PingCAP TiDB

TiDB Database Auditing User Guide

2023/05/31

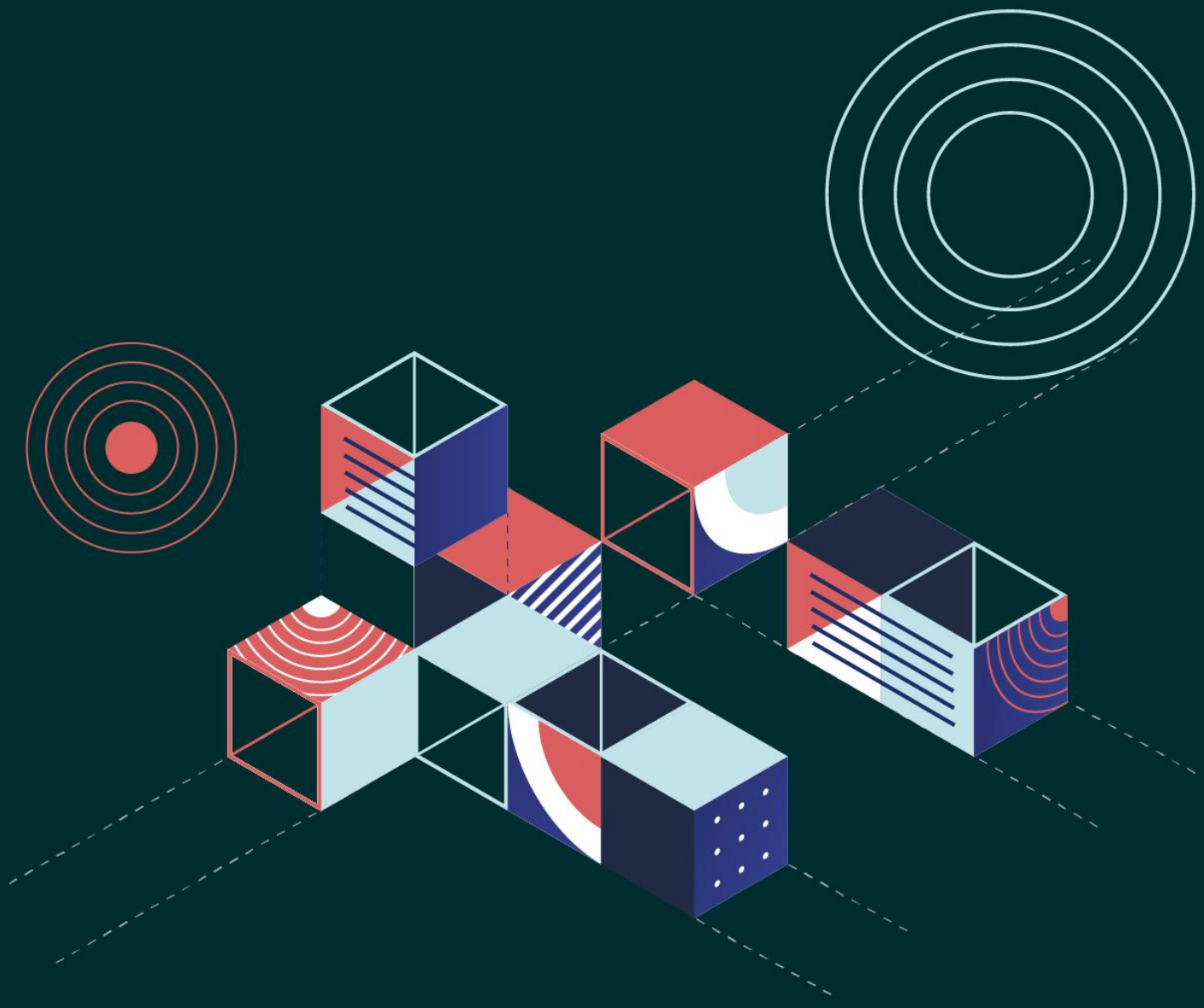

Table of Contents

Introduction **1**
Differences between database auditing and the audit plugin **1**
Obtain the database auditing feature **2**
The range of database auditing **2**
The events of database auditing **3**
Recorded information in audit logs **5**
General information 5
SQL statement information 6
Connection information 6
Audit operation information 7
Audit log filters and rules **7**
Filters 8
Filter rules 9
File formats of audit log **10**
Rotation of audit log **11**
The number and duration for reserving audit logs **11**
Audit log redaction **11**
System tables **11**
mysql.audit_log_filters 11
mysql.audit_log_filter_rules 12
System variables **13**
tidb_audit_enabled 13
tidb_audit_log 14
tidb_audit_log_format 14
tidb_audit_log_max_filesize 14
tidb_audit_log_max_lifetime 15
tidb_audit_log_reserved_backups 15
tidb_audit_log_reserved_days 15
tidb_audit_log_redacted 16
Functions **16**
audit_log_rotate 16
audit_log_create_filter 16
Examples 17
audit_log_remove_filter 17
audit_log_create_rule 18
Examples 18
audit_log_remove_rule 19
audit_log_enable_rule 19

audit_log_disable_rule 19
Dynamic privileges **20**
AUDIT_ADMIN 20
RESTRICTED_AUDIT_ADMIN 20
Migration from audit plugin to database auditing **20**

Introduction

Database auditing is an important feature in TiDB Enterprise Edition, which provides a powerful monitoring and auditing tool for enterprises. Auditing can record information such as query statements, transaction operations, and user login events. This information can be stored in the audit log for future queries and analysis, thus ensuring the data security and compliance for enterprises. The database auditing feature can help managers in enterprises track the source and impact of database operations to ensure that data would not be illegally stolen or tampered with. At the same time, database auditing can also help enterprises to comply with various regulatory and compliance requirements to safeguard the legal and ethical compliance of enterprises.

