Zabbix：在5分钟内对MySQL / MariaDB数据库表进行分区

WWW.BESTMONITORINGTOOLS.COM
ZABBIX
Partitioning Tables on MySQL Database
Best Monitoring tools

Zabbix：在5分钟内对MySQL / MariaDB数据库表进行分区

46条留言 / Zabbix

在本教程中，我们将逐步学习如何使用分区脚本在MySQL或MariaDB上对Zabbix数据库（历史记录和趋势表）进行分区。

Zabbix从主机收集数据，并使用历史记录和趋势表将其存储在数据库中。Zabbix历史记录保留原始数据（Zabbix收集的每个值），趋势存储平均小时的合并小时数据，平均值为最小值，平均值和最大值。

Zabbix的内务处理负责删除旧的趋势和历史数据。使用SQL删除查询从数据库中删除旧数据可能会对数据库性能产生负面影响。因此，我们许多人都收到了令人讨厌的警报“ `Zabbix housekeeper processes more than 75% busy`”。

使用数据库分区可以轻松解决该问题。分区为每个小时或一天创建表，并在不再需要它们时将其删除。SQL DROP比DELETE语句更有效。

您可以将本教程用于3.0之后的任何Zabbix版本（3.2、3.4、4.0、4.2、4.4、5.0、5.2等）。

在继续之前，请备份Zabbix数据库，但是如果安装是新安装，则无需备份。

内容

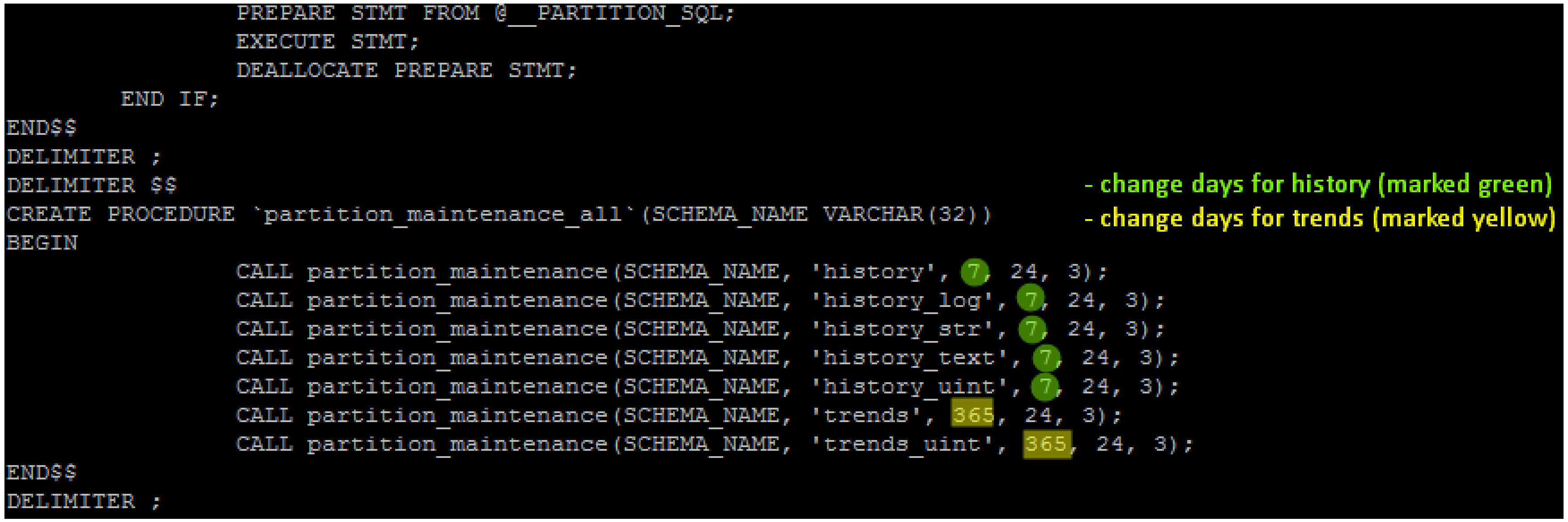
步骤1：下载SQL脚本进行分区

`zbx_db_partitiong.sql`在您的数据库服务器上下载并解压缩SQL脚本“ ”：

```
wget http://bestmonitoringtools.com/dl/zbx_db_partitiong.tar.gz
tar -zxvf zbx_db_partitiong.tar.gz
```

