



Toska SQL Tuning Expert for MySQL

It is not just another SQL tuning tool

There are not many SQL tuning tools for MySQL database, but most of them are focused on plan visualization or query plan analysis, it is not helpful if you don't have in-depth SQL tuning knowledge and are not willing to spend extra effort to tune a SQL apart from your daily duties. If you are eager for getting one-button-solution tool that can tune a SQL statement automatically without the need of your intervention, then Toska SQL Tuning Expert for MySQL may be your only choice.

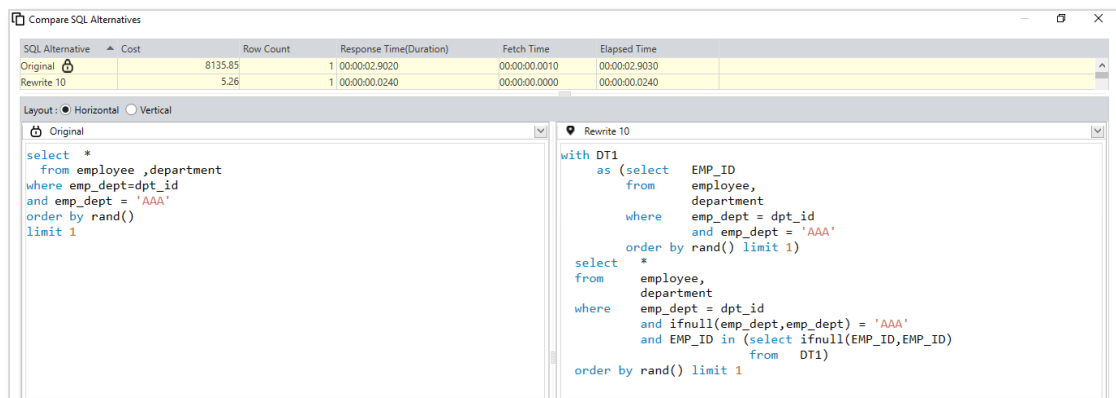
Expensive SQL tuning effort by human expert

It may be up to days or weeks for a DBA or experienced SQL developer to tune a problematic SQL statement. SQL tuning is a very skillful job that not many developers are able to carry out in an enterprise. Should such expensive and valuable time be saved and used for other even more productive tasks inside a company?

World's Leading A.I. SQL Rewrite technology for MySQL database

Manually rewriting SQL syntax to tune a SQL statement is commonly adopted by experienced developers and DBAs, but the effort and cost to rewrite a SQL statement is quite expensive. It requires an in-depth SQL tuning knowledge developer to spend hours or even days to rewrite and test a problematic SQL statement, and the tuning result might not be the best among most of potential SQL rewrites due the limitation of manual effort.

Toska A.I. SQL Rewrite Engine is the world's leading machine SQL Rewrite technology for MySQL database. It not only saves developer's time to rewrite a problematic SQL statement to make it faster, but it also explores the potential best SQL syntax that even an experienced developer cannot discover. Moreover, Toska A.I. rewritten SQL syntax can work with our Hints-injection algorithm to further explore more potential query plans that even most experienced developers cannot achieve.



SQL Alternative	Cost	Row Count	Response Time(Duration)	Fetch Time	Elapsed Time
Original	8135.85	1	00:00:02.9020	00:00:00.0010	00:00:02.9030
Rewrite 10	5.26	1	00:00:00.0240	00:00:00.0000	00:00:00.0240

Layout	Original	Rewrite 10
Horizontal	<pre>select * from employee ,department where emp_dept=dpt_id and emp_dept = 'AAA' order by rand() limit 1</pre>	<pre>with DT1 as (select EMP_ID from employee, department where emp_dept = dpt_id and emp_dept = 'AAA' order by rand() limit 1) select * from employee, department where emp_dept = dpt_id and ifnull(emp_dept,emp_dept) = 'AAA' and EMP_ID in (select ifnull(EMP_ID,EMP_ID) from DT1) order by rand() limit 1</pre>

