

云数据库 RDS for MySQL

内核介绍

文档版本 01
发布日期 2025-10-31



版权所有 © 华为云计算技术有限公司 2025。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 RDS for MySQL 内核版本说明.....	1
2 主要内核功能介绍.....	19
2.1 连接线程池.....	19
2.2 MDL 锁视图.....	22
2.3 在线扩展 varchar 类型字段.....	25
2.4 主动终止空闲事务.....	26
2.5 DDL 进度显示.....	29
2.6 优化阶段拦截写事务.....	30

1 RDS for MySQL 内核版本说明

本章节介绍云数据库RDS的内核版本更新说明。

- [RDS for MySQL 8.0](#)
- [RDS for MySQL 5.7](#)
- [RDS for MySQL 5.6](#)

RDS for MySQL 8.0

表 1-1 云数据库 RDS for MySQL 8.0 版本说明

日期	特性描述
2025-10-20	内核版本正式升级到8.0.41。 • 新特性及性能优化 - 引入社区8.0.41版本，相关变化参见社区8.0.41版本发布说明。 - 在单次事务中执行大量数据操作（如插入、更新或删除数百万行）时，如果用户终止事务，系统将进入回滚阶段。此过程通常与事务原始执行时间相近，期间不仅会持续持有数据库锁，还会占用大量 I/O 资源，对后续业务造成严重影响。为避免这类性能问题，本版本支持在执行计划估算阶段提前识别并拦截可能带来风险的大规模事务，并提醒用户关注，保障数据库运行的稳定性和服务的高可用性，详见优化阶段拦截写事务。 • 修复问题 - 优化解决二级分区部分场景下的分区扫描问题。

日期	特性描述
2025-07-28	内核版本升级到8.0.41。 - 新特性及性能优化 引入社区8.0.41版本，本版本仅提供公测，相关变化参见社区8.0.41版本发布说明。 - 修复问题 - 修复动态脱敏功能添加超长脱敏规则时可能导致的进程异常问题。 - 优化动态脱敏生效规则较多时，部分场景的只读性能。
2025-04-23	8.0.32版本更新以下内容： - 问题修复： - 修复部分场景库级回收站不生效问题。
2025-01-14	8.0.32版本更新以下内容： - 修复问题 - 内核针对不满足口令复杂度的账号密码会报warning。 - 修复从8.0.28升级到8.0.32版本升级后，实例上root拥有的部分权限被取消的问题。 - 修复部分场景下，审计日志记录问题。 - max_execution_time参数的最小值从60s调整为1s。 说明 - 如需升级到8.0.32版本，请提交工单申请。 8.0.32版本由于社区引入Instant col特性导致在一些场景下存在性能劣化，详见社区bug111538。
2024-10-28	8.0.32版本更新以下内容： - 修复问题 - 优化statement outline，支持explain analyze语句。 - 解决8.0.28.x社区版本死锁问题。 - 解决SQL限流功能的一些Bug。 说明 - 如需升级到8.0.32版本，请提交工单申请。 8.0.32版本由于社区引入Instant col特性导致在一些场景下存在性能劣化，详见社区bug111538。
2024-06-18	8.0.28版本更新以下内容： 修复问题 修复因会话引入LOCK_thd_security_ctx锁，而导致执行SHOW PROCESSLIST语句时可能出现死锁的问题。详见社区Bug#32320541。

日期	特性描述
2024-05-23	内核版本升级到8.0.32。 ● 新特性及性能优化 - MySQL 8.0的动态权限功能允许SUPER用户向其他用户动态授予高级管理权限，例如SESSION_VARIABLES_ADMIN和SYSTEM_VARIABLES_ADMIN等。此版本支持通过内核能力禁用用户为自身赋予可能导致实例异常的危险权限。 - 对表进行加减列时，默认算法支持由INSTANT更改为INPLACE或COPY算法。 ● 修复问题 具体修改点同社区版一致，详见社区8.0.32版本发布说明。 说明 ● 如需升级到8.0.32版本，请提交工单申请。 ● 8.0.32版本由于社区引入Instant col特性导致在一些场景下存在性能劣化，详见社区bug111538。 ● 安全加固 - 解决安全漏洞：CVE-2018-25032、CVE-2021-22570、CVE-2022-21412、CVE-2022-21413、CVE-2022-21414、CVE-2022-21415、CVE-2022-21417、CVE-2022-21418、CVE-2022-21423、CVE-2022-21425、CVE-2022-21427、CVE-2022-21435、CVE-2022-21436、CVE-2022-21437、CVE-2022-21438、CVE-2022-21440、CVE-2022-21444、CVE-2022-21451、CVE-2022-21452、CVE-2022-21455、CVE-2022-21459、CVE-2022-21460、CVE-2022-21462、CVE-2022-21478、CVE-2022-21479、CVE-2022-21509、CVE-2022-21515、CVE-2022-21517、CVE-2022-21522、CVE-2022-21525、CVE-2022-21526、CVE-2022-21527、CVE-2022-21528、CVE-2022-21529、CVE-2022-21530、CVE-2022-21531、CVE-2022-21534、CVE-2022-21537、CVE-2022-21538、CVE-2022-21539、CVE-2022-21547、CVE-2022-21553、CVE-2022-21556、CVE-2022-21569、CVE-2022-21592、CVE-2022-21594、CVE-2022-21599、CVE-2022-21604、CVE-2022-21605、CVE-2022-21607、CVE-2022-21608、CVE-2022-21611、CVE-2022-21617、CVE-2022-21625、CVE-2022-21632、CVE-2022-21633、CVE-2022-21635、CVE-2022-21637、CVE-2022-21638、CVE-2022-21640、

日期	特性描述
	CVE-2022-21641、CVE-2022-27778、 CVE-2022-32221、CVE-2022-39400、 CVE-2022-39402、CVE-2022-39403、 CVE-2022-39408、CVE-2022-39410、 CVE-2023-21836、CVE-2023-21863、 CVE-2023-21864、CVE-2023-21865、 CVE-2023-21866、CVE-2023-21867、 CVE-2023-21868、CVE-2023-21869、 CVE-2023-21870、CVE-2023-21871、 CVE-2023-21872、CVE-2023-21873、 CVE-2023-21874、CVE-2023-21875、 CVE-2023-21876、CVE-2023-21877、 CVE-2023-21878、CVE-2023-21879、 CVE-2023-21880、CVE-2023-21881、 CVE-2023-21882、CVE-2023-21883、 CVE-2023-21887、CVE-2023-21912、 CVE-2023-21913、CVE-2023-21917、 CVE-2023-21963、CVE-2023-22015、 CVE-2023-22026、CVE-2023-22028、 CVE-2023-22084。
2023-12-01	8.0.28版本更新以下内容： ● 新特性及性能优化 - SQL限流功能变化：所有账户当前都会被限流（之前的行为：对root不限流）；如需设定不限流的账户请联系客服。 - 新增information_schema.rds_sql_filter_info，可以查看SQL限流功能的当前并发数和历史拦截数。 ● 修复问题 - 修复了社区审计日志插件不记录prepare协议语句的问题。

