`',
                            ' ADD PARTITION (PARTITION ', PARTITIONNAME, ' VALUES
LESS THAN (', ENDTIME, '));' );
            PREPARE STMT FROM @SQL;
            EXECUTE STMT;
            DEALLOCATE PREPARE STMT;
    ELSE
    SELECT CONCAT("partition `", PARTITIONNAME, "` for table `",IN_SCHEMANAME, ".",
IN_TABLENAME, "` already exists") AS result;
    END IF;

END$$
DELIMITER ;
```

# Let’s wrap things up

- Preferably partition before you put Zabbix into production
- Prepare your partition commands
- Partition the database, use **screen**
- Setup your partition management (Perl Script / Stored Procedures)
- Keep an eye on your partitions for a few days
- Enjoy!

For a complete text guide see:
https://blog.zabbix.com/partitioning-a-zabbix-mysql-database-with-perl-or-stored-procedures/13531/

# More content like this



ZABBIX BLOG

English

NEW BOOK ON ZABBIX IS OUT
ZABBIX + Signal
ZABBIX HOW TO SETUP REDUNDANT ZABBIX PROXIES

**Zabbix 5 IT Infrastructure Monitoring Cookbook: Interview with the Co-Author**

February 8, 2021 0 2368

By Jekaterina Petruhina

Read More ⟶

**Getting your notifications via Signal**

January 26, 2021 0 3917

By Brian van Baekel

Read More ⟶

**How to setup redundant Zabbix proxies, without complex cluster configurations.**

October 13, 2020 0 4351

By Brian van Baekel

Read More ⟶

PARSING SNMP TRAPS WITH PYTHON OR BASH – A NET-SNMP-PERL ALTERNATIVE
ZABBIX 2020 Conference
MONITORING A DUAL-HOMED MACHINE WITHIN ZABBIX
ZABBIX
Opsgenie

**Parsing SNMP Traps with Python or Bash – A net-snmp-perl alternative**

August 18, 2020 0 5013

By Nathan Liefting

Read More ⟶

**Monitoring a dual-homed machine within Zabbix**

July 14, 2020 0 2678

By Brian van Baekel

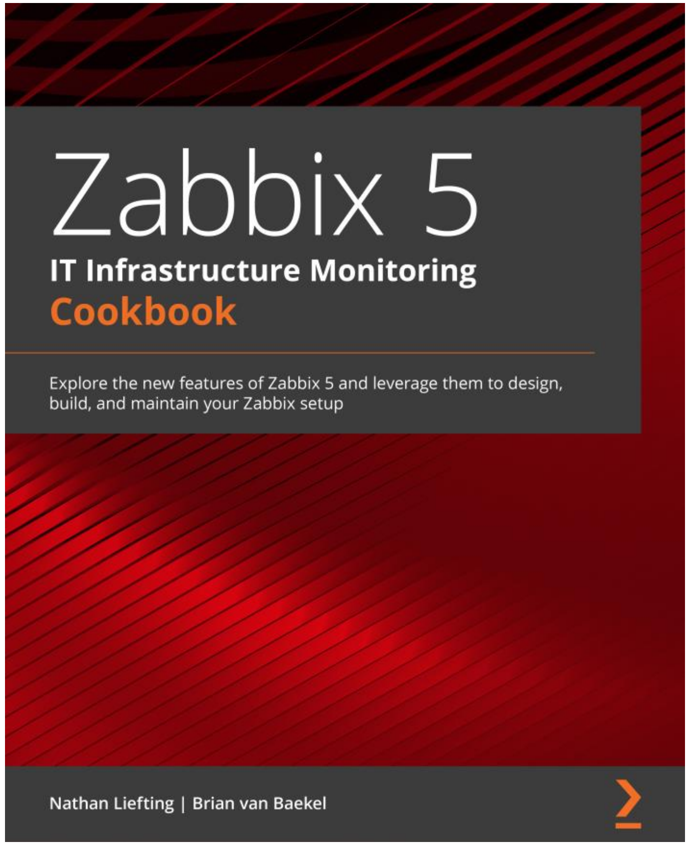
Read More ⟶

**Scheduling and escalation made easy – Zabbix and Opsgenie integration**

April 23, 2020 0 2013

By Nathan Liefting

Read More ⟶



Zabbix 5
IT Infrastructure Monitoring
Cookbook
Explore the new features of Zabbix 5 and leverage them to design, build, and maintain your Zabbix setup
Nathan Liefting | Brian van Baekel

# Questions?