平凯数据库用户指南

平凯星辰

目录

介绍

数据库审计是平凯数据库的重要功能，为企业提供了一个强大的监管和审计工具。审计可以记录所有的查询语句、事务操作、用户登录等信息，并且可以将这些信息存储到审计日志中，以备将来的查询和分析，从而保证企业的数据安全和合规性。数据库审计功能可以帮助企业管理人员追踪数据库操作的来源和影响，确保数据不被非法窃取或篡改。同时，数据库审计还可以帮助企业遵守各种法规和合规要求，保障企业在法律和道德方面的合规性。

TiDB 在 v7.1 发布了重新设计的数据库审计功能（以下简称 “数据库审计”）。

本文介绍如何使用平凯数据库审计功能。

数据库审计与审计插件的差异

	数据库审计	**审计插件**
开关审计功能	`set global tidb_audit_enabled = 1;` `set global tidb_audit_enabled = 0;`	`admin plugins enable audit;` `admin plugins disable audit;`
一条 SQL 语句可对应的审计事件类型数量	多个	一个
修改审计对象	使用 `audit_log_create_filter` 创建过滤器之后使用 `audit_log_create_rule` 将过滤器绑定到用户	修改 `mysql.tidb_audit_table_access` 表之后使用 `flush tidb plugins audit;` 刷新配置
审计目标过滤	通配符结合 JSON	正则表达式
修改日志路径和格式	通过 `tidb_audit_log` 设置路径，通过 `tidb_audit_log_format` 设置格式	不支持
审计日志格式	文本、JSON	文本

	数据库审计	审计插件
开关审计功能	`set global tidb_audit_enabled = 1;` `set global tidb_audit_enabled = 0;`	`admin plugins enable audit;` `admin plugins disable audit;`
开启和关闭审计日志脱敏	通过 `tidb_audit_log_redacted` 设置审计日志是否脱敏	通过 `tidb_audit_redact_log` 设置审计日志是否脱敏
日志轮替	支持手动轮替，或者根据文件大小和存续时间触发轮替	根据文件大小触发自动轮替
通过动态权限限制用户修改审计日志相关设置	`AUDIT_ADMIN` 和 `RESTRICTED_AUDIT_ADMIN`	不支持
清除审计日志	可以通过 `tiup cluster` 手动清除，也可以通过 `tidb_audit_log_reserved_backups` 和 `tidb_audit_log_reserved_days` 设置自动清除	只能手动清除：`tiup cluster clean <cluster-name> --audit-log`

获取数据库审计功能

数据库审计是平凯数据库提供的功能，请咨询 TiDB 顾问团队以获取平凯数据库，同时享受商业专家的支持服务。

操作指南

本章将从使用的角度出发，简单介绍如何在平凯数据库中开启与使用数据库审计功能。更多细节可参考后续章节。

基础操作

1. 确认数据库审计功能已经打开

可以通过查询系统变量 tidb_audit_enabled 判断当前是否开启数据库审计功能。
可以通过修改系统变量 tidb_audit_enabled 开启与关闭数据库审计功能。

```
 TiDB root@127.0.0.1:test> set global tidb_audit_enabled = 1;
Query OK, 0 rows affected
Time: 0.004s
```

2. 查询当前已经存在的审计日志过滤器与过滤规则

可以通过系统表 mysql.audit_log_filters 查询已经定义的审计日志过滤器，通过系统表 mysql.audit_log_filter_rules 查询已经定义和开启的审计日志过滤规则。未有任何定义的过滤器和过滤规则时，该两表均为空表。

3. 创建一个过滤器，并用一个过滤规则将这个过滤器应用到指定用户上

使用 audit_log_create_filter 函数创建一个审计所有日志事件的过滤器 all：

```
 TiDB root@127.0.0.1:test> select audit_log_create_filter('all', '{}')
+--------------------------------------+
| audit_log_create_filter('all', '{}') |
+--------------------------------------+
| OK                                   |
+--------------------------------------+
1 row in set
```