日期	特性描述
2023-09-01	内核版本升级到8.0.28。 • 新特性及性能优化 - 优化了SQL限流功能。 - 优化了大文件异步Purge的性能。 • 修复问题 - 引入社区8.0.28版本的bugfix，详见社区8.0.28版本发布说明。 - 修复了社区并行DDL导致的数据不一致问题。 - 修复了社区审计日志内存泄漏、线程挂住的问题。 • 安全加固 - 解决安全漏洞：CVE-2019-17543、CVE-2020-1971、CVE-2021-22901、CVE-2021-22922、CVE-2021-22923、CVE-2021-22924、CVE-2021-22925、CVE-2021-22926、CVE-2021-22945、CVE-2021-22946、CVE-2021-22947、CVE-2021-2339、CVE-2021-2340、CVE-2021-2342、CVE-2021-2352、CVE-2021-2354、CVE-2021-2356、CVE-2021-2357、CVE-2021-2367、CVE-2021-2370、CVE-2021-2372、CVE-2021-2374、CVE-2021-2383、CVE-2021-2384、CVE-2021-23841、CVE-2021-2385、CVE-2021-2387、CVE-2021-2389、CVE-2021-2390、CVE-2021-2399、CVE-2021-2402、CVE-2021-2410、CVE-2021-2412、CVE-2021-2417、CVE-2021-2418、CVE-2021-2422、CVE-2021-2424、CVE-2021-2425、CVE-2021-2426、CVE-2021-2427、CVE-2021-2429、CVE-2021-2437、CVE-2021-2440、CVE-2021-2441、CVE-2021-2478、CVE-2021-2479、CVE-2021-2481、CVE-2021-3449、CVE-2021-35537、CVE-2021-35546、CVE-2021-35575、CVE-2021-35577、CVE-2021-35583、CVE-2021-35591、CVE-2021-35596、CVE-2021-35602、CVE-2021-35604、CVE-2021-35607、CVE-2021-35610、CVE-2021-35612、CVE-2021-35622、CVE-2021-35623、CVE-2021-35624、CVE-2021-35625、CVE-2021-35626、CVE-2021-35627、CVE-2021-35628、CVE-2021-35629、CVE-2021-35630、CVE-2021-35631、CVE-2021-35632、CVE-2021-35633、CVE-2021-35634、CVE-2021-35635、CVE-2021-35636、CVE-2021-35637、CVE-2021-35638、CVE-2021-35639、CVE-2021-35640、CVE-2021-35641、CVE-2021-35642、CVE-2021-35643、CVE-2021-35644、CVE-2021-35645、CVE-2021-35646、CVE-2021-35647、CVE-2021-35648、

日期	特性描述
	CVE-2021-3711、CVE-2022-0778、CVE-2022-21245、CVE-2022-21249、CVE-2022-21253、CVE-2022-21254、CVE-2022-21264、CVE-2022-21265、CVE-2022-21278、CVE-2022-21297、CVE-2022-21301、CVE-2022-21302、CVE-2022-21303、CVE-2022-21304、CVE-2022-21339、CVE-2022-21342、CVE-2022-21344、CVE-2022-21348、CVE-2022-21351、CVE-2022-21352、CVE-2022-21358、CVE-2022-21362、CVE-2022-21367、CVE-2022-21368、CVE-2022-21370、CVE-2022-21372、CVE-2022-21374、CVE-2022-21378、CVE-2022-21595、CVE-2022-21600、CVE-2023-21950。
2023-03-15	8.0.25版本更新以下内容： ● 新特性及性能优化 - 优化了超大SQL审计日志的打印。 - 强化了内核日志打印的安全性。 ● 修复问题 修复并发实例DDL和DML操作可能导致异常重启问题。用户不再有权限赋予connection_admin权限。
2022-09-09	8.0.25版本更新以下内容： ● 新特性及性能优化 - KILL会话支持per thread连接方式。 - Performance Schema使用增加内存限制。 - 优化特定场景下全量SQL的性能。 - 优化特定场景下internal_tmp_mem_storage_engine为memory引擎的性能问题。 - 升级编译器到GCC 10.3。 ● 修复问题 - 修复临时文件写入可能报错的问题。 - 修复CTE查询返回结果不符合预期的问题。 ● 安全加固 - 解决安全漏洞：CVE-2021-2417、CVE-2021-2339、CVE-2021-2425、CVE-2021-2426、CVE-2021-2427、CVE-2021-2424、CVE-2021-2383、CVE-2021-2384、CVE-2021-2410。

日期	特性描述
2022-06-01	● 新特性及性能优化 - 内核版本升级到8.0.25。 - 支持SQL限流。 - 升级编译器到GCC 9.3。 ● 修复问题 - 修复单个超大Binlog可能导致的复制中断问题。 - 修复社区innodb_row_lock_current_waits统计不准确的问题。 - 修复使用blob字段可能导致的异常重启问题。 ● 安全加固 - 解决安全漏洞：CVE-2021-2307、CVE-2021-2180、CVE-2021-2194。
2021-08-07	● 新特性及性能优化 - 线程池静态连接，提高性能。 - 开启PGO（Profile-Guided Optimization）编译优化。 - 优化MySQL HASH算法。 - 支持数据库添加备注功能。 - 系统库防止被DDL修改。 - 添加参数innodb_total_tablespace用于统计innodb表空间数量。 - 在i_s中提供innodb锁视图。 - 升级openssl, jemalloc, curl开源组件。 ● 修复问题 - 修复XA事务在binlog rotate后可能异常重启的问题。 - 修复全量SQL未记录预编译SQL操作类型的问题。 - 修复flush privieges操作的executed time统计错误的问题。 - 修复审计日志错误写入其他文件的问题。
2021-04-13	● 修复XA事务在数据库异常重启后可能发生丢失的问题。 ● 优化自适应HASH分割算法。 ● 内核版本升级到8.0.21。 ● 安全加固 - 解决安全漏洞：CVE-2020-14697、CVE-2020-14680、CVE-2020-14678、CVE-2020-14663、CVE-2021-2020、CVE-2020-14619、CVE-2020-14591、CVE-2020-14576、CVE-2020-14539。
2021-01-26	● SQL限流特性性能优化。 ● 全量SQL采集性能优化。

日期	特性描述
2020-12-31	性能优化 升级编译器到GCC 9。
2020-12-01	- 性能优化 优化慢日志额外信息的采集效率。 - 修复以下问题 修复XA事务回滚可能会导致备机复制中断问题。
2020-11-06	修复问题 修复微秒计时器中gettimeofday多线程导致计时错误问题。
2020-09-21	- 支持在错误日志中记录详细的连接断开信息。 - 支持index hint功能。
2020-08-03	- 显示大事务执行时间和等待时间。 - 为管理用户使用独立的连接控制。 - 支持SQL限流，业务高峰期可以限制特定SQL的执行频率。 - 优化内核性能。
2020-06-19	- 内核版本升级到8.0.20。 - 优化内核性能。
2020-02-15	- RDS for MySQL 8.0版本正式商用。 - ARM内核版本上线，该内核基于华为云鲲鹏服务器。
2019-12-15	- 内核版本升级到8.0.17。 - 并行创建索引：并行创建索引速度最快提升2.5倍。
2019-10-15	修复“SQL_MODE”为“PAD_CHAR_TO_FULL_LENGTH”时的主备复制异常问题。
2019-09-15	- 支持线程池。详见连接线程池。 - 升级Openssl版本至1.1.1a。 - 支持CTS语法：支持create table xx select语法。 - 用户线程内存使用信息、CPU时间使用信息，您可以通过show full processlist进行查询。

RDS for MySQL 5.7

表 1-2 云数据库 RDS for MySQL 5.7 版本说明

日期	特性描述
2025-04-23	5.7.44版本更新以下内容： ● 修复问题 - 清除server层元数据，解决DDL非原子性导致server层与innodb层元数据不一致问题。
2025-01-14	5.7.44版本更新以下内容： ● 修复问题 - 内核针对不满足口令复杂度的账号密码会报warning。 - 修复部分场景下，审计日志记录问题。 - max_execution_time参数的最小值从60s调整为1s。
2024-10-28	5.7.44版本更新以下内容： ● 修复问题 - 解决SQL限流功能的一些Bug。
2024-04-11	内核版本升级到5.7.44。 ● 具体修改点同社区版一致，详见社区5.7.44版本发布说明。 ● 安全加固 - 解决安全漏洞：CVE-2023-22028、CVE-2023-22084、CVE-2023-38545。
2023-12-01	● 新特性及性能优化 - 升级到5.7.43版本。 - SQL限流功能变化：所有账户当前都会被限流（之前的行为：对root不限流）；如需设定不限流的账户请联系客服。 - 新增information_schema.rds_sql_filter_info，可以查看SQL限流功能的当前并发数和历史拦截数。 ● 修复问题 - 修复了社区审计日志插件不记录prepare协议语句的问题。 ● 安全加固 - 解决安全漏洞：CVE-2022-43551、CVE-2023-21912、CVE-2023-21980、CVE-2023-22007、CVE-2023-22015、CVE-2023-22026、CVE-2023-22053。