A redesigned database auditing feature (hereinafter referred to as "database auditing") is released in TiDB v7.1.0, replacing the legacy audit plugin (hereinafter referred to as "audit plugin").

This article describes how to use Database Auditing in TiDB.

Differences between database auditing and the audit plugin

	Database Auditing	**Audit Plugin**
Turn on/off auditing	`set global tidb_audit_enabled = 1;` `set global tidb_audit_enabled = 0;`	`admin plugins enable audit;` `admin plugins disable audit;`
The number of audit event classes for one SQL statement	One or more	One
Modify the auditing objects	Create the filter by using `audit_log_create_filter`, then bind the filter to users by using `audit_log_create_rule`	Modify the table `mysql.tidb_audit_table_access`, then run `flush tidb plugins audit` to refresh the configuration
Filter the	Wildcards with JSON	Regular expressions

auditing objects		
Modify the path and format of log	Set path by `tidb_audit_log` and set format by `tidb_audit_log_format`	Not supported
Supported formats of log	Text, JSON	Text
Turn on/off the redaction of audit	Set redaction by variable `tidb_audit_log_redacted`	Set redaction by variable `tidb_audit_redact_log`
Audit log rotation	Support manual rotation, or automatic rotation by reaching specific file size or time	Automatic rotation by reaching specific file size
Limit users from modifying configuration related to audit by dynamic privileges	AUDIT_ADMIN, RESTRICTED_AUDIT_ADMIN	Not supported
Cleanup audit log	Support manual cleanup by `tiup cluster`, or automatic cleanup by setting variables `tidb_audit_log_reserved_backups` and `tidb_audit_log_reserved_days`	Only support manual cleanup: `tiup cluster clean <cluster-name> --audit-log`

Obtain the database auditing feature

Database auditing is a feature available in TiDB Enterprise Edition. Navigate to the TiDB Enterprise page to obtain the TiDB Enterprise Edition. You can get the support services of business experts.