使用 audit_log_create_rule 函数创建一个过滤规则，将 all 过滤器应用于所有用户

```
 TiDB root@127.0.0.1:test> SELECT audit_log_create_rule('%@%', 'all')
+-------------------------------------+
| audit_log_create_rule('%@%', 'all') |
+-------------------------------------+
| OK                                  |
+-------------------------------------+
1 row in set
```

再次查询 mysql.audit_log_filters 和 mysql.audit_log_filter_rules 系统表确认过滤器与过滤规则均被创建。此时，所有类型的日志事件将被审计。

更多设置

1. 可以通过系统变量 `tidb_audit_log` 设置日志文件的地址，通过 `tidb_audit_log_format` 设置日志文件格式为文本格式或者 JSON 格式
2. 可以设置日志轮替来限制日志文件的大小
3. 可以设置日志文件保留数量与时间实现对审计归档文件的自动管理
4. 可以设置日志脱敏防止审计日志泄漏敏感信息
5. 可以通过函数 audit_log_disable_rule 暂时停用过滤规则
6. 可以通过动态权限 AUDIT_ADMIN 和 RESTRICTED_AUDIT_ADMIN 限制用户设置与修改审计日志相关配置

审计覆盖范围

所有用户接口的操作都在审计范围内，包括 Prepared Statement 接口。由此类操作派生 (Recursive) 出的操作不在审计范围内。

下列外部工具操作在审计范围内：

- TiCDC 的写入
- 通过 TiDB Data Migration (DM) 的写入
- 利用 SQL 接口的 BR 命令发起的备份操作

下列操作**不**在审计范围内：

- 数据库系统内部操作
- 由用户操作派生出的数据库操作
- 通过 TiDB Lightning 的导入操作
- TiSpark 相关操作
- 利用外部 BR 工具发起的备份操作
- 文本和协议形式的 `PREPARE` 和 `EXECUTE` 命令（实际执行的语句会被审计）
- 数据库启动、停止、扩容和缩容操作

审计日志事件

TiDB 审计日志根据 SQL 语句的类型将 SQL 操作分为若干事件类型（Event Class）。一条 SQL 语句对应一种或多种事件类型。一种事件类型可以是另一种事件类型的子类型（Subclass），比如：

- `SELECT * FROM t` 对应的事件类型是 `QUERY` 和 `SELECT`。`SELECT` 属于 `QUERY` 的子类型。

- `INSERT INTO t VALUES (1),(2),(3)` 对应的事件类型是 `QUERY`、`QUERY_DML` 和 `INSERT`。`QUERY_DML` 属于 `QUERY` 的子类型，`INSERT` 属于 `QUERY_DML` 的子类型。
- `SET GLOBAL tidb_audit_enabled = 1` 对应的事件类型是 `AUDIT`、`AUDIT_SET_SYS_VAR` 和 `AUDIT_ENABLE`。`AUDIT_SET_SYS_VAR` 属于 `AUDIT` 的子类型，`AUDIT_ENABLE` 属于 `AUDIT_SET_SYS_VAR` 的子类型。

使用事件类型来区别不同的 SQL 操作有以下优点：

- 不同的事件类型的审计日志记录不同的信息，比如 `CONNETION` 事件需要记录客户端的 IP 地址、端口等信息，而 `SELECT` 事件则需要记录查询的 SQL 语句、访问的表等信息。
- 可以根据不同的事件类型来选择你关心的 SQL 语句，过滤掉不需要的审计日志。

