阿里云
云数据库 MySQL 版

AliSQL内核

文档版本：20190916

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告:** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 AliSQL功能概览

本文介绍AliSQL和其他版本的功能对比。

AliSQL介绍

AliSQL是阿里云深度定制的独立MySQL分支，除了社区版的所有功能外，AliSQL提供了类似于MySQL企业版的诸多功能，如企业级备份恢复、线程池、并行查询等，并且AliSQL还提供兼容Oracle的能力，如sequence引擎等。RDS for MySQL使用AliSQL内核，为用户提供了MySQL所有的功能，同时提供了企业级的安全、备份、恢复、监控、性能优化、只读实例等高级特性。

功能列表

分类	功能	社区版	官方企业版	AliSQL内核（5.7&8.0）	阿里云 RDS for MySQL
企业增值服务	24*7 支持	未提供	√	√	√
	紧急故障救援	未提供	√	√	√
	专家服务顾问支持	未提供	√	√	√
MySQL Features	MySQL Database Server	√	√	√	√
	MySQL Document Store	√	√	MySQL 8.0支持	MySQL 8.0支持
	MySQL Connectors	√	√	支持公开发行版	支持公开发行版
	MySQL Replication	√	√	√	√
	MySQL Router	√	√	MaxScale（MySQL 8.0支持）	数据库单租户代理
	MySQL Partitioning	√	√	√	√

分类	功能	社区版	官方企业版	AliSQL内核（5.7&8.0）	阿里云 RDS for MySQL
	Storage Engine	InnoDB MyISAM NDB	InnoDB MyISAM NDB	InnoDB X-Engine	InnoDB X-Engine
Oracle Compatibility	Sequence Engine	未提供	未提供	MySQL 8.0支持	MySQL 8.0支持
MySQL Enterprise Monitor	Enterprise Dashboard	未提供	√	开发中	Enhanced Monitor
	Enterprise Advisors	未提供	√	开发中	CloudDBA
	Query Analyzer	未提供	√	开发中	Performance Insight
	Replication Monitor	未提供	√	开发中	√
	Enhanced OS Metrics	未提供	未提供	未提供	Enhanced Monitor
MySQL Enterprise Backup	Hot backup for InnoDB	未提供	√	√	√
	Full, Incremental, Partial, Optimistic Backups	未提供	√	√	库表级备份
	Full, Partial, Selective, Hot Selective restore	未提供	√	√	库表级恢复
	Point-In-Time-Recovery	未提供	√	√	√
	Cross-Region Backup	未提供	未提供	未提供	跨地域备份

分类	功能	社区版	官方企业版	AliSQL内核（5.7&8.0）	阿里云 RDS for MySQL
	Recycle bin	未提供	未提供	MySQL 8.0支持	MySQL 8.0支持
	Flashback	未提供	未提供	√	√
MySQL Enterprise Security	Enterprise TDE	本地秘钥替换	√	BYOK TDE, Key Rotating	BYOK TDE, Key Rotating
	Enterprise Disk Data Encryption at Rest	未提供	未提供	未提供	BYOK 落盘加密
	Enterprise Encryption	SSL	√	SSL	SSL
	Enterprise Audit	未提供	√	SQL洞察	SQL洞察
	国内安全加密算法SM4	未提供	未提供	√	√
MySQL Enterprise Scalability	Thread Pool	未提供	√	MySQL 8.0支持	MySQL 8.0支持
	Enterprise Readonly Request Extention	未提供	未提供	√	只读实例
MySQL Enterprise Reliability	Zero Data Loss	未提供	未提供	√	三节点企业版
	SQL Outline	未提供	未提供	√	√
	Hot Massive Update	未提供	未提供	√	√
	Hot SQL Limit	未提供	未提供	√	√
	Hot SQL Firewall	未提供	未提供	√	√
MySQL Enterprise High-Availability	Enterprise Automatic Failover Switch	未提供	未提供	需要第三方HA机制	高可用版

分类	功能	社区版	官方企业版	AliSQL内核（5.7&8.0）	阿里云 RDS for MySQL
	InnoDB Cluster	√	√	√	三节点企业版
	Multi-Source Replication	√	√	√	只读实例高可用
	Cross-Region Standby	未提供	未提供	未提供	灾备实例

版本支持情况

版本	功能
MySQL 8.0	Performance Insight
	#unique_28
	Sequence Engine
	Statement Outline
	Statement Concurrency Control
	Thread Pool
	#unique_32
	优化实例锁状态
MySQL 5.7	合并官方5.7.26版本
	优化实例锁状态
MySQL 5.6	优化实例锁状态