脚本“ `zbx_db_partitiong.sql`”配置为保留7天的历史数据和365天的趋势数据–如果您可以接受这些设置，请转到步骤2。

但是，如果要更改趋势或历史记录的天数，请打开文件“ ***zbx_db_partitiong.sql*** ”，如下图所示更改设置，然后保存文件。



```
                PREPARE STMT FROM @__PARTITION_SQL;
                EXECUTE STMT;
                DEALLOCATE PREPARE STMT;
        END IF;
END$$
DELIMITER ;
DELIMITER $$
CREATE PROCEDURE `partition_maintenance_all`(SCHEMA_NAME VARCHAR(32))
BEGIN
                CALL partition_maintenance(SCHEMA_NAME, 'history', 7, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'history_log', 7, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'history_str', 7, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'history_text', 7, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'history_uint', 7, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'trends', 365, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'trends_uint', 365, 24, 3);
END$$
DELIMITER ;
```

- change days for history (marked green)
- change days for trends (marked yellow)

该图显示了如何在MySQL“创建过程”步骤中更改趋势和历史记录的日期

步骤2：使用SQL脚本创建分区过程

运行脚本的语法是" ***mysql -u'<db_username>'-p'<db_password>'<zb_database_name> <zbx_db_partitiong.sql*** "。

现在，使用您的Zabbix数据库 名称，用户名和密码运行它以创建分区过程：

```
mysql -u 'zabbix' -p'zabbixDBpass' zabbix < zbx_db_partitiong.sql
```

在新安装的Zabbix上，脚本将非常快速地创建MySQL分区过程，但是在大型数据库上，此过程可能持续数小时。

步骤3：自动运行分区过程

我们已经创建了分区过程，但是在我们运行它们之前它们什么也不做！

这一步是最重要的，因为必须使用分区过程定期（每天）删除和创建分区！

不用担心，您不必手动执行此操作。我们可以使用两种工具来执行这些任务：**MySQL事件调度程序**或**Crontab** –选择您喜欢的任何东西。

配置MySQL事件调度程序或Crontab时要小心。如果配置不正确，Zabbix将停止收集数据！您会注意到，在Zabbix日志文件中，通过空图和错误“ [Z3005]查询失败：[1526]表没有值..的分区”。

选项1：使用MySQL事件调度程序自动管理分区（推荐）

默认情况下，MySQL事件调度程序是禁用的。您需要通过在“ [mysqld]”行之后的MySQL配置文件中设置“ *event_scheduler = ON* ”来启用它。

```
[mysqld]
event_scheduler = ON
```

不知道该文件位于何处？如果您使用我的教程来安装和优化Zabbix，则MySQL配置文件（*10_my_tweaks.cnf*）应该位于“ */etc/mysql/mariadb.conf.d/* ”或“ */etc/my.cnf.d/* ”下，否则请尝试使用以下命令进行搜索：

```
sudo grep --include=*.cnf -irl / -e "\[mysqld\]"
```

进行更改后，请重新启动MySQL服务器，以使设置生效！

```
sudo systemctl restart mysql
```

好的！应该启用MySQL事件调度程序，让我们使用以下命令进行检查：

```
root@dbserver:~ $ mysql -u 'zabbix' -p'zabbixDBpass' zabbix -e "SHOW VARIABLES LIKE 'event_scheduler';"
+-----------------+-------+
| Variable_name | Value |
+-----------------+-------+
| event_scheduler | ON |
+-----------------+-------+
```