日期	特性描述
2023-09-01	5.7.41版本更新以下内容： ● 新特性及性能优化 - 优化了SQL限流功能。 - 优化了jemalloc组件释放内存缓慢，导致OOM的行为。 ● 修复问题 - 修复了社区审计日志内存泄漏、线程挂住的问题。 - 修复了社区间隙锁范围过大的问题。
2023-06-28	● 新特性及性能优化 - 升级到5.7.41版本。 - 增加安全编译选项。 ● 修复问题 - 修复参考表添加索引与另一张表添加外键并发执行时可能出现的复制异常。 - 修复删除外键表后再删除子表时可能出现的复制异常。 ● 安全加固 - 解决安全漏洞：CVE-2023-21963、CVE-2022-32221、CVE-2023-21840、CVE-2022-2097、CVE-2022-21617、CVE-2022-21608、CVE-2022-21592、CVE-2022-21589、CVE-2022-1292、CVE-2022-27778、CVE-2018-25032、CVE-2022-21515。
2022-09-09	● 新特性及性能优化 - 升级到5.7.38版本。 - 升级编译器到GCC 10.3。 - KILL会话支持per thread连接方式。 - 慢日志的阈值可考虑等锁耗时。 - ALT功能安全加固。 ● 修复问题 - 增强分布式XA事务主库崩溃恢复安全。 - 修复开启proxy可能导致的异常重启。 - 修复插件可能内存申请失败导致的异常重启。 ● 安全加固 - 解决安全漏洞：CVE-2022-21454、CVE-2022-21417、CVE-2022-21427、CVE-2022-21451、CVE-2022-21444、CVE-2022-21460。

日期	特性描述
2022-06-01	● 新特性及性能优化 - 内核版本升级到5.7.37。 - 升级编译器到GCC 9.3。 - 升级openssl、curl开源组件。 ● 修复问题 - 修复单个超大Binlog可能导致的复制中断问题。 - 修复slave并发回放grant可能导致的异常重启的问题。 - 修复隐藏自增键可能导致的复制中断问题。 - 修复虚拟字段表在ROLLBACK时可能异常重启的问题。 - 修复加密表在recovery异常重启问题。 - 修复Seconds Behind Master特定场景不准确问题。 ● 安全加固 - 解决安全漏洞：CVE-2022-21367、CVE-2022-21304、CVE-2022-21344。
2022-01-26	● 新特性及性能优化 - 支持隐藏自增键特性。 ● 修复问题 - 修复XA事务重复提交可能导致复制异常问题。 - 修复社区innodb_row_lock_current_waits统计不准确的问题。
2021-11-26	● 新特性及性能优化 - 全量SQL放开单条记录长度限制。 - RDS for MySQL透明应用连续性(ALT)特性第一期。 ● 修复问题 - 修复线程池极端场景的内存问题。 - 修复XA事务备机回放概率性卡住。

日期	特性描述
2021-08-07	● 新特性及性能优化 - 线程池静态连接，提高性能。 - 开启PGO (Profile-Guided Optimization) 编译优化。 - 优化MySQL HASH算法。 - 支持数据库添加备注功能。 - 系统库防止被DDL修改。 - 在errlog中提供IO延迟信息。 - 支持minidump。 - 内核版本升级到5.7.33。 - 升级openssl, jemalloc, curl开源组件。 ● 修复问题 - 修复由于回放保序导致备机小概率复制中断的问题。 - 修复XA事务在binlog rotate后可能异常重启的问题。 - 修复全量SQL可能存在未记录预编译SQL操作类型的问题。 - 修复flush privieges操作的executed time统计错误的问题。 - 修复审计日志错误写入其他文件的问题。 ● 安全加固 解决安全漏洞：CVE-2021-2011、CVE-2021-2178、CVE-2021-2202。
2021-04-13	● 修复XA事务在数据库异常重启后可能发生丢失的问题。 ● 优化自适应HASH分割算法。 ● 内核版本升级到5.7.32。
2021-01-26	● 新特性 支持在代理情况下显示真实的客户端地址。 ● 修复问题 全量SQL采集性能优化。 修复revoke权限可能导致主备权限不一致的问题。 回合8.0中对Instant add column的优化。
2020-12-31	● 性能优化 - 优化慢日志额外信息的采集效率。 - 升级编译器至GCC 9。 ● 修复问题 修复XA事务回滚可能会导致备机复制中断问题。
2020-12-01	减少fil_sys互斥锁之间的冲突频率。

日期	特性描述
2020-11-06	● 新特性 - 内存申请编译器优化。 - 添加utf8优化对非ARM平台的支持。 ● 修复问题 修复微秒计时器中gettimeofday多线程导致计时错误问题。
2020-09-21	● 内核版本升级到5.7.31。 ● 支持SQL限流，业务高峰期可以限制特定SQL的执行频率。
2020-08-03	● 优化内核性能。 ● 支持回收站。 ● 增强本地盘清理数据可靠性。
2020-07-09	● 优化内核性能。 ● 支持在错误日志中记录用户的操作历史。 ● 提高XA事务的稳定性。
2020-06-30	● 优化内核性能。 ● 支持本地盘磁盘日志切换。 ● bufferpool内存初始化模块优化，提高初始化效率。 ● 提升ARM上部分操作的线程安全性。
2020-05-30	● 新特性 - 支持index hint功能。 - 支持抓取全量SQL日志。 ● 修复以下问题 修复用户低概率连接数据库失败问题。
2020-04-30	优化内核性能。
2020-03-30	● 内核版本升级到5.7.29。 ● 优化内核性能。 ● 支持线程池统计IO waits事件。
2020-02-15	● 复制双通道：新增复制状态通道，准确判断主库崩溃时的复制状态，确保事务不丢失。 ● Optimized ROW_IMAGE模式：缩减binlog大小，同时支持迁移，SQL闪回等场景。

日期	特性描述
2019-12-15	- DDL操作进度：通过“information_schema.innodb_alter_table_progress”获取添加列或添加索引操作的进度信息。详见DDL进度显示。 - 长事务：通过show processlist获取事务的执行时间Trx_Executed_Time。 - Online扩展string字段长度：扩展varchar类型字段长度从COPY模式默认变为INPLACE模式。详见在线扩展varchar类型字段。 - InnoDB死锁信息丰富：通过show engine innodb status可以观察到InnoDB层发生死锁情况时完整的现场信息。
2019-10-15	- 性能优化 - ARM内核版本上线，该内核基于华为云鲲鹏服务器。 - 新特性 - 内核版本升级到5.7.27。 - 快速加列：表可以快速添加列，不拷贝数据，不占用磁盘空间和磁盘IO，业务高峰期可以实时变更。 - MDL锁视图：通过“information_schema.metadata_lock_info”获取线程持有或等待的MDL锁信息。详见MDL锁视图。
2019-08-15	Jemalloc内存管理：替换Glibc内存管理模块，降低内存占用，内存分配效率更高效。
2019-06-15	内核版本升级到5.7.25。
2019-05-15	- 新特性 - 备库安全极速模式：在备库“sync_binlog”和“innodb_flush_log_at_trx_commit”为非1配置下，保证备库crash safe数据安全。sysbench高并发纯写模式下，主备复制延迟一直基本为0。 - 修复以下问题 - 修复“relay_log_recovery” = “ON”时，特定场景Kill备库后无法重启的问题。 - 修复“SQL_MODE”为“PAD_CHAR_TO_FULL_LENGTH”时，主备复制异常的问题。 - 修复“performance_schema”重复统计错误的问题。 - 修复对“Performance_schema”中关于复制信息的表进行order by查询时，返回为空的问题。
2019-01-15	- 修复只读执行flush操作后导致主备不一致，复制中断的问题。 - 修复REPAIR，OPTIMIZE等语句导致备机复制线程卡住问题。