The range of database auditing

All user interface operations are covered in the range of database auditing, including the prepared statement interface. Recursive operations from such operations are not audited.

The following external tool operations are audited:

- Writes from TiCDC

- Writes via TiDB Data Migration (DM)
- Backup operations by the BR commands using the SQL interface

The following operations are not audited:

- Internal operations of the database
- Database operations derived from user operations
- Import operations by TiDB Lightning
- Operations related to TiSpark
- Backup operations initiated by external BR tools
- The `PREPARE` and `EXECUTE` statements in SQL or protocol form (the actual executed statements are audited)
- Operations of starting, stopping, scaling up or scaling down the database.

The events of database auditing

TiDB database auditing classifies SQL operations into several *Event Classes* based on the type of SQL statements. A SQL statement corresponds to one or more event classes. One event class can be a *subclass* of another event class. For example:

- The event classes for `SELECT * FROM t` are `QUERY` and `SELECT`. `SELECT` is a subclass of `QUERY`.
- The event classes for `INSERT INTO t VALUES (1),(2),(3)` are `QUERY`, `QUERY_DML` and `INSERT`. `QUERY_DML` is a subclass of `QUERY`, and `INSERT` is a subclass of `QUERY_DML`.
- The event classes for `SET GLOBAL tidb_audit_enabled = 1` are `AUDIT`, `AUDIT_SET_SYS_VAR` and `AUDIT_ENABLE`. `AUDIT_SET_SYS_VAR` is a subclass of `AUDIT`, and `AUDIT_ENABLE` is a subclass of `AUDIT_SET_SYS_VAR`.

There are several advantages to distinguish between different SQL operations by event classes:

- Different information is recorded according to different event classes of the audit log. For example, the `CONNECTION` events record the IP address and port of clients, while the `SELECT` events record the queried SQL statements and accessed tables.
- Based on event classes, you can select the SQL statements that you care about and filter out unwanted audit logs.

Here is the summary of all event classes in TiDB database auditing:

Event Class	**Description**	**Parent-class**
CONNECTION	Record all operations related to connections, such as handshaking,	-

Event Class	Description	Parent-class
	connections, disconnections, connection reset, and changing users	
CONNECT	Record all operations of the handshaking in connections	CONNECTION
DISCONNECT	Record all operations of the disconnections	CONNECTION
CHANGE_USER	Record all operations of changing users	CONNECTION
QUERY	Record all operations of SQL statements, including all errors about querying and modifying data	-
TRANSACTION	Record all operations related to transactions, such as BEGIN, COMMIT, and ROLLBACK	QUERY
EXECUTE	Record all operations of the EXECUTE statements	QUERY
QUERY_DML	Record all operations of the DML statements, including INSERT, REPLACE, UPDATE, DELETE, and LOAD DATA	QUERY
INSERT	Record all operations of the INSERT statements	QUERY_DML
REPLACE	Record all operations of the REPLACE statements	QUERY_DML
UPDATE	Record all operations of the UPDATE statements	QUERY_DML
DELETE	Record all operations of the DELETE statements	QUERY_DML
LOAD DATA	Record all operations of the LOAD DATA statements	QUERY_DML
SELECT	Record all operations of the SELECT	QUERY

Event Class	Description	Parent-class
	statements	
QUERY_DDL	Record all operations of the DDL statements	QUERY
AUDIT	Record all operations related to setting TiDB database auditing, including setting system variables and calling system functions	-
AUDIT_SET_SYS_VAR	Record all operations of setting system variables related to TiDB database auditing	AUDIT
AUDIT_FUNC_CALL	Record all operations of calling system functions related to TiDB database auditing	AUDIT
AUDIT_ENABLE	Record all operations of enabling TiDB database auditing	AUDIT_SET_SYS_VAR
AUDIT_DISABLE	Record all operations of disabling TiDB database auditing	AUDIT_SET_SYS_VAR

Recorded information in audit logs

General information

All classes of audit logs contain the following information:

Information	Description
ID	The unique identifier that identifies the audit record of an operation
EVENT	The event classes of the audit record. Multiple event types are separated by commas (,).
USER	The username of the audit record
ROLES	The roles of the user at the time of the operation

CONNECTION_ID	The identifier of the the user's connection
TABLES	The accessed tables related to this audit record
STATUS_CODE	The status code of the audit record. 1 means success. 0 means failure.
REASON	The error message of the audit record. Only recorded when an error occurs during the operation.