2 AliSQL Release Notes

MySQL 8.0

20190915

Bug修复

修复Cmd_set_current_connection内存泄露问题。

20190816

- 新特性
  - Thread Pool：将线程和会话分离，在拥有大量会话的同时，只需要少量线程完成活跃会话的任务即可。
  - Statement Concurrency Control：通过控制并发数应对突发的数据库请求流量、资源消耗过高的语句访问以及SQL访问模型的变化，保证MySQL实例持续稳定运行。
  - Statement Outline：利用Optimizer Hint和Index Hint让MySQL稳定执行计划。
  - #unique_28：临时将删除的表转移到回收站，还可以设置保留的时间，方便您找回数据。
  - Sequence Engine：简化获取序列值的复杂度。
  - Purge Large File Asynchronously：删除单个表空间时，会将表空间文件重命名为临时文件，等待异步清除进程清理临时文件。
  - Performance Insight：专注于实例负载监控、关联分析、性能调优的利器，帮助您迅速评估数据库负载，找到性能问题的源头，提升数据库的稳定性。
  - 优化实例锁状态：实例锁定状态下，可以drop或truncate表。
- Bug修复
  - 修复文件大小计算错误的问题。
  - 修复偶尔出现的内存空闲后再次使用的问题。
  - 修复主机缓存大小为0时的崩溃问题。
  - 修复隐式主键与CTS语句的冲突问题。
  - 修复慢查询导致的slog出错问题。

20190601

- 性能优化
  - 缩短日志表MDL范围，减少MDL阻塞的可能性。
  - 重构终止选项的代码。

- Bug修复
  - 修复审计日志中没有记录预编译语句的问题。
  - 屏蔽无效表名的错误日志。

MySQL 5.7

20190915

新特性

Thread Pool：将线程和会话分离，在拥有大量会话的同时，只需要少量线程完成活跃会话的任务即可。

20190815

- 新特性
  - Purge Large File Asynchronously：删除单个表空间时，会将表空间文件重命名为临时文件，等待异步清除进程清理临时文件。
  - Performance Insight：专注于实例负载监控、关联分析、性能调优的利器，帮助您迅速评估数据库负载，找到性能问题的源头，提升数据库的稳定性。
  - 优化实例锁状态：实例锁定状态下，可以drop或truncate表。
- Bug修复
  - 禁止在`set rds_current_connection`命令中设置rds_prepare_begin_id。
  - 允许更改已锁定用户的信息。
  - 禁止用关键字actual作为表名。
  - 修复慢日志导致时间字段溢出的问题。

20190510版本

新特性：允许在事务内创建临时表。

20190319版本

新特性：支持在handshake报文内代理设置threadID。

20190131版本

- 升级到官方5.7.25版本。
- 关闭内存管理功能jemalloc。
- 修复内部变量net_lenth_size计算错误问题。

20181226版本

- 新特性：支持动态修改binlog-row-event-max-size，加速无主键表的复制。
- 修复Proxy实例内存申请异常的问题。

20181010版本

- 支持隐式主键。
- 加快无主键表的主备复制。
- 支持Native AIO，提升I/O性能。

20180431版本

新特性：

- 支持高可用版。
- 支持#unique_36。
- 增强对处于快照备份状态的实例的保护。

MySQL 5.6

20190815

优化实例锁状态：实例锁定状态下，可以drop或truncate表。

20190130版本

修复部分可能导致系统不稳定的bug。

20181010版本

添加参数rocksdb_ddl_commit_in_the_middle（MyRocks）。如果这个参数被打开，部分DDL在执行过程中将会执行commit操作。

201806** （5.6.16）版本

新特性：slow log精度提升为微秒。

20180426（5.6.16）版本

- 新特性：引入隐藏索引，支持将索引设置为不可见，详情请参见参考文档。
- 修复备库apply线程的bug。
- 修复备库apply分区表更新时性能下降问题。
- 修复TokuDB下alter table comment重建整张表问题，详情请参见参考文档。
- 修复由show slave status/show status可能触发的死锁问题。

20171205（5.6.16）版本

- 修复OPTIMIZE TABLE和ONLINE ALTER TABLE同时执行时会触发死锁的问题。

- 修复SEQUENCE与隐含主键冲突的问题。
- 修复SHOW CREATE SEQUENCE问题。
- 修复TokuDB引擎的表统计信息错误。
- 修复并行OPTIMIZE表引入的死锁问题。
- 修复QUERY_LOG_EVENT中记录的字符集问题。
- 修复信号处理引起的数据库无法停止问题，详情请参见参考文档。
- 修复RESET MASTER引入的问题。
- 修复备库陷入等待的问题。
- 修复SHOW CREATE TABLE可能触发的进程崩溃问题。