TiDB 审计日志有以下事件类型：

事件类型	**描述**	**父类型**
CONNECTION	记录所有与连接相关的操作，包括握手、建立连接、断开连接、重设连接、变更用户等	-
CONNECT	记录连接过程中的握手操作	CONNECTION
DISCONNECT	记录断开连接的操作	CONNECTION
CHANGE_USER	记录变更用户的操作	CONNECTION
QUERY	记录所有执行 SQL 语句的操作，包括所有对数据的查询和修改的报错	-
TRANSACTION	记录所有与事务相关的语句，比如 BEGIN，COMMIT，ROLLBACK 等	QUERY
EXECUTE	记录所有执行 EXECUTE 语句的操作	QUERY
QUERY_DML	记录所有 DML 语句的操作，包括 INSERT、REPLACE、UPDATE、DELETE 和 LOAD DATA	QUERY
INSERT	记录所有 INSERT 语句的操作	QUERY_DML
REPLACE	记录所有 REPLACE 语句的操作	QUERY_DML
UPDATE	记录所有 UPDATE 语句的操作	QUERY_DML

事件类型	描述	父类型
DELETE	记录所有 DELETE 语句的操作	QUERY_DML
LOAD DATA	记录所有 LOAD DATA 语句的操作	QUERY_DML
SELECT	记录所有 SELECT 语句的操作	QUERY
QUERY_DDL	记录所有 DDL 语句的操作	QUERY
AUDIT	记录所有 TiDB 审计日志相关设置语句的操作，包括系统变量和函数调用	-
AUDIT_SET_SYS_VAR	记录所有设置 TiDB 审计日志相关系统变量语句的操作	AUDIT
AUDIT_FUNC_CALL	记录所有调用 TiDB 审计日志相关函数的操作	AUDIT
AUDIT_ENABLE	记录所有开启 TiDB 审计日志的操作	AUDIT_SET_SYS_VAR
AUDIT_DISABLE	记录所有关闭 TiDB 审计日志的操作	AUDIT_SET_SYS_VAR

审计日志记录信息

通用信息

所有类型的审计日志均包含以下日志信息：

日志信息	描述
ID	标识该操作审计记录的唯一标识符
EVENT	该操作审计记录对应的事件类型，多个事件类型之间以逗号 (,) 分隔
USER	该操作审计记录对应的用户名
ROLES	用户在执行操作时所拥有的角色列表
CONNECTION_ID	上述用户所在连接的标识符

TABLES	该操作审计记录相关的所有表的表名
STATUS_CODE	该操作审计记录对应的操作是否成功，1 表示成功，0 表示失败。
REASON	该操作审计记录对应的错误信息，仅当该操作出现错误时才会记录该信息。

SQL 语句信息

当事件类型为 `QUERY` 或其子类型时，审计日志将记录以下信息：

信息	描述
CURRENT_DB	当前选择的数据库名。
SQL_TEXT	记录执行的 SQL 语句。如果已经开启日志脱敏，则记录脱敏后的 SQL 语句。
EXECUTE_PARAMS	记录传入 EXECUTE 语句的参数。仅当事件类型包含 EXECUTE，且未开启日志脱敏时才会记录该信息。
AFFECTED_ROWS	该 SQL 语句影响的行数，仅当事件类型包含 QUERY_DML 时记录该信息。

连接信息

当事件类型为 `CONNECTION` 或其子类型时，审计日志将记录以下信息：

信息	**描述**
CURRENT_DB	当前选择的数据库名。当事件类型包含 DISCONNECT 时，不记录该信息。
CONNECTION_TYPE	连接的类型，包括 Socket、UnixSocket 和 SSL/TLS
PID	当前连接的进程 ID
SERVER_VERSION	当前连接的 TiDB 服务器版本

SSL_VERSION	当前连接使用的 SSL 版本
HOST_IP	当前连接的 TiDB 服务器的 IP 地址
HOST_PORT	当前连接的 TiDB 服务器的端口
CLIENT_IP	当前连接的客户端的 IP 地址
CLIENT_PORT	当前连接的客户端的端口