日期	特性描述
2018-11-15	- 内核版本升级到5.7.23。 - GTID开启时支持事务中创建和删除临时表。 - 支持表级别MTS并行回放。
2018-07-15	- 内核版本升级到5.7.22。 - 支持线程池。详见连接线程池。 - 支持CTS语法：支持create table xx select语法。 - 算子下推：聚合算子下推到存储引擎层，提升count()，sum()执行速度。 - KILL空闲事务：通过设置超时时间，自动KILL长时间空闲事务。详见主动终止空闲事务。 - 用户线程内存使用信息、CPU时间使用信息，用户通过show full processlist进行查询。

RDS for MySQL 5.6

表 1-3 云数据库 RDS for MySQL 5.6 版本说明

日期	特性描述
2024-10-28	5.6.51版本更新以下内容： - 修复问题 - 回合开源社区row_search_mvcc函数崩溃Bug导致数据库实例崩溃问题。详见社区Bug #96610。 - 优化XA事务在commit后长事务的计算问题。
2023-09-01	5.6.51版本更新以下内容： - 新特性及性能优化 - 优化了SQL限流功能。 - 修复问题 - 修复了社区审计日志线程挂住的问题。
2023-03-15	5.6.51版本更新以下内容： - 新特性及性能优化 - 优化了超大SQL审计日志的打印。 - 强化了日志打印的安全性。 - 修复问题 - 修复参考表添加索引与另一张表添加外键并发执行时可能出现的复制异常。

日期	特性描述
2022-09-09	● 新特性及性能优化 - KILL会话支持per thread连接方式。 ● 修复问题 - 修复开启proxy可能导致的异常重启。 - 修复main.proxy_connect缓冲区溢出。 - 修复插件可能内存申请失败导致的异常重启。
2022-06-01	修复社区innodb_row_lock_current_waits统计不准确的问题。
2021-08-07	● 新特性 - 支持数据库添加备注功能。 - 系统库防止被DDL修改。 - 升级openssl, jemalloc开源组件。 ● 修复问题 - 修复修改密码可能会导致同步断开的问题。 - 修复审计日志错误写入其他文件的问题。
2021-04-13	● 修复由于回放保序导致备机小概率复制中断的问题。 ● 内核版本升级到5.6.51。 ● 安全加固 社区MySQL 5.6版本补丁不再发布，该版本的漏洞补丁服务也不再发布。
2021-01-26	● 新特性 支持在代理情况下显示真实的客户端地址。 ● 修复问题 修复执行select 1 for update报语法错误问题。 全量SQL采集性能优化。
2020-12-31	重构SQL限流功能，提高易用性。
2020-11-06	内核版本升级到5.6.50。
2020-09-23	支持SQL限流，业务高峰期可以限制特定SQL的执行频率。
2020-08-03	内核版本升级到5.6.49。
2020-07-09	● 支持本地盘磁盘日志切换。 ● 支持在错误日志中记录用户的操作历史。
2020-05-30	bufferpool内存初始化模块优化，提高初始化效率。
2020-04-30	修复用户低概率连接数据库失败问题。
2020-03-30	● 支持全量SQL采集功能。 ● 编译器升级到7.3。 ● 内核版本升级到5.6.47。

日期	特性描述
2020-02-15	- 复制双通道：新增复制状态通道，准确判断主库崩溃时的复制状态。确保事务不丢失。 - Optimized ROW_IMAGE模式：缩减binlog大小，同时支持迁移，SQL闪回等场景。
2019-12-15	- 长事务：通过show processlist获取事务的执行时间Trx_Executed_Time。 - Online扩展String字段长度：扩展VARCHAR类型字段长度从COPY模式默认变为INPLACE模式。详见在线扩展varchar类型字段。 - InnoDB死锁信息丰富：通过show engine innodb status命令可以观察到InnoDB层死锁发生时完整的现场信息。
2019-10-15	- 内核版本升级到5.6.45。 - 用户线程内存使用信息、CPU时间使用信息，用户通过show full processlist进行查询。 - KILL空闲事务：通过设置超时时间，自动KILL长时间空闲事务。详见主动终止空闲事务。
2019-08-15	Jemalloc内存管理：替换Glibc内存管理模块，降低内存占用，内存分配效率更高效。
2019-06-15	- 内核版本升级到5.6.43。 - 支持审计功能。
2019-05-15	- 修复特定场景主备复制时延“Seconds_Behind_Master”不准确的问题。 - 修复“SQL_MODE”为“PAD_CHAR_TO_FULL_LENGTH”时的主备复制异常问题。
2019-01-15	- 修复只读执行flush操作后导致主备不一致，复制中断的问题。 - 修复REPAIR，OPTIMIZE等语句导致备机复制线程卡住问题。 - 修复grant select(column_name) 语句复制线程报错问题。
2018-11-15	- 内核版本升级到5.6.41。 - GTID开启时支持事务中创建和删除临时表。 - 支持表级别MTS并行回放。

日期	特性描述
2018-07-15	● 新特性 - 内核版本升级到5.6.40。 - 支持线程池。详见连接线程池。 - 支持CTS语法：支持create table xx select语法。 ● 修复以下问题 - 修复binlog和relay log名依赖PID文件名的问题。 - 修复“relay_log_basename”变量空指针导致备机卡住的问题。 - 修复force index语法在group_by xx order_by xx limit n1,n2场景下失效的问题。

2 主要内核功能介绍

2.1 连接线程池

功能介绍

当数据库并发连接过多时，存在大量资源竞争，MySQL服务器的性能会发生明显的下降。华为云RDS for MySQL提供了连接线程池，能够以少量活跃线程来服务大量数据库连接，将连接和执行解耦，在高并发场景下提高数据库性能。

功能特点

华为云RDS for MySQL连接线程池具有以下特点：

- 避免性能瓶颈，可以处理大量数据库连接，减少资源的争抢和上下文切换。
- 限制并发事务的数量，当数据库负载较高时，优先保障正在执行的事务。
- 避免由于连接得不到及时处理而出现线程异常问题。
- 当事务在等待IO和锁时，释放CPU资源以服务其他连接。

线程池相关操作

- 查询线程池参数

通过**show variables**命令查询线程池参数。

```
show variables like 'threadpool%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| threadpool_enabled | ON |
| threadpool_high_prio_tickets | 4294967295 |
| threadpool_idle_timeout | 60 |
| threadpool_long_conn_time | 2 |
| threadpool_max_threads | 100000 |
| threadpool_oversubscribe | 3 |
| threadpool_prio_kickup_timer | 1000 |
| threadpool_rec_launch_time | ON |
| threadpool_size | 128 |
| threadpool_slow_conn_log | ON |
| threadpool_slow_conn_log_interval | 30 |
| threadpool_slow_launch_time | 2 |
| threadpool_stall_limit | 500 |
+-----+-----+
```

表 2-1 线程池参数说明

参数名	参数描述
threadpool_enabled	开启或关闭线程池。
threadpool_high_prio_tickets	高优先级线程持有的tickets数量。
threadpool_idle_timeout	线程销毁前等待的空闲时间（单位：秒）。
threadpool_long_conn_time	登录时间超过此值，则在日志中打印此次登录信息。
threadpool_max_threads	线程池共能创建的线程上限。
threadpool_oversubscribe	每个线程组中最多能创建的额外线程数。
threadpool_prio_kickup_timer	低优先级队列中最长停留时间（单位：毫秒）。
threadpool_rec_launch_time	记录线程启动时间。
threadpool_size	线程组数量。
threadpool_slow_conn_log	是否记录慢登录到error log。
threadpool_slow_conn_log_interval	记录频率，记录一次慢登录后，在此时间间隔内不再记录。
threadpool_slow_launch_time	如登录、查询时间大于此值，则status中threadpool_slow_launch_request增加1。
threadpool_stall_limit	线程组忙碌检查间隔。

