Discussion-7 database tuning

Discussion Topic:

Explain the purpose of database tuning. List at least three MySQL tools or commands you can use to tune your MySQL database. Provide an example for each tool.

My Post:

Hello Class,

Three tools or commands that are used to tune your MySQL database are the EXPLAIN command, MySQLTuner, and Performance Schema.

The EXPLAIN Command

The EXPLAIN command is used within SQL databases to provide information about how a specific query is executed (Bonic, 2024). In MySQL, EXPLAIN works with SELECT, DELETE, INSERT, REPLACE, and UPDATE statements (MySQL, n.d.a). The EXPLAIN command, instead of executing a query and returning the query data results, it outputs the statement execution plan. The plan describes the sequence of operations the database will perform, how tables will be accessed and joined, and which indexes, if any, will be utilized. This information helps identify inefficiencies and bottlenecks within a query, allowing the database administrator to make informed decisions to optimize the query performance.

For example, the query below fetches the first name, last name, and email address of customers, as well as their order dates, and the names, quantities, and prices of the products they ordered. Then, it filters customers with a Yahoo address.

Figure 1

EXPLAIN Command Example

```
1  Explain SELECT
2      c.first_name,
3      c.last_name,
4      o.order_date,
5      p.product_name,
6      oi.quantity,
7      oi.item_price,
8      c.email_address
9  FROM
10     customers c
11 JOIN
12     orders o ON c.customer_id = o.customer_id
13 JOIN
14     order_items oi ON o.order_id = oi.order_id
15 JOIN
16     products p ON oi.product_id = p.product_id
17 WHERE
18     c.email_address LIKE '%@yahoo.com';
19
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	c	NULL	ALL	PRIMARY	NULL	NULL	NULL	8	12.50	Using where
1	SIMPLE	o	NULL	ref	PRIMARY,orders_fk_customers	orders_fk_customers	4	copy_my_guitar_shop.c.customer_id	1	100.00	NULL
1	SIMPLE	oi	NULL	ref	items_fk_orders,items_fk_products	items_fk_orders	4	copy_my_guitar_shop.o.order_id	1	100.00	NULL
1	SIMPLE	p	NULL	eq_ref	PRIMARY	PRIMARY	4	copy_my_guitar_shop.oi.product_id	1	100.00	NULL

Note: The figure showcases the output and use of the EXPLAIN command on a query filtering email addresses containing “yahoo.com” from a table, which is the result of several joined tables.

Table 1

Explanation of the EXPLAIN Table Output

Column	Description	Tuning Use Case
id	ID of the `selected type` within the query.	Helps understand the order of execution in complex queries with subqueries or unions.
select_type	The type of `SELECT` query, for example: `SIMPLE`, `PRIMARY`, `SUBQUERY`, `DERIVED`, `UNION`.	Indicates the role of the `selected type` in the query.
table	The table which the data was fetched from	Identifies the specific table being accessed.
type	The join type used, indicates how MySQL accesses the table.	Identify how the table is scanned/filtered/joined. For example, `ALL` (full table scan, not good for performance usually), `system`, `const`, `eq_ref`, `ref`, or `range`.
possible_keys	Shows alternative indexes that MySQL can use to find rows in the table.	If `NULL`, no suitable indexes were found. If indexes are listed but not used (see key), you may want to investigate if they are a better alternative.
key	The actual index MySQL used.	If `NULL`, no index is used, often a sign of poor performance.
key_len	The length of the key, in bytes, that MySQL used	Shorter key lengths are generally preferable.
ref	Shows which column or constant value is compared to the value in the key column.	Helps understand how the index is being utilized for Joining or filtering (for example).
rows	Estimate of the number of rows that MySQL needs to examine to execute this specific part of the query.	A high number, especially in conjunction with type: ALL, indicates inefficiency.
filtered	An estimated percentage of table rows that will be filtered by the query..	A low percentage usually indicates that the `WHERE` clause, for example, is not very selective. In other words, it is filtering a high number of rows that are not meeting the condition stated. (can be optimized)
Extra	Contains additional information about how MySQL performs the query.	Often, it is a sign of inefficiencies, as it might indicate operations like “Using filesort” (MySQL makes extra passes to sort rows).

Note: The table provides a description of the columns output by the EXPLAIN command. It also provides tuning recommendations and information.

Based on the information provided in Table 1, the EXPLAIN table output shows that in the following row

id	table	type	key	rows	filtered	Extra
1	**c** (customers)	**ALL**	NULL	8	12.50 %	Using where

the query could benefit from optimization as the WHERE clause condition. For example, makes the query read all 8 customer rows and discard ~87.5 % (filtered 12.5 %). This works ok in a small database; however, in a large database, it would be very inefficient. The wildcard (%) in the LIKE “%yahoo.com“ statement prevents B-tree index sorting from being used on the email_address column. A fix would be to generate a column that stores the e-mail domain and add an index (indexing the email_address based on email domains):

```
ALTER TABLE customers
  ADD COLUMN email_domain VARCHAR(50)
    AS (SUBSTRING_INDEX(email_address,'@',-1)) STORED,
  ADD INDEX idx_email_domain (email_domain);
```

Then the query can be modified as so