审计操作信息

当事件类型为 `AUDIT` 或其子类型时，审计日志将记录以下信息：

信息	描述
AUDIT_OP_TARGET	TiDB 审计日志相关设置的对象，包括系统变量（比如 `tidb_audit_enabled`）和函数（比如 `audit_log_create_filter`）
AUDIT_OP_ARGS	TiDB 审计日志相关设置的参数，比如开启审计日志功能时设置 `tidb_audit_enabled` 为 `ON`；或者通过 `audit_log_remove_filter` 删除过滤器时，参数为删除的过滤器的名字

审计日志过滤与规则

你可以通过过滤器 (filter) 与过滤规则 (rule) 过滤 TiDB 审计日志中的记录。"过滤器"定义审计对象，比如需要记录的事件、目标数据库对象等；"过滤规则"创建"过滤器"所作用的数据库用户。多个数据库用户可以共享一个"过滤器"，用这种方式能够实现定义的共享，比如企业可能存在几种安全级别的用户，我们可以为每种安全级别创建"过滤器"，同一安全级别的用户能够共享同一个"过滤器"。当安全规范更新时，只需要更新相应的"过滤器"即可。

过滤器

过滤器可以从以下几个方面对审计日志进行过滤。

- 审计事件类型：包括包含的事件类型 (class) 与排除的事件类型 (class_excl)。如果 class 与 class_excl 均未声明，则所有类型的事件将被记录在审计日志中。
- 审计事件相关联的表：包括包含的表 (table) 与排除的表 (table_excl)。如果 table 与 table_excl 均未声明，则所有表的操作将被记录在审计日志中。
- 审计事件是否执行成功：可以使用 status_code 指定需要审计的事件的结果，1 表示审计执行成功的事件，0 表示审计执行失败的事件。如果不声明 status_code，则所有执行成功和失败的事件都将被记录在审计日志中。

过滤器以 JSON 格式进行定义。格式如下：

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

其中：

- name 是过滤器的名字，可以为空
- filter 包含若干（可以为 0）个 filterSpec。只要任意 filterSpec 生效，该 filter 就生效
- 一个 filterSpec 中可以对 class、class_excl、table、table_excl 和 status_code 进行声明，且均可以为空。同一 filterSpec 中的声明必须同时成立，该 filterSpec 才生效

{} 是一个空过滤器，可以审计所有 CONNECTION、QUERY 和 AUDIT 事件。
下面是一个名字为 a、审计所有失败的 DDL、失败的连接，以及所有对 test 数据库的查询的过滤器：

```
{
```

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

TiDB 中通过 `audit_log_create_filter` 函数创建过滤器，创建成功后可以通过 `mysql.audit_log_filters` 查询过滤器。

注意：`AUDIT` 事件类型的审计日志会强制记录，无法通过过滤器排除。

过滤规则

创建过滤器之后，你可以通过过滤规则将过滤器与用户进行绑定。该过滤器决定绑定后的用户的操作是否需要记录到审计日志中。你也可以方便地禁用或启用该过滤器。

一条过滤规则由审计用户与过滤器名组成。

- 审计用户由用户名与用户主机名组成，并以 `@` 分隔。和数据库用户定义相同，主机名部分可以使用 `%` 和 `_` 作为通配符，匹配到最长用户名的规则生效。
- 过滤器名必须存在于 `mysql.audit_log_filters` 中。

TiDB 中通过 `audit_log_create_rule` 函数创建过滤规则，创建成功后可以通过 `mysql.audit_log_filter_rules` 查询过滤规则。
过滤规则创建成功后，过滤器将对所有指定的审计用户生效。可以通过 `audit_log_enable_rule` 启用一个过滤规则，通过 `audit_log_disable_rule` 禁用一个过滤规则。

注意：
对于一条审计日志记录，只要其满足任意一条过滤规则，则会被记录到审计日志中。这意味着如果同一时刻有多条对于同一记录冲突的过滤规则生效，则该记录仍会被记录到日志中。