现在，我们可以创建一个事件，该事件每12小时运行一次“ *partition_maintenance_all* ”过程。

```
mysql -u 'zabbix' -p'zabbixDBpass' zabbix -e "CREATE EVENT zbx_partitioning ON SCHEDULE EVERY 12 HOUR DO CALL
partition_maintenance_all('zabbix');"
```

12小时后，使用以下命令检查事件是否已成功执行。

```
mysql -u 'zabbix' -p'zabbixDBpass' zabbix -e "SELECT * FROM INFORMATION_SCHEMA.events\G"
EVENT_CATALOG: def
                   ...
CREATED: 2020-10-24 11:01:07
LAST_ALTERED: 2020-10-24 11:01:07
LAST_EXECUTED: 2020-10-24 11:43:07
                   ...
```

选项2：使用Crontab自动管理分区

如果您无法使用MySQL事件调度程序，则Crontab是一个不错的选择。使用命令" **sudo crontab -e** "打开crontab文件，并通过在文件中的任意位置添加以下行来添加一个分区Zabbix MySQL数据库的作业（每天凌晨03:30）：

```
30 03 * * * /usr/bin/mysql -u 'zabbix' -p'zabbixDBpass' zabbix -e "CALL partition_maintenance_all('zabbix');"
> /tmp/CronDBpartitiong.log 2>&1
```

保存并关闭文件。

Cron每天都会执行修补（删除旧表并创建新表），并将所有内容记录在文件“ `/tmp/CronDBpartitiong.log`”中。

但是，如果您不耐烦，请立即从终端运行命令：

```
root@dbserver:~ $ mysql -u 'zabbix' -p'zabbixDBpass' zabbix -e "CALL partition_maintenance_all('zabbix');"
+-----------------------------------------------------------+
| msg |
+-----------------------------------------------------------+
| partition_create(zabbix,history,p201910150000,1571180400) |
+-----------------------------------------------------------+
+-----------------------------------------------------------+
 ...etc.
```

然后检查分区状态：

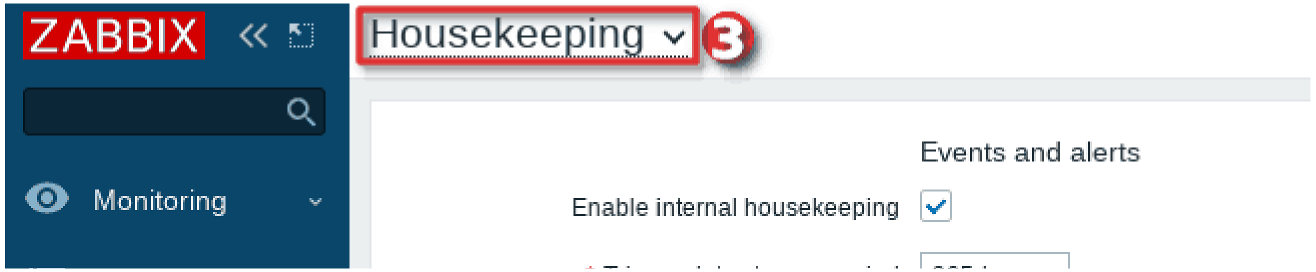
```
root@dbserver:~ $ mysql -u 'zabbix' -p'zabbixDBpass' zabbix -e "show create table history\G"
```

```
Create Table: CREATE TABLE history (
itemid bigint(20) unsigned NOT NULL,
clock int(11) NOT NULL DEFAULT '0',
value double(16,4) NOT NULL DEFAULT '0.0000',
ns int(11) NOT NULL DEFAULT '0',
KEY history_1 (itemid,clock)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin
/*!50100 PARTITION BY RANGE (clock)
(PARTITION p201910140000 VALUES LESS THAN (1571094000) ENGINE = InnoDB,
PARTITION p201910150000 VALUES LESS THAN (1571180400) ENGINE = InnoDB,
PARTITION p201910160000 VALUES LESS THAN (1571266800) ENGINE = InnoDB) */
```

