MySQL - Kleine Hilfe

Abfrage über mehrere Tabellen mit inner join

```
select familienname,vorname,1_jahr_bezahlt,2_jahr_bezahlt,avastId from
t_users u  inner join t_hosts h on (u.userId=h.userId) inner join t_avast a
on (h.hostId=a.hostId);
```

zeigen aller Views

```
SHOW FULL TABLES IN database_name WHERE TABLE_TYPE LIKE 'VIEW';
```

zeige die CREATE VIEW Syntax

```
SHOW CREATE VIEW my_view_name;
```

fügt eine neue Spalte in eine existierenden Tabelle ein

```
alter table t_hosts add mac_address varchar(17);
```

ändert den Spaltennamen einer Tabelle

```
alter table t_hosts change mac_adress mac_address varchar(17);
```

zeigt Spalteneigenschaften einer Tabelle an

```
explain t_hosts;
```

ändert einen Tabelleneintrag

```
update t_hosts set hostname='motog-nadin' where hostId='4';
```

zeigt die Rechte des Users an

```
show grants for 'marko'@'marko-ThinkPad-T500.tuxnet.local';
```

löscht alle Privilegien des Users

```
revoke all privileges on `tuxnetlocal`.* from 'marko'@'marko-thinkpad-
t500.tuxnet.local';
```

User Rechte vergeben

```
GRANT all privileges on `KMyMoney`.* TO 'marko'@'notebook-
marko.tuxnet.local' IDENTIFIED BY 'password';
```

Tips to reduce the size of MySQL Database

Backup, first but not least

Best practices suggest to backup your database before take any dangerous action. MySQL and MariaDB include the mysqldump utility to simplify the process to create a backup of a database or system of databases.

```
# Backup Single DB
mysqldump -u [username] -p [databaseName] > [filename]-$(date +%F).sql
# Backup Entire DBMS
mysqldump --all-databases --single-transaction --quick --lock-tables=false >
full-backup-$(date +%F).sql -u root -p
```

If you are using MariaDB, you have the option of full or incremental backup via `mariabackup` utility.

```
$ mariabackup --backup --target-dir=/var/mariadb/backup/ --user=myuser --
password=mypassword
```

List MySQL Table and Index Size

Use the following queries to monitor and evaluation table and index size.

Query below returns the size of each Database in MB.

```
MariaDB [(none)]> SELECT table_schema "DB Name", Round(Sum(data_length +
index_length) / 1024 / 1024, 1) "DB Size in MB"
 FROM information_schema.tables
 GROUP BY table_schema;
+--------------------+---------------+
| DB Name            | DB Size in MB |
```

```
+--------------------+---------------+
| hope               |       75714.0 |
| information_schema |           0.2 |
| mysql              |           1.9 |
| performance_schema |           0.0 |
+--------------------+---------------+
```

Query below returns the size of each Table in MB.

```
SELECT table_schema as `Database`,table_name AS `Table`,round(((data_length
+ index_length) / 1024 / 1024), 2) `Size in MB`
FROM information_schema.TABLES
ORDER BY (data_length + index_length) DESC
LIMIT 5;       -- adjust it according to your needs
+----------+-----------------------+------------+
| Database | Table                 | Size in MB |
+----------+-----------------------+------------+
| hope     | eth_products          |   44029.54 |
| hope     | eth_customers         |   28868.08 |
| hope     | eth_emails            |    1423.92 |
| hope     | eth_id                |    1392.43 |
| mysql    | help_topic            |       1.38 |
+----------+-----------------------+------------+
```

Query below returns index size ordered from the ones using the most.

```
SELECT table_schema as
database_name,table_name,round(index_length/1024/1024,2) as index_size
FROM information_schema.tables
WHERE table_type = 'BASE TABLE'
     and table_schema not in ('information_schema',
'sys','performance_schema', 'mysql')
     and table_schema = 'your database name'
ORDER BY index_size desc;
+---------------+-----------------------+------------+
| database_name | table_name            | index_size |
+---------------+-----------------------+------------+
| hope          | eth_products          |   18561.74 |
| hope          | eth_customers         |   12037.89 |
| hope          | eth_emails            |     638.70 |
| hope          | eth_id                |     607.20 |
| hope          | eth_temp              |       0.00 |
+---------------+-----------------------+------------+
```

Delete Unwanted Data

The easier and the hardest way to reduce MySQL size is by deleting all the unwanted data. DB admins usually fill in DB tables or columns with unnecessary data. A DB schema reevaluation is essential to identify such cases.