例如，有一个名为 visit_test 的过滤器 {"filter":[{"table":["test.*"]}]} 用于过滤所有访问 test 数据库的操作，另一个名为 not_visit_test 的过滤器 {"filter":[{"table_excl":["test.*"]}]} 用于排除所有访问 test 数据库的操作。当这两个过滤器对应的过滤规则同时生效时，所有对 test 数据库的操作因为满足 visit_test 而将被审计日志记录。

只要审计功能打开， AUDIT 类型的所有事件将会被强制审计，不会被任何规则移除。

日志文件格式

TiDB 审计日志支持以文本或 JSON 两种格式进行记录。可以通过 tidb_audit_log_format 选择日志格式。

对 select * from t，文本格式的审计日志示例如下：

```
[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`"]
```

JSON 格式的审计日志示例如下：

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

日志轮替

日志轮替 (Log Rotation) 可以用于限制日志文件的大小。你可以使用 tidb_audit_log_max_filesize 设置单个审计日志文件的大小，通过 tidb_audit_log_max_lifetime 设置触发审计日志轮替的时间。你可以同时设置上述两个变量，满足任一条件都会触发日志轮替。

你也可以通过 audit_log_rotate 函数手动触发一次日志轮替。

日志文件保留数量与时间

为了实现对审计归档文件的自动管理，TiDB 提供系统变量来控制审计日志保留的数量与时间：

- tidb_audit_log_reserved_backups 控制每台 TiDB 服务器上保留的审计日志文件的数量。结合日志文件大小 tidb_audit_log_max_filesize 的设置，可以限制总体日志大小的上限。
- tidb_audit_log_reserved_days 控制单个审计日志在 TiDB 服务器上保留的天数。

日志脱敏

TiDB 在提供详细的审计日志信息时，可能会把数据库敏感的数据（例如用户数据）打印出来，造成数据安全方面的风险。因此 TiDB 提供 tidb_audit_log_redacted 来控制是否对审计日志脱敏。

注意：
对于包含用户密码的 SQL 语句（CREATE USER ... IDENTIFIED BY ...，SET PASSWORD 和 ALTER USER ... IDENTIFIED BY ...），无论是否开启日志脱敏，审计日志均会对密码信息进行脱敏。
但是如果一条包含密码的 SQL 语句出现语法错误，则密码信息可能泄漏在 REASON 信息中。一个避免此泄漏风险的方法是，使用过滤器将出现错误的 SQL 语句（即 STATUS_CODE 为 0、EVENT 包含 QUERY 的记录）从审计日志中过滤掉。

系统表

mysql.audit_log_filters

记录审计日志可用的过滤器：

- FILTER_NAME：过滤器名
- CONTENT：JSON 格式的过滤器内容

```
 > desc mysql.audit_log_filters;
```

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

以下查询及其结果表示，当前有两个过滤器：

- all_query 选出所有 QUERY 事件类型的日志记录
- all_connect 选出所有 CONNECT 事件类型的日志记录

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

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

mysql.audit_log_filter_rules

记录审计日志的过滤规则，即用户与过滤器的对应关系。该表中一条记录代表一条过滤规则。一个用户可以使用多个过滤器，一个过滤器也可以被多个用户使用。

- USER：该过滤规则作用的用户，包括用户名与用户地址
- FILTER_NAME：该过滤规则使用的过滤器名，必须是 mysql.audit_log_filters 中存在的过滤器
- ENABLED：是否开启该过滤规则

```
 > 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    |       |
+-------------+--------------+------+------+---------+-------+
```

以下查询及其结果表示，当前有两条过滤规则：

- 过滤规则 1 对所有用户应用过滤器 all_query，且当前开启该过滤规则
- 过滤规则 2 对所有名为 u 的用户应用过滤器 all_connect，且当前关闭该过滤规则

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

```
+------+-------------+---------+
```