A 100 times faster SQL rewrite is generated on the right hand side



On-the-job SQL training for in-house SQL developers

An SQL Query is used to retrieve data from your database. However, there may be multiple SQL queries that generate the same results but with different levels of efficiency. An inefficient query can drain the database resources, reduce the database speed or even result in a loss of service for other users. So it is very important to improve the in-house SQL developers with solid SQL writing skills. A short term SQL training course may be helpful, but it is not sustainable to resolve all daily SQL writing challenges. Tosska SQL Tuning Expert inbuilt SQL Rewrite Engine can help developers to correct minor mistakes and generate many semantically-equivalent SQL rewrites for user to compare with their original SQL syntax. Users can observe the SQL syntax, query plans, data distribution and SQL performance of those semantically-equivalent SQL to select the best SQL syntax to replace their original SQL statement. With the on-the-job SQL training process, it not only guarantees the good SQL quality produced during development cycle, but it also improves the in-house developers' SQL writing skills through the process.

The screenshot shows the SQL Editor on the left with a query: `select * from employee ,department where emp_dept=dpt_id and emp_dept = 'AAA' order by rand() limit 1`. The SQL Scenarios window on the right displays a table of generated alternatives.

SQL Alternative	Cost	Row Count	Status	Elapsed Time	Response Time(Duration)
Original	8,135.85	1	✓	00:00:02.9030	00:00:02.9020
Rewrite 1	8,135.85	1	✗ Test run was terminated by...	>00:00:01.0240	
Rewrite 2	4.72	1	✓	00:00:00.0280	00:00:00.0280
Rewrite 3	1,380,959.12	1	✗ Test run was terminated by...	>00:00:01.0340	
Rewrite 4	8,139.67	1	✓	00:00:00.1050	00:00:00.1050
Rewrite 5	4.72	1	✓	00:00:00.0340	00:00:00.0340
Rewrite 6	5.26	1	✓	00:00:00.0670	00:00:00.0670
Rewrite 7	5.51	1	✗ Test run was terminated by...	>00:00:01.0240	
Rewrite 8	8,162.00	1	✗ Test run was terminated by...	>00:00:01.0240	
Rewrite 9	1,381,152.73	1	✗ Test run was terminated by...	>00:00:01.0240	

Generate semantically equivalent SQL alternatives with "Rewrite Only" Tuning Method

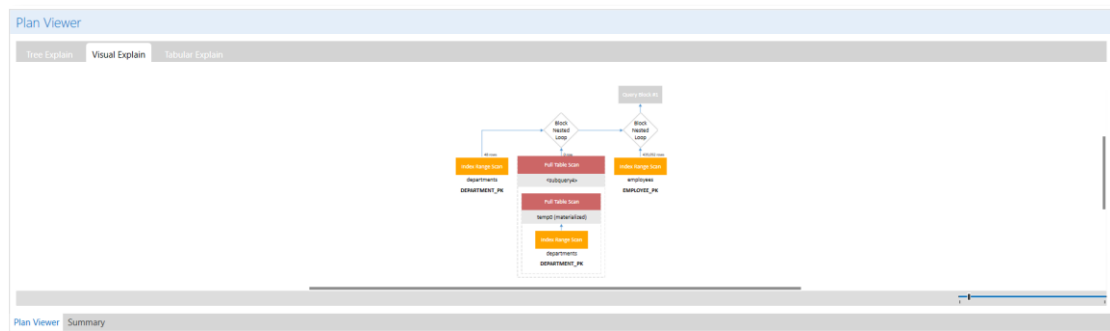
Tosska proprietary Tree Plan format for your easy understanding

Traditional MySQL tabular explain plan and visual explain plan are the standard tools for developer to understand how MySQL SQL optimizer is processing the input SQL statement, while the tabular explain plan lacking in hierarchical operation steps, visual explain plan is too focused on hierarchical structure without enough information displaying on screen. There is another fatal problem that a complex visual explain plan is normally too big to fit in a window, it is very difficult for the user to comprehend the overall structure of the query plan without scrolling around the window.