The following query helps determine the last time a table is updated.

```
SELECT table_schema,table_name,create_time,update_time from
information_schema.tables
WHERE table_schema not in ('information_schema','mysql')
    and engine is not null and ((update_time < (now() - interval 1 day)) or
update_time is NULL)
LIMIT 5;
+--------------+-----------------------+---------------------+------------
--------+
| table_schema | table_name            | create_time         | update_time
|
+--------------+-----------------------+---------------------+------------
--------+
| MariaDB      | eth_table1            | 2019-08-23 20:52:51 | 2019-08-23
22:54:34 |
| MariaDB      | eth_table2            | 2019-08-23 19:20:23 | 2019-08-23
19:20:23 |
| MariaDB      | eth_table3            | 2019-08-23 19:20:29 | 2019-08-23
19:20:29 |
| MariaDB      | eth_table4            | 2019-08-26 19:18:04 | 2019-08-26
21:05:10 |
| MariaDB      | eth_temp              | 2019-08-25 01:52:33 | 2019-08-25
21:16:16 |
+--------------+-----------------------+---------------------+------------
--------+
```

When all unused data are unidentified the following commands will help you delete them :

Be extra careful when using delete/drop commands ! Deleted data cannot be recovered!

```
DELETE FROM table1 / TRUNCATE table1        --Deletes all Records
DELETE FROM table1 WHERE condition          --Deletes records based on a
condition
DROP TABLE table                            --Deletes table
DROP DATABASE                               --Deleting database
ALTER TABLE table_name DROP column_name;    --Deletes a column
```

Find and Remove Unused Indexes

A general rule of thumb is that the more indexes you have on a table, the slower INSERT, UPDATE, and DELETE operations will be and more disk space will be consumed. It is essential to track down unused indexes that consume disk space and slow down your database.

By default, statistics are not collected. This is to ensure that statistics collection does not cause any extra load on the server unless desired.

To enable statistics dynamically execute the following command :

```
SET GLOBAL userstat=1;
```

Now, every query to the database updates the statistic tables. The TABLE_STATISTICS and INDEX_STATISTICS are the most interesting tables. The first table shows the number of rows reads from the table and the number of rows changed in the table. The second table shows statistics on index usage.

```
SELECT * FROM INFORMATION_SCHEMA.TABLE_STATISTICS;
+--------------+------------------------+----------+--------------+--------
----------------+
| TABLE_SCHEMA | TABLE_NAME             | ROWS_READ | ROWS_CHANGED |
ROWS_CHANGED_X_INDEXES |
+--------------+------------------------+----------+--------------+--------
----------------+
| hope         | eth_table1             |      1004 |            0 |
0 |
| hope         | eth_table2             |  14343683 |            0 |
0 |
| hope         | eth_table3             |      1002 |            0 |
0 |
+--------------+------------------------+----------+--------------+--------
----------------+
```

```
SELECT * FROM INDEX_STATISTICS;
+--------------+-------------------+------------+-----------+
| TABLE_SCHEMA | TABLE_NAME        | INDEX_NAME | ROWS_READ |
+--------------+-------------------+------------+-----------+
| hope         | eth_table1        | PRIMARY    |         2 |
| hope         | eth_table2        | PRIMARY    |         4 |
+--------------+-------------------+------------+-----------+
```

With the help of the new tables, you can find all unused indexes in a single query.

```
SELECT DISTINCT s.TABLE_SCHEMA, s.TABLE_NAME, s.INDEX_NAME
FROM information_schema.statistics `s` LEFT JOIN
information_schema.index_statistics INDXS
ON (s.TABLE_SCHEMA = INDXS.TABLE_SCHEMA AND
   s.TABLE_NAME=INDXS.TABLE_NAME AND
   s.INDEX_NAME=INDXS.INDEX_NAME)
WHERE INDXS.TABLE_SCHEMA IS NULL;
```

Finally, to delete index run this command in `mysql` client :

```
DROP INDEX index_name ON table_name;
```

Do not forget to set `userstat=0` when statistics are not required anymore.

Shrink and Optimize MySQL

In general MySQL InnoDB does not release disk space after deleting data rows from the table. It keeps the space to reuse it later.

`OPTIMIZE TABLE` reorganizes the physical storage of table data and associated index data, to reduce storage space and improve I/O efficiency when accessing the table. The exact changes made to each table depending on the storage engine used by that table.

Optimization is available only when `innodb_file_per_table` is enabled. Check your configuration like this :

```
MariaDB [(none)]> show variables like "innodb_file_per_table";
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| innodb_file_per_table | ON    |
+-----------------------+-------+
```

You can use `OPTIMIZE TABLE` to reclaim the unused space and to defragment the data file.

```
OPTIMIZE table MyTable;
+-------------------+----------+----------+----------+
| Table             | Op       | Msg_type | Msg_text |
+-------------------+----------+----------+----------+
| testDB.MyTable    | optimize | status   | OK       |
+-------------------+----------+----------+----------+
```

When innodb_file_per_table is OFF, then all data is going to be stored in ibdata files. If you drop some tables and delete some data, then there are two ways to reclaim that unused disk:

- Completely dump the database, then drop it and finally reload it
- Change the storage engine and revert it back to your previous configuration. This will definitively recreate the table and indexes from the start and unused disk space will be reclaimed.

```
ALTER TABLE my_table ENGINE = MyISAM;
ALTER TABLE my_table ENGINE = InnoDB;
```

Whenever you run OPTIMIZE or ALTER TABLE , MySQL will create a new data file until the operation is finished. Do not forget to have enough available space for the operations to finish successfully!

If you want to optimize a 10 GB Table, ensure that there are more than 10GB of free disk space.