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

系统变量

tidb_audit_enabled

- 作用域：GLOBAL
- 是否持久化到集群：是
- 类型：布尔型
- 默认值：`OFF`
- 控制是否开启 TiDB 审计日志。

tidb_audit_log

- 作用域：GLOBAL
- 是否持久化到集群：是
- 类型：字符串
- 默认值：`tidb-audit.log`
- 设置 TiDB 审计日志的文件名与存储路径。
- 当设置的值是一个绝对路径时，TiDB 将会设置该路径的文件为审计日志的写入文件，如果文件不存在会创建文件。
- 当设置的值是一个相对路径时，如果此时 TiDB 日志路径已经设置，则会使用相同的文件夹来创建审计日志；否则使用 TiDB 服务器的工作目录来创建审计日志。
- 当 TiDB 没有设置的审计日志文件的权限时，审计的相关事件的执行将不会收到影响，但会在 TiDB 日志中写入 ERROR 信息。
- 设置 `tidb_audit_log` 时可以使用占位符 `%e` 来表示地址信息，TiDB 会自动将 `%e` 替换为 IP 地址-端口，比如 `set global tidb_audit_log='tidb-audit-%e.log'` 将会创建名为 `tidb-audit-192-168-197-164-4000.log` 的审计日志文件。
- 发生日志轮替之后，旧的日志文件会以“文件名.时间戳”的形式被重命名，新的日志文件保留 `tidb_audit_log` 设置的值。

tidb_audit_log_format

- 作用域：GLOBAL
- 是否持久化到集群：是
- 类型：枚举型
- 默认值：`TEXT`
- 可选值：`TEXT`、`JSON`
- 设置 TiDB 审计日志的输出格式。当设置为 JSON 格式时，TiDB 会将日志写入文件名为 `@@tidb_audit_log+".json"` 的文件中。比如当 `tidb_audit_log` 为 `tidb-audit.log` 时，JSON 格式的审计日志记录在 `tidb-audit.log.json`。

tidb_audit_log_max_filesize

- 作用域：GLOBAL
- 是否持久化到集群：是
- 类型：整数型
- 默认值：`100`，单位 MB
- 范围：`[0, 102400]`，最大值对应 `100 GB`
- 设置 TiDB 审计日志的文件大小上限，超过该上限将触发日志轮替。
- 在当前版本中，设置为 `0` 会自动设置为 `100`，单日志最大为 `102400 MB`。后续版本 `0` 将会代表不做日志轮替。

tidb_audit_log_max_lifetime

- 作用域：GLOBAL
- 是否持久化到集群：是
- 类型：整数型
- 默认值：`86400`，单位为秒，对应 1 天
- 范围：`[0, 2592000]`，最大值对应 30 天
- 设置 TiDB 审计日志自动轮替的时间。
- 设置为 `0` 时表示不按照时间进行自动轮替。

tidb_audit_log_reserved_backups

- 作用域：GLOBAL
- 是否持久化到集群：是

- 类型：整数型
- 默认值：`10`
- 范围：`[0, 1024]`
- 设置每个 TiDB 节点保存的审计日志文件的数量。当某个节点中存在的审计日志文件的数量超过该设置时，将删除最旧的审计日志文件。
- 设置为 `0` 时表示不对审计日志文件的数量做限制。

tidb_audit_log_reserved_days

- 作用域：GLOBAL
- 是否持久化到集群：是
- 类型：整数型
- 默认值：`0`
- 范围：`[0, 1024]`
- 设置 TiDB 审计日志文件被轮替后保留的天数。当某个已被轮替的审计日志文件的保留天数超过该设置时，将删除该审计日志文件。
- 设置为 `0` 时表示不对审计日志文件轮替后保留的时长做限制。

tidb_audit_log_redacted

- 作用域：GLOBAL
- 是否持久化到集群：是
- 类型：布尔型
- 默认值：`ON`
- 设置是否对 TiDB 审计日志开启脱敏。所有修改用户密码的操作在 TiDB 审计日志中总是脱敏。