The screenshot shows the Plan Viewer window with the 'Tabular Explain' tab selected. It displays a table with columns: ID, Select Type, Table, Partitions, type, Possible Keys, Key, Key Length, Ref, Rows, Filtered, Extra.

ID	Select Type	Table	Partitions	type	Possible Keys	Key	Key Length	Ref	Rows	Filtered	Extra
1	PRIMARY	departments		range	DEPARTMENT_PK	DEPARTMENT_PK	30		48	100	Using where; Using inc
1	PRIMARY	<subquery4>		ALL					0	100	Using join buffer (Bloc
1	PRIMARY	employees		range	EMPLOYEE_PK	EMPLOYEE_PK	4		439092	1	Using where; Using joi
4	MATERIALIZED	<derived5>		ALL					48	33.33	Using where
5	DERIVED	departments		range	DEPARTMENT_PK	DEPARTMENT_PK	30		48	100	Using where; Using inc

Traditional Tabular Plan



Visual Plan

Tosska proprietary tree plan has a market-leading Explain Plan function for MySQL, it not only has rich statistical information like what is provided by Tubular Explain from MySQL, but it also has a hierarchical structure like what is displaying in the visual plan. The beauty is that all such information can be displayed in a small window for easy reading.

Plan Operation	Object	Object Key	Query Cost	Read Cost	Eval Cost	Prefix Cost	Rows	Data Read	Sort Cost
Query Block #1			68327594.16						
Block Nested Loop									
Block Nested Loop									
Index Range Scan	departments	DEPARTMENT_PK	5.63			10.43	48	25K	
Full Table Scan	<subquery4>						0		
Full Table Scan	temp0 (materialized)		6.30			7.90	48	639	
Index Range Scan	departments	DEPARTMENT_PK	5.63			10.43	48	25K	
Index Range Scan	employees	EMPLOYEE_PK	3468603.63			68327594.16	439092	479M	

Tosska Tree Plan

What is machine tuning for SQL statements?

Tosska SQL Tuning Expert is a SQL tuning tool that optimizes your SQL statements without the need of user's involvement. The product will give you the ultimate SQL performance solution by just point and click. What you have to do is to input your problematic SQL statement into the product and press a button. You don't have to do analysis, guessing or manual testing during the entire SQL tuning process. The improved SQL statement will be benchmarked with your original SQL statement side by side without doubt.

Summary			
Explore SQL Alternatives Summary			
17 distinguished SQLs are found after investigation of 1000 SQL rewrites			
Test Run Summary			
Best SQL Alternative Found: Rewrite 10			
Name	Elapsed Time	Improvement	SQL Running Times
Original	00:00:02.9030	N/A	Run the SQL 2 times
Rewrite 10	00:00:00.0240	99.17%	Run the SQL 2 times
Best SQL Alternative Criteria: Elapsed Time			

SQL Tuning Summary



The machine tuning for SQL statements is a proprietary technology invented by Tosska to mimic a human expert SQL tuning process, in which the engine tries every possible effective SQL rewrite and MySQL Hints combinations for a SQL statement to improve the execution speed within the given quota. As the permutation of SQL rewrites and Hints combinations to a SQL statement is so huge, it is impossible for a human expert to accomplish it for complex SQL statements within a short time. Furthermore, there is also no way for a DBA or developer to guarantee that the best solution is found after a lot of manual trials and errors.

Best solution without trial and error

With Tosska SQL Tuning Expert for MySQL, you no longer need to rewrite or try every possible Hints combination manually for a problematic SQL statement to explore potential better performance execution plans, since all those hard tasks are released by our embedded AI engine. Our intelligent engine will help you to find every possible rewrite and Hints combination to improve your SQL speed without the need of your intervention. You just sit back, relax, and wait for the best SQL alternative to come up on your screen.