如您在输出中看到的，我们为历史表创建了3个分区。

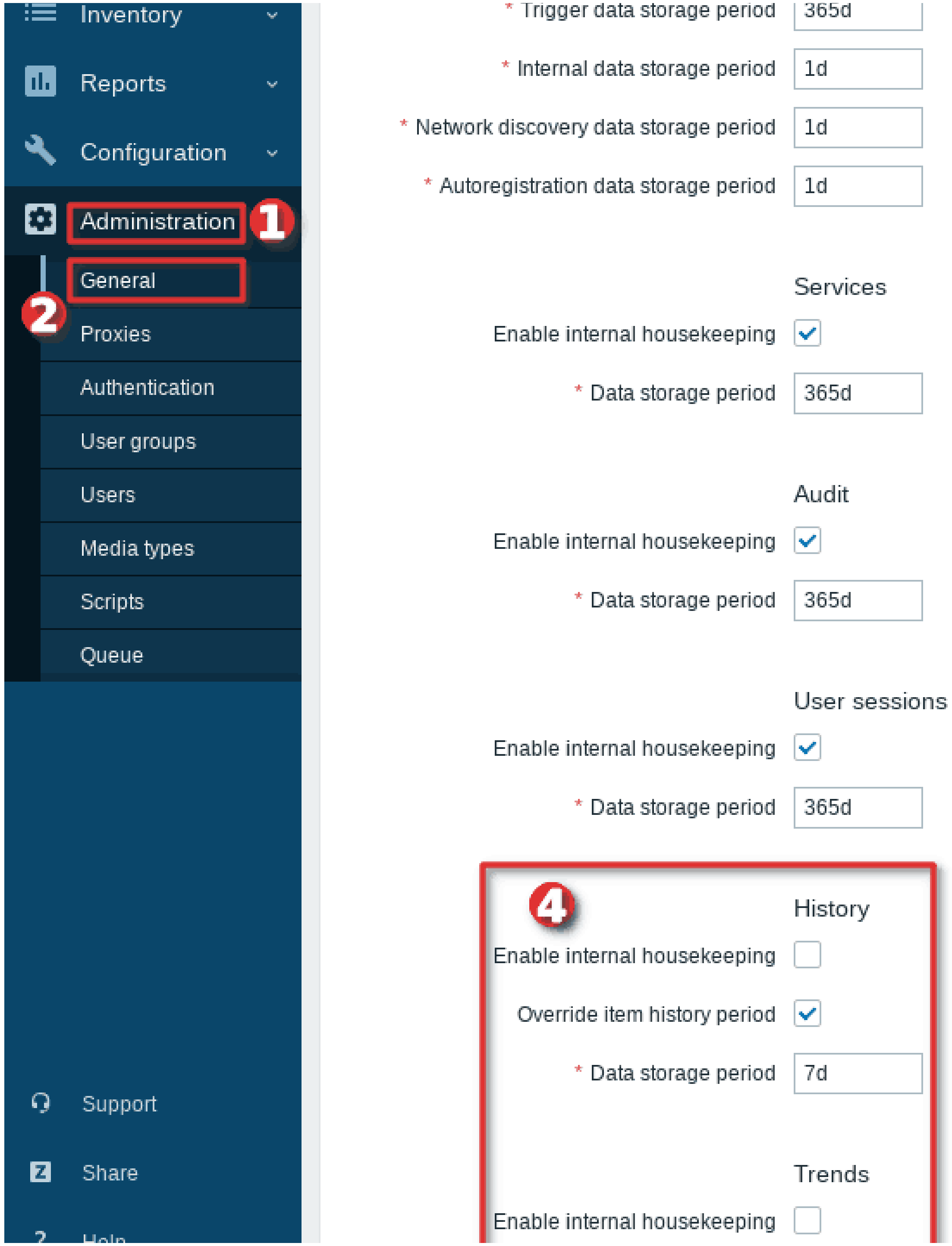
步骤4：在Zabbix前端上配置-管理-一般-管家

如下图所示，在Zabbix前端上配置客房整理。

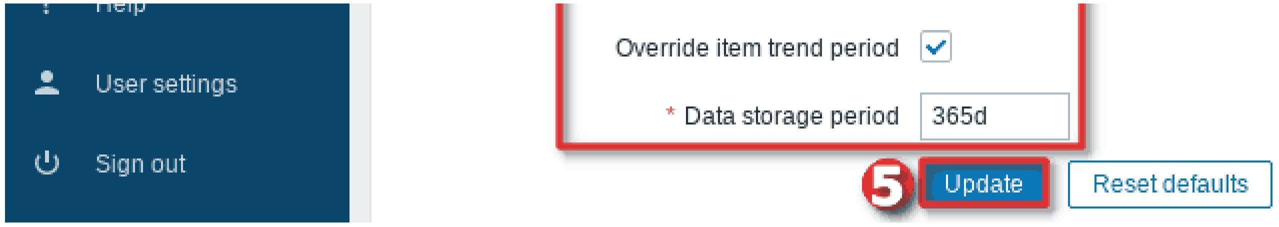


Events and alerts

Enable internal housekeeping ✓



* Trigger data storage period	365d
* Internal data storage period	1d
* Network discovery data storage period	1d
* Autoregistration data storage period	1d
	Services
Enable internal housekeeping	✓
* Data storage period	365d
	Audit
Enable internal housekeeping	✓
* Data storage period	365d
	User sessions
Enable internal housekeeping	✓
* Data storage period	365d
	History
Enable internal housekeeping	☐
Override item history period	✓
* Data storage period	7d
	Trends