表 2-2 线程池可修改参数说明

参数名	是否是动态参数	数据类型	取值范围	说明
threadpool_enabled	是	boolean	[ON,OFF]	- ON：开启线程池。 - OFF：关闭线程池。
threadpool_oversubscribe	是	integer	[1,50]	每个线程组中最多能创建的额外线程数。
threadpool_size	是	integer	[1,512]	线程组数量。

- 查询线程池状态
通过show status命令查询线程池状态。

```
show status like 'threadpool%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Threadpool_active_connections | 65 |
| Threadpool_active_threads | 5 |
| Threadpool_avg_launch_time | 0 |
| Threadpool_dump_threads | 0 |
| Threadpool_idle_threads | 63 |
| Threadpool_running | ON |
| Threadpool_slow_launch_request | 0 |
| Threadpool_threads | 237 |
| Threadpool_threads_high_water_mark | 1075 |
| Threadpool_waiting_threads | 57 |
| Threadpool_worst_launch_time | 692548 |
+-----+-----+
```

表 2-3 线程池状态说明

状态	说明
Threadpool_active_connections	线程池中活跃连接数。
Threadpool_active_threads	线程池中活跃线程数。
Threadpool_avg_launch_time	平均等待时间（单位：毫秒）。
Threadpool_dump_threads	dump线程数量。
Threadpool_idle_thread	线程池空闲线程数。
Threadpool_running	线程池是否在运行。
Threadpool_slow_launch_request	超过slow_launch_request的次数。
Threadpool_threads	线程池总连接数。
Threadpool_threads_high_water_mark	历史高位线程数量。
Threadpool_waiting_threads	waiting线程池状态。
Threadpool_worst_launch_time	最坏启动时间（单位：毫秒）。

性能测试

表 2-4 不同线程的性能测试

模型	线程	线程池开关是否开启	QPS（每秒查询数）	latency（时延，单位是毫秒）
oltp_update_n on_index	32	是	5932.47	7.84
oltp_update_n on_index	64	是	10074.11	9.39
oltp_update_n on_index	128	是	18079.61	10.65
oltp_update_n on_index	256	是	27439.38	14.46
oltp_update_n on_index	512	是	33007.96	28.16
oltp_update_n on_index	1024	是	30282.13	51.94
oltp_update_n on_index	2048	是	29836.86	95.81

2.2 MDL 锁视图

MDL 锁视图简介

社区版MySQL如果不打开performance_schema开关，则无法获取表元数据锁（MDL）的详细信息，但通常情况下performance_schema默认关闭，当用户遇到类似“Waiting for metadata lock”的问题而阻塞DML或DDL后，由于无法确定各个会话之间的关联关系，只能重启实例，增加了解决问题的成本，对业务产生了较大影响。

在业务场景较复杂的情况下，一旦涉及对数据库元数据的互斥操作（如DDL、LOCK TABLE等），会频繁发生操作的会话被其他会话阻塞的问题，给用户带来很大的困扰。

针对以上问题，华为云RDS for MySQL推出了MDL锁视图特性，可以查看数据库各会话持有和等待的元数据锁信息，用户可以有效进行系统诊断，优化自身业务，有效降低对业务影响。

MDL 锁视图详解

MDL锁视图以系统表的形式呈现，该表位于“information_schema”下，表名称是“metadata_lock_info”。表结构如下所示。

```
desc information_schema.metadata_lock_info;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
```

THREAD_ID	bigint(20) unsigned	NO	0
LOCK_STATUS	varchar(24)	NO	
LOCK_MODE	varchar(24)	YES	NULL
LOCK_TYPE	varchar(30)	YES	NULL
LOCK_DURATION	varchar(30)	YES	NULL
TABLE_SCHEMA	varchar(64)	YES	NULL
TABLE_NAME	varchar(64)	YES	NULL

表 2-5 metadata_lock_info 字段

序号	字段名	字段定义	字段说明
0	THREA D_ID	bigint(20) unsigned	会话ID。
1	LOCK_S TATUS	varchar(24)	MDL锁的两种状态。 - PENDING：表示会话正在等待该MDL锁。 - GRANTED：表示会话已获得该MDL锁。
2	LOCK_ MODE	varchar(24)	加锁的模式，如MDL_SHARED 、MDL_EXCLUSIVE 、MDL_SHARED_READ、MDL_SHARED_WRITE等。
3	LOCK_T YPE	varchar(30)	MDL锁的类型，如Table metadata lock、Schema metadata lock、Global read lock、Tablespace lock等。
4	LOCK_D URATIO N	varchar(30)	MDL锁范围，取值如下： - MDL_STATEMENT：表示语句级别。 - MDL_TRANSACTION：表示事务级别。 - MDL_EXPLICIT：表示GLOBAL级别。
5	TABLE_ SCHEM A	varchar(64)	数据库名，对于部分GLOBAL级别的MDL锁，该值为空。
6	TABLE_ NAME	varchar(64)	表名，对于部分GLOBAL级别的MDL锁，该值为空。

MDL 锁视图使用示例

使用场景：长时间未提交事务，阻塞DDL，继而阻塞所有同表的操作。

表 2-6 MDL 锁视图示例

表名	会话			
	session2	session3	session4	session5

t1	begin; select * from t1;	-	-	-
t2	-	begin; select * from t2;	-	-
t3	-	-	truncate table t2; (blocked)	-
t4	-	-	-	begin; select * from t2; (blocked)

MDL 锁视图案例分析

问题描述

用户发现对表t2执行truncate操作一直被阻塞后，业务流程中对表t2执行查询操作也全部被阻塞。

排查分析

- 无MDL锁视图

当发现DDL语句被阻塞后，执行**show processlist**查看线程信息，结果如下所示。

```
show processlist;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | User | Host | db | Command | Time | State | Info |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 2 | root | localhost | test | Sleep | 73 | | Null |
| 3 | root | localhost | test | Sleep | 63 | | Null |
| 4 | root | localhost | Null | Query | 35 | Waiting for table metadata lock | truncate table test.t2 |
| 5 | root | localhost | test | Query | 17 | Waiting for table metadata lock | select * from test.t2 |
| 6 | root | localhost | test | Query | 0 | starting | show processlist |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

上述线程列表信息显示：

- ID=4的会话执行truncate操作时被其他会话持有的table metadata lock阻塞。
- ID=5的会话执行查询操作时同样被阻塞。
- 无法确定哪个会话阻塞了ID=4的会话和ID=5的会话。

此时，如果随机KILL其他会话会给线上业务带来很大风险，因此只能等待其他会话释放该MDL锁。

- 使用MDL锁视图

执行**select * from information_schema.metadata_lock_info**查看元数据锁信息，结果如下所示。