```
... WHERE email_domain = 'yahoo.com';
```

MySQLTuner

Another method for optimizing a database is to use such tools as MySQLTuner. MySQLTuner is an open-source Perl script that provides insights and recommendations for optimizing MySQL (James, 2024). To run MySQLTuner, you first need to install it, see MySQLTuner-perl GitHub. Once installed, you need to connect to your database using the command:

```
perl mysqltuner.pl --host your_mysql_host --user your_mysql_user --pass
your_mysql_password
```

Then you can run the following command:

```
perl mysqltuner.pl
```

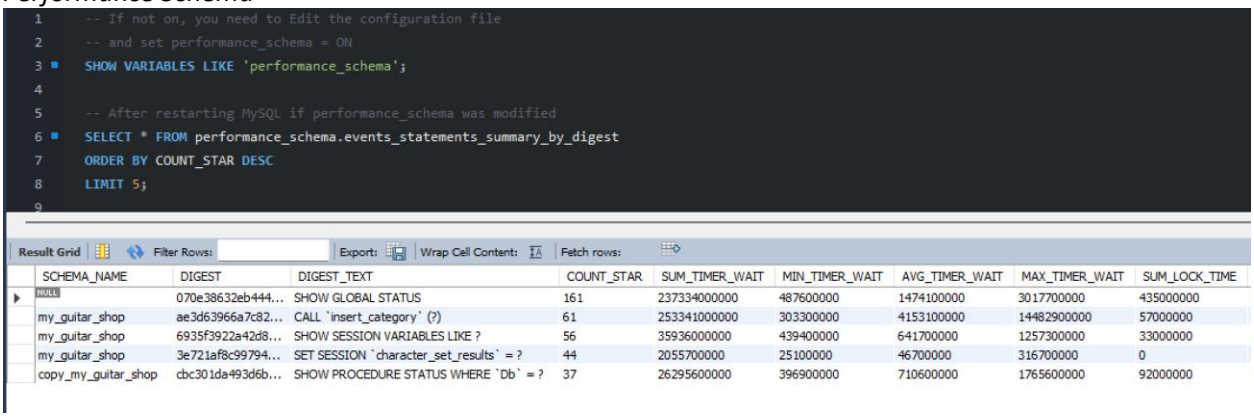
The command above will execute the script and analyze the MySQL server's performance, and provide a report with recommendations for optimization.

Performance Schema

Performance Schema is a tool for monitoring MySQL Server execution at a low level (MySQL, n.d.b) Performance Schema provides a way to inspect the server's internal execution at runtime, monitor server events, and more. To use Performance Schema, it needs to be enabled, and you can run a query such as the one below that collects performance data.
This query retrieves the top five most frequently used queries. This query can identify statements that may need optimization by collecting data from it:

Figure 2

Performance Schema



```
1   -- If not on, you need to Edit the configuration file
2   -- and set performance_schema = ON
3 * SHOW VARIABLES LIKE 'performance_schema';
4
5   -- After restarting MySQL if performance_schema was modified
6 * SELECT * FROM performance_schema.events_statements_summary_by_digest
7   ORDER BY COUNT_STAR DESC
8   LIMIT 5;
9
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

SCHEMA_NAME	DIGEST	DIGEST_TEXT	COUNT_STAR	SUM_TIMER_WAIT	MIN_TIMER_WAIT	AVG_TIMER_WAIT	MAX_TIMER_WAIT	SUM_LOCK_TIME
NULL	070e38632eb444...	SHOW GLOBAL STATUS	161	237334000000	487600000	1474100000	3017700000	435000000
my_guitar_shop	ae3d63966a7c82...	CALL `insert_category` (?)	61	253341000000	303300000	4153100000	14482900000	57000000
my_guitar_shop	6935f3922a42d8...	SHOW SESSION VARIABLES LIKE ?	56	35936000000	439400000	641700000	1257300000	33000000
my_guitar_shop	3e721af8c99794...	SET SESSION `character_set_results` = ?	44	2055700000	25100000	46700000	316700000	0
copy_my_guitar_shop	cbc301da493d6b...	SHOW PROCEDURE STATUS WHERE `Db` = ?	37	26295600000	396900000	710600000	1765600000	92000000

Note that the times are in 1 picosecond = 10^{-12} seconds. As shown in the output, only CALL insert_category() (row 2) may need query tuning; with AVG_TIMER_WAIT => 4 151 300 000 ps ≈ 4.15 ms per call does seem much, but it may be possible to optimize it and get better performance.

-Alex

References:

Bonic, S. (2024, November 25.). *Understanding MySQL queries with explain*. Exoscale. https://www.exoscale.com/syslog/explaining-mysql-queries/

James, H. (2024, January 2024). *MySQL performance tuning: Tips, scripts and tools*. Linus Blog. https://linuxblog.io/mysql-performance-tuning-tips-scripts-tools/

MySQL (n.d.a). 15.8.2 EXPLAIN statement. *MySQL 9.2 reference manual*. MySQl. https://dev.mysql.com/doc/refman/9.2/en/explain.html#explain-table-structure

MySQL (n.d.b). Chapter 29 MySQL performance schema. *MySQL 8.4 reference manual.* https://dev.mysql.com/doc/refman/8.4/en/performance-schema.html