Enable internal housekeeping	☐



Override item trend period ✓
* Data storage period 365d

该图显示了如何在Zabbix前端上配置内务管理

如果图片不明了，请按照以下步骤在Zabbix前端上配置管家：

- 导航到“管家”部分：“*管理*”→“一般”→“管家”；
- 删除“历史记录和趋势”部分下“开启内部管家”中的复选标记；
- 在“历史记录和趋势”部分的“覆盖监控项趋势期间”上打勾；
- 在“历史记录和趋势”部分下，为趋势和历史记录定义“*数据存储期*”的天数（必须与数据库分区中配置的天数相同–历史记录应为7天，趋势图应为365天，如果您未更改脚本中的默认设置）；
- 点击“*更新*”按钮。

您完成了！请记住，分区将根据您在分区过程中配置的内容删除历史记录和趋势表。例如，如果您已配置为保留7天的历史记录，分区将在第8天开始删除历史记录。之后，它将每天删除一个历史记录表，以便数据库始终具有7天的历史记录数据。趋势数据也是如此，如果您配置为保留365天的趋势数据，则仅在365天之后它将开始删除旧的趋势表。

恭喜！
您已成功在Zabbix数据库上对MySQL表进行分区！
无需更改其他任何内容，因为其他步骤是可选的。

继续了解更多：
如何更改分区设置
阅读有关脚本中使用的分区过程的更多信息。

步骤5：更改分区设置（历史记录和趋势的天数）

有时可能会因为您最初为Zabbix数据库的历史记录和趋势设置了太多的时间，因此磁盘空间填充得太快。或相反，您没有为历史或趋势配置足够的天数。那该怎么办呢?