```
select * from information_schema.metadata_lock_info;
+-----+-----+-----+-----+-----+
| THREAD_ID | LOCK_STATUS | LOCK_MODE          | LOCK_TYPE          | LOCK_DURATION |
| TABLE_SCHEMA | TABLE_NAME |
+-----+-----+-----+-----+-----+
| 2         | GRANTED    | MDL_SHARED_READ    | Table metadata lock | MDL_TRANSACTION | |
| test      | t1         |
| 3         | GRANTED    | MDL_SHARED_READ    | Table metadata lock | MDL_TRANSACTION |
| test      | t2         |
| 4         | GRANTED    | MDL_INTENTION_EXCLUSIVE | Global read lock   | MDL_STATEMENT   |
|           |            |
| 4         | GRANTED    | MDL_INTENTION_EXCLUSIVE | Schema metadata lock |
| MDL_TRANSACTION | test      |
| 4         | PENDING    | MDL_EXCLUSIVE       | Table metadata lock |           | test      |
| t2        |
| 5         | PENDING    | MDL_SHARED_READ    | Table metadata lock |           | test      |
| t2        |
+-----+-----+-----+-----+-----+
```

结合show processlist的结果，从元数据锁视图中可以明显看出：

上述线程信息和元数据锁视图信息显示：

- THREAD_ID=4的会话正在等待表t2的metadata lock。
- THREAD_ID=3的会话持有表t2的metadata lock，该MDL锁为事务级别，因此只要THREAD_ID=3的会话的事务不提交，THREAD_ID=4的会话将会一直阻塞。

因此，用户只需在THREAD_ID=3的会话中执行命令commit或终止THREAD_ID=3的会话，便可以让业务继续运行。

2.3 在线扩展 varchar 类型字段

功能介绍

当字段类型是varchar时，原生MySQL只支持不跨越256字节的在线扩展（online），对于跨越256字节的扩展只支持复制方式（copy），扩展时需要锁表且禁止对数据表进行写操作。华为云RDS for MySQL支持跨越256字节的扩展，支持对任何长度的varchar类型的字段进行在线扩展操作。

支持的 RDS for MySQL 版本

建议您使用最新小版本的实例在线扩展varchar类型字段。更多操作，详见[升级内核小版本](#)。

表 2-7 在线扩展 varchar 类型字段支持的版本

varchar类型字段长度	RDS for MySQL 5.6	RDS for MySQL 5.7	RDS for MySQL 8.0
长度小于256字节	不支持	支持	支持
长度跨越256字节长度	支持	支持	不支持
长度大于256字节	不支持	支持	支持

扩展分类

- 长度小于256字节的varchar类型字段的在线扩展
create table t1(a varchar(10));
Query OK, 0 rows affected (0.03 sec)
alter table t1 modify a varchar(100),ALGORITHM=INPLACE, LOCK=NONE;
Query OK, 0 rows affected (0.06 sec)
Records: 0 Duplicates: 0 Warning: 0
- 长度跨越256字节长度的varchar类型字段的在线扩展
create table t1(a varchar(100));
Query OK, 0 rows affected (0.05 sec)
alter table t1 modify a varchar(300),ALGORITHM=INPLACE, LOCK=NONE;
Query OK, 0 rows affected (0.11 sec)
Records: 0 Duplicates: 0 Warning: 0
- 长度大于256字节的varchar类型字段的在线扩展
create table t1(a varchar(300));
Query OK, 0 rows affected (0.08 sec)
alter table t1 modify a varchar(500),ALGORITHM=INPLACE, LOCK=NONE;
Query OK, 0 rows affected (0.06 sec)
Records: 0 Duplicates: 0 Warning: 0

2.4 主动终止空闲事务

功能介绍

当一个事务长时间空闲且不提交，执行回滚操作时，会对数据库的资源 and 性能造成损耗。如果有大量的空闲事务长期不执行也不提交、回滚，尤其是在业务高峰期的时候，性能损耗会较为严重。

功能特点

华为云RDS for MySQL针对空闲事务有超时回滚断开连接机制，具有以下特点：

- 针对不同类型的事务有不同的参数进行控制。
- 当空闲事务超时后会被自动的回滚并断开连接。

参数介绍

通过show variables查询相关参数。

```
show variables like '%idle%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| idle_readonly_transaction_timeout | 0 |
| idle_transaction_timeout | 0 |
| idle_write_transaction_timeout | 0 |
+-----+-----+
```

表 2-8 参数总览

参数名	参数类型	取值范围	是否动态生效	参数描述
idle_readonly_transaction_timeout	integer	正整数	是	控制只读事务连接的超时时间（单位：秒），参数设置为0时不生效，即表示只读事务连接的超时时间没有限制。
idle_transaction_timeout	integer	正整数	是	控制一般空闲事务连接的超时时间（单位：秒），参数设置为0时不生效，即表示一般空闲事务的连接超时时间没有限制。 说明 “idle_readonly_transaction_timeout”和 “idle_write_transaction_timeout”参数的优先级高于 “idle_transaction_timeout”参数。 - 如果设置 “idle_readonly_transaction_timeout”或 “idle_write_transaction_timeout”参数值并使之生效，那么 “idle_transaction_timeout”参数将会失效。 - 如果仅设置 “idle_transaction_timeout”参数值并使之生效，那么在事务的读写操作时，超时时间均以 “idle_transaction_timeout”设置的参数值为准。
idle_write_transaction_timeout	integer	正整数	是	控制读写事务连接的超时时间（单位：秒），参数设置为0时不生效，即表示读写事务连接的超时时间没有限制。

应用场景

设置相关参数值如下，分别对下列参数适用的场景进行分析。

```
show variables like '%idle%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| idle_readonly_transaction_timeout | 5 |
| idle_transaction_timeout | 10 |
| idle_write_transaction_timeout | 15 |
+-----+-----+
```

- 设置idle_readonly_transaction_timeout
设置参数idle_readonly_transaction_timeout=5。
 - a. 使用begin开启事务，执行查询语句，查询结果如下：
begin;
Query OK, 0 rows affected (0.00 sec)

```
select * from t1;
+-----+-----+-----+
| a | b | c | d |
+-----+-----+-----+
| 1 | b | 303 | d |
+-----+-----+-----+
1 row in set (0.00 sec)
```

- b. 在idle_readonly_transaction_timeout设置的5s范围以外执行一次查询操作，结果如下：

```
select * from t1;
+-----+-----+-----+
| a | b | c | d |
+-----+-----+-----+
| 1 | b | 303 | d |
+-----+-----+-----+
1 row in set (0.00 sec)
select * from t1;
ERROR 2006(HY000): MySQL server has gone away
```

- 设置idle_transaction_timeout，idle_readonly_transaction_timeout，idle_write_transaction_timeout

设置参数idle_transaction_timeout=10，idle_readonly_transaction_timeout=0，idle_write_transaction_timeout=0。

```
show variables like '%idle%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| idle_readonly_transaction_timeout | 0 |
| idle_transaction_timeout | 10 |
| idle_write_transaction_timeout | 0 |
+-----+-----+
3 rows in set (0.01 sec)
```

– 只读事务

当idle_readonly_transaction_timeout=0时，idle_transaction_timeout参数生效。

- i. 使用**begin**开启事务，查询表数据，结果如下：

```
begin;
Query OK, 0 rows affected (0.00 sec)
select * from t1;
+-----+-----+-----+
| a | b | c | d |
+-----+-----+-----+
| 1 | b | 43 | d |
+-----+-----+-----+
1 row in set (0.00 sec)
```

- ii. 10s之后重复执行上述查询操作，结果如下：

```
select * from t1;
ERROR 2006(HY000): MySQL server has gone away
```

– 读写事务

当idle_write_transaction_timeout=0时，idle_transaction_timeout参数生效。

- i. 使用**begin**开启事务后，插入数据，在10s以内执行一条查询语句，结果如下：

```
begin;
Query OK, 0 rows affected (0.00 sec)
INSERT INTO t1(a,b,c,d) VALUES (1,'b',FLOOR( 1 + (RAND()*1000)) , 'd');
Query OK, 1 rows affected (0.00 sec)
select * from t1;
+-----+-----+-----+
| a | b | c | d |
+-----+-----+-----+
```

- ```
| 1 | b | 425 | d |
+-----+-----+-----+
1 row in set (0.00 sec)
```
- ii. 10s后执行一条查询语句，结果如下：  

```
select * from t1;
ERROR 2006(HY000): MySQL server has gone away
```
- iii. 之后单独执行一条查表语句，结果如下，表示此时事务已经回滚。  

```
select * from t1;
Empty set (0.00 sec)
```
- 设置idle\_write\_transaction\_timeout  
设置参数idle\_write\_transaction\_timeout=15。

a. 首先由begin开启事务并插入一条数据，结果如下：  

```
begin;
Query OK, 0 rows affected (0.00 sec)
INSERT INTO t1(a,b,c,d) VALUES (1,'b',FLOOR(1 + (RAND()*1000)) , 'd');
Query OK, 1 rows affected (0.00 sec)
```

b. 在idle\_write\_transaction\_timeout设置的15s范围以内执行一次查询操作，结果如下：  