SQL statement information

When the event class is `QUERY` or a subclass of `QUERY`, the audit logs contain the following information:

Information	**Description**
CURRENT_DB	The name of the current database
SQL_TEXT	Record the executed SQL statements. If the audit log redaction is enabled, the redacted SQL statements are recorded.
EXECUTE_PARAMS	Record the parameters for the EXECUTE statements. The information is recorded only when the event classes include `EXECUTE`, and the audit log redaction is disabled.
AFFECTED_ROWS	Record the number of affected rows of the SQL statements. The information is recorded only when the event classes include `QUERY_DML`.

Connection information

When the event class is `CONNECTION` or a subclass of `CONNECTION`, the audit logs contain the following information:

Information	**Description**
CURRENT_DB	Name of the current database. When the event classes include `DISCONNECT`, this information is not recorded.
CONNECTION_TYPE	Type of connection, including Socket, UnixSocket, and SSL/TLS

PID	Process ID of the current connection
SERVER_VERSION	Current version of the connected TiDB server
SSL_VERSION	Current version of SSL in use
HOST_IP	Current IP address of the connected TiDB server
HOST_PORT	Current port of the connected TiDB server
CLIENT_IP	Current IP address of the client
CLIENT_PORT	Current port of the client

Audit operation information

When the event class is `AUDIT` or a subclass of `AUDIT`, the audit logs contain the following information:

Information	**Description**
AUDIT_OP_TARGET	The objects of the setting related to TiDB database auditing, including system variables (such as `tidb_audit_enabled`) and system functions (such as `audit_log_create_filter`).
AUDIT_OP_ARGS	The arguments of the setting related to TiDB database auditing, for example, setting `tidb_audit_enabled` to `ON` to enable database auditing, and setting the filter name to remove the filter using `audit_log_remove_filter`.

Audit log filters and rules

You can choose the records to be audited in the TiDB database auditing by *filters* and *filter rules*. A filter defines the auditing objects, such as the events to be recorded and the target database names. A filter rule specifies the database users to which the filter applies. One filter can apply to multiple database users to share the same filter definitions. For example, there are several security levels for users in an enterprise. You can create one filter for each security level, and the users of the same security level can share the same filter. When the specification for one security level is updated, you only need to update the corresponding filter.

Filters

A filter filters the auditing objects by the following:

- Event classes: `class` to be included and `class_excl` to be excluded. If neither is declared, events of all classes are audited.
- Tables accessed in auditing events: `table` to be included and `table_excl` to be excluded. If neither is declared, operations of all tables are audited.
- The execution status of audit events: `status_code` to choose the execution result of an auditing event. `1` means successful events and `0` means failed events. If `status_code` is not declared, all successful and failed events are audited.

Filters are defined in JSON in the following format:

```
JavaScript
{
  "name": "<nameOfTheFilter>",
  "filter": [
    {
      "class": ["<stringArray>"],
      "class_excl": ["<stringArray>"],
      "table": ["<stringArray>"],
      "table_excl": ["<stringArray>"],
      "status_code": ["<intArray>"],
    },
    {
      // more filterSpec
    }
  ]
}
```

In the example above:

- `name` is the name of the filter. It is optional and can be empty.
- `filter` consists of several (or zero) `filterSpec`. If any `filterSpec` of a `filter` takes effect, the whole `filter` takes effect.
- `filterSpec` can declare `class`, `class_excl`, `table`, `table_excl` and `status_code`, all of which can be empty. Only if all declarations of one `filterSpec` take effect, the `filterSpec` takes effect.