您无需再次运行该脚本，只需创建一个将要运行的新过程，而不是原来的过程即可。

a）创建一个新的分区过程

连接到MySQL / MariaDB服务器:

```
mysql -u 'zabbix' -p'zabbixDBpass' zabbix
```

根据您的需要创建一个新的程序，但改变的天趋势和历史上的号码，我将设置为历史30天和400天的趋势:

```
DELIMITER $$
CREATE PROCEDURE partition_maintenance_all_30and400(SCHEMA_NAME VARCHAR(32))
BEGIN
CALL partition_maintenance(SCHEMA_NAME, 'history', 30, 24, 3);
CALL partition_maintenance(SCHEMA_NAME, 'history_log', 30, 24, 3);
CALL partition_maintenance(SCHEMA_NAME, 'history_str', 30, 24, 3);
CALL partition_maintenance(SCHEMA_NAME, 'history_text', 30, 24, 3);
CALL partition_maintenance(SCHEMA_NAME, 'history_uint', 30, 24, 3);
CALL partition_maintenance(SCHEMA_NAME, 'trends', 400, 24, 3);
CALL partition_maintenance(SCHEMA_NAME, 'trends_uint', 400, 24, 3);
END$$
DELIMITER ;
```

b）更新MySQL事件调度程序或Crontab

我们在上一步中创建了分区过程，但是该过程尚未激活！现在，我们必须用新过程替换旧过程，该过程将定期删除和添加分区。根据您在Zabbix实例上配置的内容，选择以下两个选项之一。

选项1：更新MySQL事件计划程序

如果您按照本教程创建了事件调度程序，请使用此命令将旧过程替换为新过程。

```
mysql -u 'zabbix' -p'zabbixDBpass' zabbix -e "ALTER EVENT zbx_partitioning ON SCHEDULE EVERY 12 HOUR DO CALL
partition_maintenance_all_30and400('zabbix');"
```

选项2：更新Crontab

对于使用Crontab的用户，请使用命令“ ***sudo crontab -e*** ”打开crontab文件，注释掉旧的过程作业，然后添加一个新的

```
# old procedure, still exists in the database so it can be used if needed
# 30 03 * * * /usr/bin/mysql -u 'zabbix' -p'zabbixDBpass' zabbix -e "CALL
partition_maintenance_all('zabbix');" > /tmp/CronDBpartitiong.log 2>&1

30 03 * * * /usr/bin/mysql -u 'zabbix' -p'zabbixDBpass' zabbix -e "CALL
partition_maintenance_all_30and400('zabbix');" > /tmp/CronDBpartitiong.log 2>&1
```