函数

audit_log_rotate

`SELECT audit_log_rotate()` 在所有 TiDB 节点上手动强制触发一次审计日志轮替。

audit_log_create_filter

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

- 创建一个名为 <name> 的过滤器，内容为 <filter>，即前文所述的 JSON 形式的过滤器。
- <name> 为大小写敏感的字符串，最大长度为 128 个字符。不能创建两个同名的过滤器。<name> 不需要与 <filter> 中的 "name" 字段相同，因为 "name" 字段是可选的。
- 如果需要创建一个新的过滤器覆盖掉旧的同名过滤器，可以在调用函数时使用第三个参数 1。
- 创建成功后，对应的过滤器信息可通过查找 mysql.audit_log_filters 得到。

示例

创建一个名为 all_dml 的过滤器，对所有的 DML 操作进行审计，mysql 数据库除外：

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

创建一个名为 all 的过滤器，对所有连接、查询和审计操作进行审计：

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

创建一个名为 fail_connect 的过滤器，对所有握手失败的连接进行审计：

```
 set @r = '{
   "filter": [
      {
         "class": ["CONNECT"],
```

```
        "status_code": [0]
      }
   ]
}';
select audit_log_create_filter('fail_connect', @r);
```

audit_log_remove_filter

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

- 删除名为 <name> 的过滤器。
- 正被过滤规则引用的过滤器无法删除，需要先通过 audit_log_remove_rule 删除所有引用该过滤器的规则之后才能删除。

audit_log_create_rule

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

- 对审计用户 <user>@<host> 创建一个关联过滤器 <name> 的过滤规则，并立即启用。
- 过滤器 <name> 必须已经存在于 mysql.audit_log_filters 中。
- 审计用户 <user>@<host> 包含用户名和用户主机名，以 @ 作为分隔符，其中 @ 和 <host> 可忽略，表示包含所有的主机名。用户名和主机名可以是具体的标识符，也可以使用通配符进行匹配：
  - % 表示匹配任意用户名/主机名
  - _ 表示匹配任一字符
- 不能对同一个审计用户创建关联同一个过滤器的过滤规则。如果需要创建一个新的过滤规则覆盖掉旧的同名过滤规则，可以在调用函数时使用第三个参数 1。
- 创建成功后，对应的过滤规则信息可通过查找 mysql.audit_log_filter_rules 得到。

示例

假设现在已有名为 f 的过滤器。

将该过滤器与所有用户相关联：

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

将该过滤器与名为 root 的所有用户相关联:

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

将该过滤器与所有 192. 开头的主机用户相关联：

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

将该过滤器与用户 admin@localhost 相关联：

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

audit_log_remove_rule

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

删除 <user>@<host> 与 <name> 的过滤规则，即取消 <user>@<host> 与 <name> 的关联。

audit_log_enable_rule

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

启用 <user>@<host> 与 <name> 的过滤规则。

audit_log_disable_rule

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

停止使用（但不删除）<user>@<host> 与 <name> 的过滤规则。

动态权限

AUDIT_ADMIN

当安全增强模式 (Security Enhanced Mode, SEM) 关闭时，只有拥有 AUDIT_ADMIN 或者 SUPER 权限的用户才能设置和修改审计日志相关的配置，包括：

- 使用审计日志相关函数
- 查询和修改审计日志相关系统变量
- 查询和修改审计日志相关系统表

RESTRICTED_AUDIT_ADMIN

当安全增强模式 (SEM) 开启时，只有拥有 RESTRICTED_AUDIT_ADMIN 权限的用户才能设置与修改审计日志相关配置，包括：

- 使用审计日志相关函数
- 查询和修改审计日志相关系统变量
- 查询和修改审计日志相关系统表

审计插件切换到数据库审计

TiDB 旧审计插件与新的数据库审计功能之间并不冲突，可以共存。

在 v7.1 之后推荐使用新的审计功能替代原有插件，从而获取更强大的数据库审计能力。 如果要从旧审计插件切换，你需要根据本文的描述重新配置并定义规则。
配置完成并验证无误后，执行如下命令禁用原有审计插件：

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