`{}` is an empty filter, which can audit all `CONNECTION`, `QUERY` and `AUDIT` events.
Here is a filter named `"a"` to audit all failed DDL statements, failed connections, and queries to the `test` database:

```
JavaScript
{
  "name": "a",
  "filter": [
    {
      "class": ["QUERY_DDL", "CONNECTION"],
      "status_code": [0]
    },
    {
      "class": ["SELECT"],
      "table": ["test.*"]
    }
  ]
}
```

In TiDB, filters are created by the function `audit_log_create_filter`, which can be queried in the table `mysql.audit_log_filters`.

Note: `AUDIT` events must be audited and CAN NOT be filtered out by filters.

Filter rules

After creating a filter, you can apply the filter via a filter rule to a user. The filter decides whether to audit the operations of the user. You can conveniently disable or enable the filter by the filter rule.

A filter rule consists of an audit user and a filter name.

- An audit user consists of a username and a user hostname separated by `@`. Same as the definition of database users, the hostname part can use `%` and `_` as wildcards, and the rule that matches the longest username takes effect.
- The filter name must exist in `mysql.audit_log_filters`.

You can create a filter rule by `audit_log_create_rule`, and later search it in the table `mysql.audit_log_filter_rules`.

After you create a filter rule, the related filter takes effect on the specified audit users. You can enable and disable a filter rule by `audit_log_enable_rule` and `audit_log_disable_rule`.

Note:
An audit log record will be recorded in the audit log as long as it satisfies ***any*** of the filter rules. This means that if there are conflicting filter rules for the same record at the same time, the record can still be recorded in the log.
For example, there is a `visit_test` filter defined as `{"filter":[{"table":["test.*"]}]}` to include all operations of accessing the `test` database, and another `not_visit_test` filter defined as `{"filter":[{"table_excl":["test.*"]}]}` to exclude all operations of accessing the `test` database. When the filter rules corresponding to these two filters take effect at the same time, all operations of accessing the `test` database are audited because they meet the `visit_test` rule.

As long as database auditing is enabled, all `AUDIT` events are forced to be audited, and cannot be excluded by any filter rule.

File formats of audit log

TiDB database auditing supports the following file formats: text and JSON. You can set the format by the system variable `tidb_audit_log_format`.

For query `select * from t`, the following is a log example in text:

```
[2023/03/13 16:51:40.174 +08:00] [INFO]
[ID=851e8de3-b016-4384-8440-7386c033ef54-0031]
[EVENT="[QUERY,SELECT]"] [USER=root] [ROLES="[]"]
[CONNECTION_ID=2199023255955] [TABLES="[`test`.`t`]"] [STATUS_CODE=1]
[CURRENT_DB=test] [SQL_TEXT="select * from `t`"]
```

The following is a log example in JSON:

```
{"TIME":"2023/03/13 16:51:30.660
+08:00","ID":"851e8de3-b016-4384-8440-7386c033ef54-002f","EVENT":["QUE
RY","SELECT"],"USER":"root","ROLES":[],"CONNECTION_ID":"2199023255955"
,"TABLES":["`test`.`t`"],"STATUS_CODE":1,"CURRENT_DB":"test","SQL_TEXT
":"select * from `t`"}
```


Rotation of audit log

Log rotation can limit the size of log files. In TiDB, you can use the system variable `tidb_audit_log_max_filesize` to set the size of a single audit log file, and use `tidb_audit_log_max_lifetime` to set the time to trigger automatic audit log rotation. You can set both system variables to trigger a log rotation when either system variable takes effect. You can also use the `audit_log_rotate` function to manually trigger a log rotation.