保存更改并退出Crontab。

步骤6：有关Zabbix分区脚本的信息

本指南中使用的Zabbix分区SQL脚本包含以下分区过程：

```
DELIMITER $$
CREATE PROCEDURE `partition_create`(SCHEMANAME varchar(64), TABLENAME varchar(64), PARTITIONNAME varchar(64), CLOCK
int)
BEGIN
        /*
           SCHEMANAME = The DB schema in which to make changes
           TABLENAME = The table with partitions to potentially delete
           PARTITIONNAME = The name of the partition to create
        */
        /*
           Verify that the partition does not already exist
        */

        DECLARE RETROWS INT;
        SELECT COUNT(1) INTO RETROWS
        FROM information_schema.partitions
        WHERE table_schema = SCHEMANAME AND table_name = TABLENAME AND partition_description >= CLOCK;

        IF RETROWS = 0 THEN
                /*
                   1. Print a message indicating that a partition was created.
                   2. Create the SQL to create the partition.
                   3. Execute the SQL from #2.
                */
                SELECT CONCAT( "partition_create(", SCHEMANAME, ",", TABLENAME, ",", PARTITIONNAME, ",", CLOCK, ")"
) AS msg;
                SET @sql = CONCAT( 'ALTER TABLE ', SCHEMANAME, '.', TABLENAME, ' ADD PARTITION (PARTITION ',
PARTITIONNAME, ' VALUES LESS THAN (', CLOCK, '));' );
                PREPARE STMT FROM @sql;
                EXECUTE STMT;
                DEALLOCATE PREPARE STMT;
        END IF;
END$$
DELIMITER ;


DELIMITER $$
CREATE PROCEDURE `partition_drop`(SCHEMANAME VARCHAR(64), TABLENAME VARCHAR(64), DELETE_BELOW_PARTITION_DATE BIGINT)
BEGIN
        /*
           SCHEMANAME = The DB schema in which to make changes
           TABLENAME = The table with partitions to potentially delete
           DELETE_BELOW_PARTITION_DATE = Delete any partitions with names that are dates older than this one (yyyy-
mm-dd)
        */
        DECLARE done INT DEFAULT FALSE;
        DECLARE drop_part_name VARCHAR(16);

        /*
           Get a list of all the partitions that are older than the date
           in DELETE_BELOW_PARTITION_DATE.  All partitions are prefixed with
           a "p", so use SUBSTRING TO get rid of that character.
        */
        DECLARE myCursor CURSOR FOR
                SELECT partition_name
                FROM information_schema.partitions
                WHERE table_schema = SCHEMANAME AND table_name = TABLENAME AND CAST(SUBSTRING(partition_name FROM 2)
AS UNSIGNED) < DELETE_BELOW_PARTITION_DATE;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

        /*
           Create the basics for when we need to drop the partition.  Also, create
           @drop_partitions to hold a comma-delimited list of all partitions that
           should be deleted.
        */
        SET @alter_header = CONCAT("ALTER TABLE ", SCHEMANAME, ".", TABLENAME, " DROP PARTITION ");
        SET @drop_partitions = "";

        /*
```

```
           Start looping through all the partitions that are too old.
        */
        OPEN myCursor;
        read_loop: LOOP
                FETCH myCursor INTO drop_part_name;
                IF done THEN
                        LEAVE read_loop;
                END IF;
                SET @drop_partitions = IF(@drop_partitions = "", drop_part_name, CONCAT(@drop_partitions, ",",
drop_part_name));
        END LOOP;
        IF @drop_partitions != "" THEN
                /*
                   1. Build the SQL to drop all the necessary partitions.
                   2. Run the SQL to drop the partitions.
                   3. Print out the table partitions that were deleted.
                */
                SET @full_sql = CONCAT(@alter_header, @drop_partitions, ";");
                PREPARE STMT FROM @full_sql;
                EXECUTE STMT;
                DEALLOCATE PREPARE STMT;

                SELECT CONCAT(SCHEMANAME, ".", TABLENAME) AS `table`, @drop_partitions AS `partitions_deleted`;
        ELSE
                /*
                   No partitions are being deleted, so print out "N/A" (Not applicable) to indicate
                   that no changes were made.
                */
                SELECT CONCAT(SCHEMANAME, ".", TABLENAME) AS `table`, "N/A" AS `partitions_deleted`;
        END IF;