```
select * from t1;
+-----+-----+-----+
| a | b | c | d |
+-----+-----+-----+
| 1 | b | 987 | d |
+-----+-----+-----+
1 row in set (0.00 sec)
```

c. 15s后再次执行查询，结果如下：  

```
select * from t1;
ERROR 2006(HY000): MySQL server has gone away
```

d. 重新连接数据库，执行一条查询语句，结果如下，表示此时事务已经回滚。  

```
select * from t1;
Empty set (0.00 sec)
```

## 2.5 DDL 进度显示

### 功能介绍

大表的DDL操作往往会比较耗时，社区版MySQL无法感知整个DDL的进行阶段与该阶段的进度，可能给用户带来很大的困扰。

针对以上问题，华为云RDS for MySQL推出了DDL进度显示特性，用户可以通过查询“INFORMATION\_SCHEMA.INNODB\_ALTER\_TABLE\_PROGRESS”表来实时查看当前的DDL语句所进行的阶段，以及该阶段的进度。

### 约束限制

该功能仅支持RDS for MySQL 5.7版本。

### 功能特点

表 2-9 INNODB\_ALTER\_TABLE\_PROGRESS 表字段详解

| 字段        | 说明   |
|-----------|------|
| THREAD_ID | 线程ID |

| 字段                | 说明                  |
|-------------------|---------------------|
| QUERY             | ALTER TABLE SQL语句内容 |
| START_TIME        | DDL开始时间             |
| ELAPSED_TIME      | 已执行的时间（s）           |
| ALTER_TABLE_STAGE | 当前执行的内容             |
| STAGE_COMPLETED   | 已完成的工作单位            |
| STAGE_ESTIMATED   | 预估的工作单位             |

一个DDL从开始到结束可能有下面几个阶段（ALTER\_TABLE\_STAGE字段的值）：

- stage/innodb/alter table (read PK and internal sort)：读取主键。
- stage/innodb/alter table (merge sort)：根据主键排序，这个过程可能较慢，因为会有临时文件生成。
- stage/innodb/alter table (insert)：将排序后的数据插入表中。
- stage/innodb/alter table (log apply index)：将DDL过程中客户执行的DML日志应用到本次创建/修改的索引。
- stage/innodb/alter table (flush)：数据刷盘。
- stage/innodb/alter table (log apply table)：将DDL过程中客户执行的DML日志应用到本次创建/修改的表。
- stage/innodb/alter table (end)：结束阶段。

## 2.6 优化阶段拦截写事务

### 功能介绍

在MySQL中，单个事务如果涉及大规模数据操作（如插入、更新或删除数百万行），会长时间持有锁并大量消耗I/O资源，严重影响后续业务的正常执行。用户往往不得不强制终止会话，触发回滚操作。回滚过程通常与原事务执行时间相当，期间仍会占用数据库锁和I/O资源，进一步阻塞其他工作负载。

为避免此类性能瓶颈，华为云RDS for MySQL在此类大规模写入事务执行前进行拦截，防止因数据变更过大导致系统整体性能下降，并及时提醒客户关注相关风险，从而保障数据库的稳定运行和服务的高可用性。此外，该特性开启后对数据库写性能会有轻微影响。

### 前提条件

- 对于RDS for MySQL 5.7，要求内核版本大于等于5.7.44.241207。
- 对于RDS for MySQL 8.0，要求内核版本大于等于8.0.41.250900。

配置参数

表 2-10 参数说明

| 参数名称                              | 是否动态参数 | 数据类型    | 取值范围                       | 描述                                                                                                                    |
|-----------------------------------|--------|---------|----------------------------|-----------------------------------------------------------------------------------------------------------------------|
| rds_max_data_mutation_num_per_dml | 是      | integer | [0, 10000-999999999 99999] | INSERT/UPDATE/DELETE可影响的行数的最大值，超过了该配置值，则提前拦截报错。<br>该参数是会话级参数。<br>默认值为0，表示不拦截。可配置0，以及[10000, 999999999999999]。         |
| rds_data_mutation_counting_mode   | 是      | enum    | [PRECISE, ESTIMATED]       | INSERT/UPDATE/DELETE影响的行数评估模式。在无法获取结果集行数的相对精确值时，ESTIMATED模式下依赖优化器估算值，PRECISE模式下则不做拦截。<br>该参数是会话级参数。<br>默认值为ESTIMATED。 |

使用限制

当前仅支持语句级的限制，以下语句在执行前会提前风险识别并拦截：

表 2-11 使用限制

| SQL语句              | 约束条件                                                                                                                      |
|--------------------|---------------------------------------------------------------------------------------------------------------------------|
| INSERT ... VALUES  | 仅对SQL语句VALUES增加的行数进行限制，非实际插入的行数。                                                                                          |
| REPLACE ... VALUES | 仅对SQL语句VALUES增加的行数进行限制，非实际插入的行数。                                                                                          |
| INSERT ... SELECT  | <ul style="list-style-type: none"><li>仅对优化器估算的SELECT语句输出行数进行限制。</li><li>SELECT语句结果未做缓存处理，且对应的执行计划包含多表JOIN时不做限制。</li></ul> |
| REPLACE ... SELECT | <ul style="list-style-type: none"><li>仅对优化器估算的SELECT语句输出行数进行限制。</li><li>SELECT语句结果未做缓存处理，且对应的执行计划包含多表JOIN时不做限制。</li></ul> |

| SQL语句                   | 约束条件                                                                                                                          |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| CREATE TABLE ... SELECT | <ul style="list-style-type: none"><li>• 仅对优化器估算的SELECT语句输出行数进行限制。</li><li>• SELECT语句结果未做缓存处理，且对应的执行计划包含多表JOIN时不做限制。</li></ul> |
| UPDATE TABLE ...        | <ul style="list-style-type: none"><li>• 仅支持单表。</li><li>• 仅对优化器估算的整表或者where条件过滤后输出行数进行限制。</li></ul>                            |
| DELETE FROM TABLE ...   | <ul style="list-style-type: none"><li>• 仅支持单表。</li><li>• 仅对优化器估算的整表或者where条件过滤后输出行数进行限制。</li></ul>                            |

由于SQL尚未实际执行，系统只能依赖优化器的估算值进行拦截判断，因此无法做到绝对精确。然而，在以下几类场景中，可以获取到相对准确的行数信息：

- 对表执行无限制条件的INSERT ... VALUES操作。
- INSERT/REPLACE ... SELECT或CREATE TABLE ... SELECT语句，其中 SELECT 的结果在插入前需要缓存（比如物化临时表、SORT缓存等）。
- UPDATE/DELETE语句在执行前需要缓存所有待更新行的ROW ID（比如临时文件、SORT 缓存等）。
- 单表查询可直接使用索引访问（如ref或者range），无额外过滤等操作。

在上述场景中，无论当前配置为PRECISE模式还是ESTIMATED模式，系统都会基于获取到的行数来决定是否进行拦截处理。

## 使用示例

以下是优化阶段拦截写事务的使用示例：

### 1. 表结构。

```
mysql> SHOW CREATE TABLE t1\G
***** 1. row *****
 Table: t1
 Create Table: CREATE TABLE `t1` (
 `id` int NOT NULL AUTO_INCREMENT,
 `category` int NOT NULL,
 `value` int NOT NULL,
 `flag` enum('active','inactive') DEFAULT 'active',
 `created_at` timestamp NOT NULL DEFAULT '2025-05-05 01:01:01',
 PRIMARY KEY (`id`),
 KEY `idx_category` (`category`),
 KEY `idx_value` (`value`),
 KEY `idx_flag` (`flag`),
 KEY `idx_category_value` (`category`,`value`)
) ENGINE=InnoDB AUTO_INCREMENT=120109 DEFAULT CHARSET=utf8mb4
 COLLATE=utf8mb4_0900_ai_ci
 1 row in set (0.00 sec)

mysql> CREATE TABLE t2 LIKE t1;
Query OK, 0 rows affected (0.15 sec)
```

### 2. INSERT ... SELECT语句拦截示例。