The screenshot displays the Tosska SQL Tuning Expert for MySQL 2.0.0 application. The interface includes a menu bar (Tune, Explain Plan, Explore Alternatives, Test Run, Compare SQL, Abort All, Clear, Save), a toolbar, and a status bar. The main area is divided into three panes:

- SQL Editor:** Contains a SQL query:

```
with DT1
as (select EMP_ID
from employee,
department
where emp_dept = dpt_id
and emp_dept = 'AAA'
order by rand() limit 1)
select *
from employee,
department
emp_dept = dpt_id
and emp_dept = 'AAA'
and EMP_ID in (select EMP_ID
from DT1)
order by rand() limit 1
```
- SQL Scenarios:** A table listing various SQL alternatives and their performance metrics.
- Summary:** A section providing a test run summary and the best alternative found.

SQL Alternative	Cost	Row Count	Status	Elapsed Time	Response Time
Original (H3)	348,634.80		Test run was terminated by...	>00:00:01.0840	
Original (H4)	327,339.30		Test run was terminated by...	>00:00:01.0840	
Original (H5)	8,828.79		Test run was terminated by...	>00:00:01.0840	
Original (H6)	349,371.22		Test run was terminated by...	>00:00:01.0840	
Original (H7)	405,967.41		Test run was terminated by...	>00:00:01.0840	
Original (H8)	40,584.20		Test run was terminated by...	>00:00:01.0840	
Original (H9)	324,547.28		Test run was terminated by...	>00:00:01.0840	
Original (H10)	380,780.75		Test run was terminated by...	>00:00:01.0840	
Original (H11)	40,238.33		Test run was terminated by...	>00:00:01.0840	
Rewrite 1	8,162.00		Test run was terminated by...	>00:00:01.0840	
Rewrite 1 (H1)	8,205.10		Test run was terminated by...	>00:00:01.0840	
Rewrite 2	4.72	1	Test run was terminated by...	00:00:00.0840	00:00:00.0840
Rewrite 2 (H1)	8,907.71		Test run was terminated by...	>00:00:01.0840	

Summary

Test Run Summary

Best SQL Alternative Found: Rewrite 2

Name	Elapsed Time	Improvement	SQL Running Times
Original	00:00:03.3600	N/A	Run the SQL 2 times
Rewrite 2	00:00:00.0840	97.50%	Run the SQL 2 times

Best SQL Alternative Criteria: Elapsed Time

Plan Viewer: Summary

MySQL80 (tosska@192.168.0.101)

Best alternative found after automatic SQL tuning



Provide even better than a human expert's solution

Tuning SQL is a time-consuming job that requires in-depth knowledge on SQL tuning skill and most SQL developers are not trained to accomplish this job apart from their daily development tasks. Furthermore, there is no way for a DBA to explore all alternative execution plans within a short time. The following screenshot shows that Toska SQL Tuning Expert can explore thousands of SQL alternatives in just a few minutes that may require a human expert months' effort to do so.

SQL Editor

```
select emp_name,
       dpt_name,
       grd_desc
from   employee,
       department DEPARTMENT1,
       grade
where  emp_grade = grd_id
and    emp_dept = dpt_id
and    EXISTS (SELECT 'X'
              from department DEPARTMENT2 |
              WHERE dpt_avg_salary in (select min(dpt_avg_salary)
                                       from department DEPARTMENT3)
              AND dpt_id = EMPLOYEE.emp_dept)
```

SQL Scenarios

SQL Alternative	Cost	Row Count	Status	Elapsed Time
Original	5,658,296.04	7500	✓	00:01:02.5000
Original (H1)	5,658,296.04			
Original (H2)	7,500,241.38			
Original (H3)	13,345,301.21			
Original (H4)	7,500,241.38			
Original (H5)	5,658,296.04			
Original (H6)	3,291,622,038.51			
Rewrite 1	5,658,296.04			
Rewrite 2	5,658,296.04			
Rewrite 2 (H1)	5,658,296.04			
Rewrite 2 (H2)	7,500,241.38			

Summary

Explore SQL Alternatives Summary

163 distinguished SQLs are found after investigation of 6516 SQL rewrites and hints applications

Exploring SQL alternatives: 90.22%

Automatic SQL tuning in process

After benchmarking partial or all SQL alternative execution plans, the best SQL alternative will be displayed with Original SQL statement side by side on the screen. The process is fully automatic without the need of user's involvement and the result is the best out of thousands of potential execution plans that MySQL can generate for the SQL statement. This exhaustive search and test process is impossible to be accomplished even by a human expert.