The number and duration for reserving audit logs

To automate the management of archiving audit logs, TiDB provides the following system variables to control the number and duration of audit logs to be reserved:

- `tidb_audit_log_reserved_backups` controls the number of audit log files to be reserved on each TiDB server. By setting this system variable and `tidb_audit_log_max_filesize`, you can limit the upper limitation of the overall log size.
- `tidb_audit_log_reserved_days` controls the number of days that a single audit log is reserved on the TiDB server.

Audit log redaction

TiDB might print out confidential data (such as user data) when providing detailed audit log information, causing the risk in data security. Therefore TiDB provides a system variable `tidb_audit_log_redacted` to control whether to redact audit logs.

Note:
For SQL statements containing user passwords, such as `CREATE USER … IDENTIFIED BY …` and `ALTER USER …IDENTIFIED BY …`, TiDB always redacts the audit log. However, if a parser error occurs, the password might be leaked in the REASON information. A workaround for the leakage is to filter out all failed SQL statements from the audit log, by a filter declaring that `STATUS_CODE` is 0 and events contain QUERY records.

System tables

`mysql.audit_log_filters`

Records available filters for database auditing:

- FILTER_NAME: the name of the filter

- CONTENT: the filter declaration in JSON

```
Unset
> desc mysql.audit_log_filters;

+-------------+--------------+------+------+---------+-------+
| Field       | Type         | Null | Key  | Default | Extra |
+-------------+--------------+------+------+---------+-------+
| FILTER_NAME | varchar(128) | NO   | PRI  | NULL    |       |
| CONTENT     | text         | YES  |      | NULL    |       |
+-------------+--------------+------+------+---------+-------+
```

The following query shows that currently there are two filters:

- `all_query` includes all audit logs of QUERY events
- `all_connect` includes all audit logs of CONNECT events

```
Unset
> select * from mysql.audit_log_filters;

+-------------+----------------------------------------+
| FILTER_NAME | CONTENT                                |
+-------------+----------------------------------------+
| all_query   | {"filter":[{"class":["QUERY"]}]}       |
| all_connect | {"filter":[{"class":["CONNECT"]}]}     |
+-------------+----------------------------------------+
```

mysql.audit_log_filter_rules

Records filter rules for database auditing. A record in this table represents a filter rule. You can apply multiple filters to one database user, and let multiple users use one filter.

- `USER`: the user to whom this filter rule applies, consisting of the username and hostname.
- `FILTER_NAME`: the filter name for which this filter rule works. It must exist in the table `mysql.audit_log_filters`.
- `ENABLED`: whether the filter rule is enabled.

```
Unset
> desc mysql.audit_log_filter_rules;

+-------------+--------------+------+------+---------+-------+
| Field       | Type         | Null | Key  | Default | Extra |
+-------------+--------------+------+------+---------+-------+
| USER        | varchar(64)  | NO   | PRI  | NULL    |       |
| FILTER_NAME | varchar(128) | NO   | PRI  | NULL    |       |
| ENABLED     | tinyint(4)   | YES  |      | NULL    |       |
+-------------+--------------+------+------+---------+-------+
```

The following query shows that currently there are two filter rules:

- Rule 1 applies and enables the filter `all_query` for all users
- Rule 2 applies and disables the filter `all_connect` for all users named `u`

```
Unset
> select * from mysql.audit_log_filter_rules;

+------+-------------+---------+
| USER | FILTER_NAME | ENABLED |
+------+-------------+---------+
| %@%  | all_query   |       1 |
| u@%  | all_connect |       0 |
+------+-------------+---------+
```

System variables

`tidb_audit_enabled`

- Scope: GLOBAL
- Persists to cluster: Yes
- Type: Boolean
- Default value: `OFF`
- Controls whether to enable database auditing.

