Table Optimization

Table optimization is necessary to help certain tables run more efficiently. Optimization happens automatically, but you might need to adjust the schedule to make sure it does not conflict with users working in the system.

Two types of tables are usually optimized:

- Internal reverse index tables used to support full-text search
- Certain commonly-used system tables

Optimizing Index Tables

The optimization settings for the first type of table are located in the corresponding project, and calculated using the Revindex Optimization Period variable for the knowledgebase with the KB timezone. The optimization of these tables does not affect the work of other projects and does not block the work of users. The only inconvenience is a temporary performance reduction.

By default, index table optimization requests are placed in the queue to run every second Saturday night at 2 a.m. in the knowledgebase timezone.

Optimizing System Tables

For system tables the Revindex Optimization Period variable in the admin console is used to set the start time. For instance, the default value is "22-05", meaning that optimization can start between 10 p.m. and 5 a.m. System table optimization runs every fourth Saturday.

Additionally, the system must be in an idle state, which the system defines as whether there has been any user activity in the past 5 minutes. If there was no idle time during this entire period, the optimization is postponed till the next optimization range. This kind of optimization blocks the work of users and therefore should preferably be done at maintenance server time.

Example

If a table is locked for optimization, the log shows a message like this:

```
server.log.2018-11-20-07:2018-11-20 07:13:20,988 WARN   [earch.utils.
FTSSQLMonitor] Long-term query (2812 sec): optimize table swhistorylog
```

During this time any other processes, such as record deletion, are locked. If the process takes longer than the default timeout value of 150 seconds, it fails with a LockWaitException. If it can get access to the system resources before the timeout, the process can be performed correctly.

The timeout value can be modified by setting the innodb_lock_wait_timeout variable in MySQL: https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html#sysvar_innodb_lock_wait_timeout.

Disable Optimization

To turn table optimization off, set the Revindex Optimization Period value to something like "0-0".

Alternately, you can create a file in the tmp directory of AL_HOME, with the name `do_not_start_optimize`:

```
sudo -u enterprisewizard touch /opt/server/Agiloft/tmp/do_not_start_optimize
```

Disabling optimization means that the system doesn't initiate any new optimizations, but any optimizations that are currently running can be completed.

Manual Optimization

If you have disabled optimization, or if you need to run optimization manually at any time, there are two possible methods:

- In AL_HOME/bin, the optimize_tables utility can perform table optimization.
- The following SQL stored procedure runs on all tables under 5gb in size, which should be safe from the MYSQL lock timeout and offers more flexibility in terms of customization than the utility:

```
delimiter //
drop procedure if exists sp2//
CREATE PROCEDURE sp2 ()
BEGIN
  DECLARE myout TEXT default '';
  DECLARE newname VARCHAR(250);
  DECLARE done TINYINT DEFAULT 0;
  -- this will optimize ALL tables at the AL dictionary
  --DECLARE cur1 CURSOR FOR SELECT distinct dbname FROM swtable order by dbname;


  -- this will optimize first 1000 tables with lot of free space and data size less
then 5GB
  DECLARE cur1 CURSOR FOR SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE
TABLE_SCHEMA = 'sw2_std' and (data_length + index_length) < 500000000 order by round
(data_free / (data_free + data_length + index_length) * 100) desc limit 1000;


  -- this will optimize first 100 space waste tables with data size less then 5GB.
  --DECLARE cur1 CURSOR FOR SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE
TABLE_SCHEMA = 'sw2_std' and (data_length + index_length) < 500000000 order by
data_free desc limit 100;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
  DECLARE CONTINUE HANDLER FOR 1146 SET myout := concat(myout, ' ', newname);
  DECLARE CONTINUE HANDLER FOR 1243 select 1;

  OPEN cur1;
  read_loop: LOOP
    FETCH FROM cur1 INTO newname;
    IF done THEN LEAVE read_loop; END IF;
    SET @vsql := CONCAT('optimize table ', newname);
--    select @vsql;
    Prepare stmt FROM @vsql;
    Execute stmt;
    DEALLOCATE PREPARE stmt;
  END LOOP;
  CLOSE cur1;
  select myout;
END; //

call sp2 //
delimiter ;
```