SQL Editor

```
with DT1
as (select min(dpt_avg_salary) col1
     from department DEPARTMENT3
     where ifnull(DEPARTMENT3.DPT_ID,DEPARTMENT3.DPT_ID) >= '')
select /*+ QB_NAME(QB1) BKA('employee @QB1) */ emp_name,
       dpt_name,
       grd_desc
from   employee,
       department DEPARTMENT1,
       grade
where  emp_grade = grd_id
and    emp_dept = dpt_id
and    exists ( select 'X'
              from department DEPARTMENT2
              where dpt_avg_salary in (select col1
                                       from DT1)
              and dpt_id = EMPLOYEE.emp_dept)
```

SQL Scenarios

SQL Alternative	Cost	Row Count	Status	Elapsed Time
Rewrite 4 (H5)	5,658,296.04		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 4 (H6)	3,291,622,038.51		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 4 (H7)	5,658,296.04		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 5	6,468,564.49		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 5 (H1)	6,468,564.49		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 5 (H2)	6,475,901.67		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 5 (H3)	6,497,323.42	7500	✓	00:00:03.4351
Rewrite 5 (H4)	4,920,868.74		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 5 (H5)	4,920,868.74		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 5 (H6)	1,295,541,989.91		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 5 (H7)	3,261,842,014.62		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 5 (H8)	4,920,868.74		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 5 (H9)	3,291,435,992.62		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 5 (H10)	4,920,868.74		✗ Test run was terminated by...	>00:00:04.0500
Rewrite 6	5,658,296.04		✗ Test run was terminated by...	>00:00:04.0500

Best syntax rewrite with hints injected is found after benchmarking

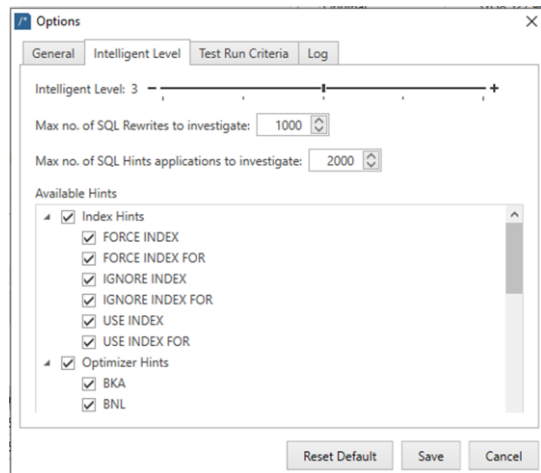


Uncompromised SQL tuning solution within manageable time

It is known that the longer you spend on the SQL tuning process, the more chance you may get a better SQL alternative. It is also true in Tosska SQL Tuning Expert that the user can adjust the Intelligent Level to control the time spent on a specific SQL statement according to the complexity of your SQL statement. For complex SQL statements with huge potential execution plans, users can allocate more resources and time to explore the ultimate performance solution for the SQL.

The predefined intelligent level is 5 sets of “Max no. of SQL Rewrites to investigate” and “Max no. of SQL Hints applications to investigate” setting to control the size of search space. The larger the search space, the more chance the engine can find a better SQL solution for a problematic SQL statement.

Furthermore, individual user-defined “Max no. of SQL Rewrites to investigate” and “Max no. of SQL Hints applications to investigate” options are also available for experienced users to tackle complicated SQL performance



Intelligent Level option