tidb_audit_log

- Scope: GLOBAL
- Persists to cluster: Yes
- Type: String
- Default value: `tidb-audit.log`
- Specifies the name and path of audit logs.
- When the value is an absolute path, TiDB sets the file in that path as the audit log file, and creates the file if it does not exist.
- When the value is a relative path, if the TiDB log path is already set at this time, the same folder is used to create the audit log; otherwise the audit log is created in the working directory of the TiDB server.
- When TiDB does not have file permissions for the audit log file, the execution of audited events is affected, but an error message is logged in the TiDB general log.
- When you set this value, you can use the placeholder `%e` to represent the address. TiDB can automatically rewrite this placeholder to *IP Address-Port* when creating audit log files. For example, `set global tidb_audit_log='tidb-audit-%e.log'` creates a log file `tidb-audit-192-168-197-164-4000.log`.
- After log rotation, the old log file is renamed as *filename.timestamp*, and the new log file keeps the name specified in `tidb_audit_log`.

tidb_audit_log_format

- Scope: GLOBAL
- Persists to cluster: Yes
- Type: Enumeration
- Default value: `TEXT`
- Possible values: `TEXT`, `JSON`
- Controls the format of audit logs. If set to JSON format, TiDB saves the audit logs in a file named `@@tidb_audit_log+".json"`. For example, if the tidb_audit_log is `tidb-audit.log`, the JSON audit logs are saved in `tidb-audit.log.json`.

tidb_audit_log_max_filesize

- Scope: GLOBAL
- Persists to cluster: Yes
- Type: Integer
- Default value: `100`. The unit is MB.
- Range: [`0`, `102400`], the maximum value `102400` (100 GB).

- Set the file size limitation for audit logs. A log rotation is triggered if the log file exceeds this limitation.
- In the current version, If you set this variable to `0`, TiDB automatically changes it to `100`. The maximum size for a single audit log is `102400` MB. In future versions, setting this variable to `0` will disable automatic log rotation by file size.

tidb_audit_log_max_lifetime

- Scope: GLOBAL
- Persists to cluster: Yes
- Type: Integer
- Default value: `86400` seconds (1 day)
- Range: [`0`, `2592000`]. The maximum value is `2592000` seconds (30 days).
- Set the time limitation for audit logs. A log rotation is triggered if the log file exceeds this limitation.
- Setting this variable to `0` means to disable automatic log rotation by time.

tidb_audit_log_reserved_backups

- Scope: GLOBAL
- Persists to cluster: Yes
- Type: Integer
- Default value: `10`
- Range: [`0`, `1024`]
- Set the audit log file number reserved in each TiDB server. When the number of files exceeds the value in one TiDB server, the oldest audit log file is removed.
- Setting this variable to `0` means not to limit the audit log file number.

tidb_audit_log_reserved_days

- Scope: GLOBAL
- Persists to cluster: Yes
- Type: Integer
- Default value: `0`
- Range: [`0`, `1024`]
- Set how many days to reserve the audit log file after rotating. When a rotated log file exists longer than the reserved days, it will be removed.
- Setting this variable to `0` means not to limit the reserved time after rotation.

tidb_audit_log_redacted

- Scope: GLOBAL
- Persists to cluster: Yes
- Type: Boolean
- Default value: ON
- Controls whether to redact the database auditing.
- Note that all statements related to changing user passwords are always redacted regardless of the value of this variable.

Functions

audit_log_rotate

SELECT audit_log_rotate() manually triggers an audit log rotation in all TiDB servers.

audit_log_create_filter

```
Unset
SELECT audit_log_create_filter('<name>', '<filter>');
SELECT audit_log_create_filter('<name>', '<filter>', 1);
```

- Create a filter (in JSON).
- <name> is a case-sensitive string with a max-length of 128 characters. You cannot create filters with the same name. <name> is not required to be the same as the "name" field in <filter>, because the "name" field is optional.
- If you want to create a new filter to replace the filter with the same name, you can specify the third argument to 1.
- After creating a new filter, you can find the corresponding information of the filter in the table mysql.audit_log_filters.