```
mysql> EXPLAIN FORMAT=TREE INSERT INTO t2 SELECT * FROM t1 WHERE category=1;
+-----+
```

```
| EXPLAIN
+-----+
| -> Insert into t2
| -> Index lookup on t1 using idx_category (category=1) (cost=6276 rows=60112)
|
+-----+
1 row in set (0.02 sec)

mysql> SET rds_max_data_mutation_num_per_dml=60111;
Query OK, 0 rows affected (0.00 sec)

单表走ref索引无过滤条件, 估算值超过门限, 'ESTIMATED'模式下会拦截
mysql> SET rds_data_mutation_counting_mode='ESTIMATED';
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t2 SELECT * FROM t1 WHERE category=1;
ERROR 4167 (HY000): The number [60112] of data mutations in the SQL statement has exceeded the
maximum limit. Please check the configuration parameter 'rds_max_data_mutation_num_per_dml'.
mysql>

单表走ref索引无过滤条件, 估算值超过门限, 'PRECISE'模式下会拦截
mysql> SET rds_data_mutation_counting_mode='PRECISE';
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t2 SELECT * FROM t1 WHERE category=1;
ERROR 4167 (HY000): The number [60112] of data mutations in the SQL statement has exceeded the
maximum limit. Please check the configuration parameter 'rds_max_data_mutation_num_per_dml'.
mysql>

mysql> EXPLAIN FORMAT=TREE INSERT INTO t2 SELECT * FROM t1 WHERE category=1 AND
flag='active';
+-----+
|
| EXPLAIN
|
+-----+
| -> Insert into t2
| -> Filter: (t1.flag = 'active') (cost=3270 rows=30056)
| -> Index lookup on t1 using idx_category (category=1) (cost=3270 rows=60112)
|
+-----+
1 row in set (0.00 sec)

mysql> SET rds_max_data_mutation_num_per_dml=30055;
Query OK, 0 rows affected (0.00 sec)

存在无法走索引的过滤条件, 且估算值超过门限, 'ESTIMATED'模式下会拦截
mysql> SET rds_data_mutation_counting_mode='ESTIMATED';
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t2 SELECT * FROM t1 WHERE category=1 AND flag='active';
ERROR 4167 (HY000): The number [30056] of data mutations in the SQL statement has exceeded the
maximum limit. Please check the configuration parameter 'rds_max_data_mutation_num_per_dml'.
mysql>
mysql> SET rds_data_mutation_counting_mode='PRECISE';
Query OK, 0 rows affected (0.00 sec)

存在无法走索引的过滤条件, 且估算值超过门限, 'PRECISE'模式下不拦截
mysql> INSERT INTO t2 SELECT * FROM t1 WHERE category=1 AND flag='active';
Query OK, 32275 rows affected (37.11 sec)
Records: 32275 Duplicates: 0 Warnings: 0

mysql>
```

### 3. UPDATE语句拦截示例。

```
mysql> EXPLAIN FORMAT=TREE SELECT * FROM t1 WHERE category=1;
+-----+
```

```
| EXPLAIN |
+-----+
| -> Index lookup on t1 using idx_category (category=1) (cost=4710 rows=44448)
|
+-----+
1 row in set (0.00 sec)

mysql> SET rds_max_data_mutation_num_per_dml=44447;
Query OK, 0 rows affected (0.00 sec)

单表走ref索引无过滤条件, 估算值超过门限, 'ESTIMATED'模式下会拦截
mysql> SET rds_data_mutation_counting_mode='ESTIMATED';
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE t1 SET value=value+100 WHERE category=1;
ERROR 4167 (HY000): The number [44448] of data mutations in the SQL statement has exceeded the
maximum limit. Please check the configuration parameter 'rds_max_data_mutation_num_per_dml'.
mysql>

单表走ref索引无过滤条件, 估算值超过门限, 'PRECISE'模式下会拦截
mysql> SET rds_data_mutation_counting_mode='PRECISE';
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE t1 SET value=value+100 WHERE category=1;
ERROR 4167 (HY000): The number [44448] of data mutations in the SQL statement has exceeded the
maximum limit. Please check the configuration parameter 'rds_max_data_mutation_num_per_dml'.
mysql>
mysql> EXPLAIN FORMAT=TREE SELECT * FROM t1 WHERE category=1 AND flag='active';
+-----+
| EXPLAIN |
+-----+
| -> Filter: (t1.flag = 'active') (cost=2487 rows=22224)
| -> Index lookup on t1 using idx_category (category=1) (cost=2487 rows=44448)
|
+-----+
1 row in set (0.01 sec)

mysql>
mysql> SET rds_max_data_mutation_num_per_dml=22223;
Query OK, 0 rows affected (0.00 sec)

mysql>

存在无法走索引的过滤条件, 且估算值超过门限, 'ESTIMATED'模式下会拦截
mysql> SET rds_data_mutation_counting_mode='ESTIMATED';
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE t1 SET value=value+100 WHERE category=1 AND flag='active';
ERROR 4167 (HY000): The number [22224] of data mutations in the SQL statement has exceeded the
maximum limit. Please check the configuration parameter 'rds_max_data_mutation_num_per_dml'.
mysql>

存在无法走索引的过滤条件, 且估算值超过门限, 'PRECISE'模式下不拦截
mysql> SET rds_data_mutation_counting_mode='PRECISE';
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE t1 SET value=value+100 WHERE category=1 AND flag='active';
Query OK, 32275 rows affected (43.55 sec)
Rows matched: 32275 Changed: 32275 Warnings: 0
```

#### 4. DELETE语句拦截示例。

```
mysql> EXPLAIN FORMAT=TREE SELECT * FROM t1 WHERE category=2;
+-----+
| EXPLAIN |
+-----+
| -> Index lookup on t1 using idx_category (category=2) (cost=6276 rows=60112)
|
```

```
+-----+
1 row in set (0.00 sec)

mysql>
mysql> SET rds_max_data_mutation_num_per_dml=60111;
Query OK, 0 rows affected (0.00 sec)

单表走ref索引无过滤条件, 估算值超过门限, 'ESTIMATED'模式下会拦截
mysql> SET rds_data_mutation_counting_mode='ESTIMATED';
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM t1 WHERE category=2;
ERROR 4167 (HY000): The number [60112] of data mutations in the SQL statement has exceeded the
maximum limit. Please check the configuration parameter 'rds_max_data_mutation_num_per_dml'.
mysql>

单表走ref索引无过滤条件, 估算值超过门限, 'PRECISE'模式下会拦截
mysql> SET rds_data_mutation_counting_mode='PRECISE';
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM t1 WHERE category=2;
ERROR 4167 (HY000): The number [60112] of data mutations in the SQL statement has exceeded the
maximum limit. Please check the configuration parameter 'rds_max_data_mutation_num_per_dml'.
mysql>
mysql> EXPLAIN FORMAT=TREE SELECT * FROM t1 WHERE category=2 AND flag='active';
+-----+
| EXPLAIN |
+-----+
| -> Filter: (t1.flag = 'active') (cost=3270 rows=30056)
| -> Index lookup on t1 using idx_category (category=2) (cost=3270 rows=60112)
|
+-----+
1 row in set (0.12 sec)

mysql> SET rds_max_data_mutation_num_per_dml=30055;
Query OK, 0 rows affected (0.00 sec)

存在无法走索引的过滤条件, 且估算值超过门限, 'ESTIMATED'模式下会拦截
mysql> SET rds_data_mutation_counting_mode='ESTIMATED';
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM t1 WHERE category=2 AND flag='active';
ERROR 4167 (HY000): The number [30056] of data mutations in the SQL statement has exceeded the
maximum limit. Please check the configuration parameter 'rds_max_data_mutation_num_per_dml'.
mysql>

存在无法走索引的过滤条件, 且估算值超过门限, 'PRECISE'模式下不拦截
mysql> SET rds_data_mutation_counting_mode='PRECISE';
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM t1 WHERE category=2 AND flag='active';
Query OK, 32040 rows affected (41.72 sec)
```