END$$
DELIMITER ;


DELIMITER $$
CREATE PROCEDURE `partition_maintenance`(SCHEMA_NAME VARCHAR(32), TABLE_NAME VARCHAR(32), KEEP_DATA_DAYS INT,
HOURLY_INTERVAL INT, CREATE_NEXT_INTERVALS INT)
BEGIN
        DECLARE OLDER_THAN_PARTITION_DATE VARCHAR(16);
        DECLARE PARTITION_NAME VARCHAR(16);
        DECLARE OLD_PARTITION_NAME VARCHAR(16);
        DECLARE LESS_THAN_TIMESTAMP INT;
        DECLARE CUR_TIME INT;
        CALL partition_verify(SCHEMA_NAME, TABLE_NAME, HOURLY_INTERVAL);
        SET CUR_TIME = UNIX_TIMESTAMP(DATE_FORMAT(NOW(), '%Y-%m-%d 00:00:00'));

        SET @__interval = 1;
        create_loop: LOOP
                IF @__interval > CREATE_NEXT_INTERVALS THEN
                        LEAVE create_loop;
                END IF;

                SET LESS_THAN_TIMESTAMP = CUR_TIME + (HOURLY_INTERVAL * @__interval * 3600);
                SET PARTITION_NAME = FROM_UNIXTIME(CUR_TIME + HOURLY_INTERVAL * (@__interval - 1) * 3600,
'p%Y%m%d%H00');
                IF(PARTITION_NAME != OLD_PARTITION_NAME) THEN
                        CALL partition_create(SCHEMA_NAME, TABLE_NAME, PARTITION_NAME, LESS_THAN_TIMESTAMP);
                END IF;
                SET @__interval=@__interval+1;
                SET OLD_PARTITION_NAME = PARTITION_NAME;
        END LOOP;

        SET OLDER_THAN_PARTITION_DATE=DATE_FORMAT(DATE_SUB(NOW(), INTERVAL KEEP_DATA_DAYS DAY), '%Y%m%d0000');
        CALL partition_drop(SCHEMA_NAME, TABLE_NAME, OLDER_THAN_PARTITION_DATE);

END$$
DELIMITER ;


DELIMITER $$
CREATE PROCEDURE `partition_verify`(SCHEMANAME VARCHAR(64), TABLENAME VARCHAR(64), HOURLYINTERVAL INT(11))
BEGIN
        DECLARE PARTITION_NAME VARCHAR(16);
        DECLARE RETROWS INT(11);
        DECLARE FUTURE_TIMESTAMP TIMESTAMP;

        /*
         * Check if any partitions exist for the given SCHEMANAME.TABLENAME.
         */
        SELECT COUNT(1) INTO RETROWS
        FROM information_schema.partitions
        WHERE table_schema = SCHEMANAME AND table_name = TABLENAME AND partition_name IS NULL;

        /*
         * If partitions do not exist, go ahead and partition the table
         */
        IF RETROWS = 1 THEN
```

```
                /*
                 * Take the current date at 00:00:00 and add HOURLYINTERVAL to it.  This is the timestamp below
which we will store values.
                 * We begin partitioning based on the beginning of a day.  This is because we don't want to generate
a random partition
                 * that won't necessarily fall in line with the desired partition naming (ie: if the hour interval
is 24 hours, we could
                 * end up creating a partition now named "p201403270600" when all other partitions will be like
"p201403280000").
                 */
                SET FUTURE_TIMESTAMP = TIMESTAMPADD(HOUR, HOURLYINTERVAL, CONCAT(CURDATE(), " ", '00:00:00'));
                SET PARTITION_NAME = DATE_FORMAT(CURDATE(), 'p%Y%m%d%H00');

                -- Create the partitioning query
                SET @__PARTITION_SQL = CONCAT("ALTER TABLE ", SCHEMANAME, ".", TABLENAME, " PARTITION BY
RANGE(`clock`)");
                SET @__PARTITION_SQL = CONCAT(@__PARTITION_SQL, "(PARTITION ", PARTITION_NAME, " VALUES LESS THAN
(", UNIX_TIMESTAMP(FUTURE_TIMESTAMP), "));");

                -- Run the partitioning query
                PREPARE STMT FROM @__PARTITION_SQL;
                EXECUTE STMT;
                DEALLOCATE PREPARE STMT;
        END IF;
END$$
DELIMITER ;


DELIMITER $$
CREATE PROCEDURE `partition_maintenance_all`(SCHEMA_NAME VARCHAR(32))
BEGIN
                CALL partition_maintenance(SCHEMA_NAME, 'history', 7, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'history_log', 7, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'history_str', 7, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'history_text', 7, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'history_uint', 7, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'trends', 365, 24, 3);
                CALL partition_maintenance(SCHEMA_NAME, 'trends_uint', 365, 24, 3);
END$$
DELIMITER ;
```

Need more information ? Watch this video about MySQL database partitioning for Zabbix.

Thank you for reading.