Examples

- Create a filter named `all_dml` to audit all DML operations except `mysql` database:

```
Unset
set @r = '{
   "filter": [
      {
         "class": ["QUERY_DML"],
         "table_excl": ["mysql.*"]
      }
   ]
}';
select audit_log_create_filter('all_dml', @r);
```

- Create a filter named `all` to audit all connections, queries and audit events:

```
Unset
select audit_log_create_filter('all', '{}');
```

- Create a filter named `fail_connect` to audit all failed handshakes:

```
Unset
set @r = '{
   "filter": [
      {
         "class": ["CONNECT"],
         "status_code": [0]
      }
   ]
}';
select audit_log_create_filter('fail_connect', @r);
```

audit_log_remove_filter

```
Unset
SELECT audit_log_remove_filter('<name>');
```

- Remove a filter.
- You cannot remove a filter in use. To remove a filter, you need to first remove the corresponding filter rules by `audit_log_remove_rule`.

audit_log_create_rule

```
Unset
SELECT audit_log_create_rule('<user>@<host>', '<name>');
SELECT audit_log_create_rule('<user>@<host>', '<name>', 1);
```

- Create a filter rule for audit user `<user>@<host>` with filter `<name>`, and enable it immediately.
- The filter `<name>` must exist in the table `mysql.audit_log_filters`.
- The audit user `<user>@<host>` consists of the username and hostname with `@` as separator, where `@` and `<host>` are optional. Both username and hostname can be identifiers with wildcards:
  - `%` for matching any username/hostname
  - `_` for matching any character
- You cannot create a filter rule with the same filter for one user. If you want to create a new rule to replace an existing rule with the same user and filter, specify the third argument as `1`.
- After creating a new rule, you can find the corresponding information in the table `mysql.audit_log_filter_rules`.

Examples

Assume there is already a filter named `f`.

Apply this filter to all users:

```
SELECT audit_log_create_rule('%@%', 'f');
```

Apply this filter to users named `root`:

```
SELECT audit_log_create_rule('root@%', 'f');
```

Apply the filter to users with the hostname starting with `192`:

```
SELECT audit_log_create_rule('%@192.%', 'f');
```

Apply the filter to the user `admin@localhost`:

```
SELECT audit_log_create_rule('admin@localhost', 'f');
```

audit_log_remove_rule

```
Unset
SELECT audit_log_remove_rule('<user>@<host>', '<name>');
```

Remove the filter rule of the `<user>@<host>` user and the `<name>` filter.

audit_log_enable_rule

```
Unset
SELECT audit_log_enable_rule('<user>@<host>', '<name>');
```

Enable the filter rule of the `<user>@<host>` user and the `<name>` filter.

audit_log_disable_rule

```
Unset
SELECT audit_log_disable_rule('<user>@<host>', '<name>');
```

Disable (but do not remove) the filter rule of the `<user>@<host>` user and the `<name>` filter.

Dynamic privileges

AUDIT_ADMIN

When the Security Enhanced Mode (SEM) mode is OFF, only users with `AUDIT_ADMIN` or `SUPER` privileges can perform settings related to database auditing, including:

- Functions for database auditing
- Querying and setting system variables for database auditing
- Querying and updating system tables for database auditing

RESTRICTED_AUDIT_ADMIN

When the SEM mode is ON, only users with `RESTRICTED_AUDIT_ADMIN` privileges can perform settings related to database auditing, including:

- Functions for database auditing
- Querying and setting system variables for database auditing
- Querying and updating system tables for database auditing

Migration from audit plugin to database auditing

The legacy TiDB audit plugin has no conflict with the new database auditing.

Starting from TiDB v7.1, it is recommended to migrate the legacy plugin to the new database auditing to get more powerful auditing capabilities. You need to reconfigure and define the rules according to the description in this user guide.
After the configuration is completed and verified, execute the following command to disable the legacy audit plugin:

```
Unset
mysql> admin plugins disable audit;
```