20170927（5.6.16）版本

修复TokuDB表查询时使用错误索引问题。

20170901（5.6.16）版本

- 新特性：
  - 升级SSL加密版本到TLS 1.2，详情请参见参考文档。
  - 支持Sequence。
- 修复NOT IN查询在特定场景下返回结果集有误的问题。

20170530 (5.6.16)版本

新特性：支持高权限账号Kill其他账号下的连接。

20170221（5.6.16）版本

新特性：支持#unique_37。

MySQL 5.5

20181212

修复调用系统函数gettimeofday(2) 返回值不准确的问题。该系统函数返回值为时间，常用来计算等待超时，时间不准确时会导致一些操作永不超时。

3 Statement Concurrency Control

为了应对突发的数据库请求流量、资源消耗过高的语句访问以及SQL访问模型的变化， 保证MySQL实例持续稳定运行，阿里云提供基于语句规则的并发控制CCL（Concurrency Control），并提供了工具包（DBMS_CCL）便于您快捷使用。

前提条件

实例版本为RDS for MySQL 8.0。

注意事项

- CCL的操作不产生Binlog，所以CCL的操作只影响当前实例。例如主实例进行CCL操作，不会同步到备实例、只读实例或灾备实例。
- CCL提供超时机制以应对DML导致事务锁死锁，等待中的线程也会响应事务超时和线程KILL操作以应对死锁。

功能设计

CCL规则定义了如下三个维度的特征：

- SQL command

  SQL命令类型，例如SELECT、UPDATE、INSERT、DELETE等。
- Object

  SQL命令操作的对象，例如TABLE、VIEW等。
- keywords

  SQL命令的关键字。

创建CCL规则表

AliSQL设计了一个系统表（concurrency_control）保存CCL规则，系统启动时会自动创建该表，无需您手动创建。这里提供表的创建语句供您参考：

```
CREATE TABLE `concurrency_control` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT,
  `Type` enum('SELECT','UPDATE','INSERT','DELETE') NOT NULL DEFAULT '
SELECT',
  `Schema_name` varchar(64) COLLATE utf8_bin DEFAULT NULL,
  `Table_name` varchar(64) COLLATE utf8_bin DEFAULT NULL,
  `Concurrency_count` bigint(20) DEFAULT NULL,
  `Keywords` text COLLATE utf8_bin,
  `State` enum('N','Y') NOT NULL DEFAULT 'Y',
  `Ordered` enum('N','Y') NOT NULL DEFAULT 'N',
  PRIMARY KEY (`Id`)
) /*!50100 TABLESPACE `mysql` */ ENGINE=InnoDB
DEFAULT CHARSET=utf8 COLLATE=utf8_bin
```

```
STATS_PERSISTENT=0 COMMENT='Concurrency control'
```

参数	说明
Id	CCL规则ID。
Type	SQL command，即SQL命令类型。
Schema_name	数据库名。
Table_name	数据库内的表名。
Concurrency_count	并发数。
Keywords	关键字，多个关键字用英文分号（;）分隔。
State	本规则是否启用。
Ordered	Keywords中多个关键字是否按顺序匹配。

管理CCL规则

为了便捷地管理CCL规则，AliSQL在DBMS_CCL中定义了四个本地存储规则。详细说明如下：

- add_ccl_rule

  增加规则。命令如下：

  ```
  dbms_ccl.add_ccl_rule('<Type>','<Schema_name>','<Table_name>',<
  Concurrency_count>,'<Keywords>');
  ```

  示例：

  SELECT语句的并发数为10。

  ```
  mysql> call dbms_ccl.add_ccl_rule('SELECT', '', '', 10, '');
  ```

  SELECT语句中出现关键字key1的并发数为20。

  ```
  mysql> call dbms_ccl.add_ccl_rule('SELECT', '', '', 20, 'key1');
  ```

  test.t表的SELECT语句的并发数为20。

  ```
  mysql> call dbms_ccl.add_ccl_rule('SELECT', '', '', 10, '');
  ```

   说明：

  Id越大，规则的优先级越高。

- **del_ccl_rule**

  删除规则。命令如下：

  ```
  dbms_ccl.del_ccl_rule(<Id>);
  ```

  示例：

  删除规则ID为15的CCL规则。

  ```
  mysql> call dbms_ccl.del_ccl_rule(15);
  ```

  说明：

  如果删除的规则不存在，系统会报相应的警告，您可以使用`show warnings;`查看警告内容。

  ```
  mysql> call dbms_ccl.del_ccl_rule(100);
    Query OK, 0 rows affected, 2 warnings (0.00 sec)

  mysql> show warnings;
  +---------+------
  +-----------------------------------------------------+
  | Level   | Code | Message
     |
  +---------+------
  +-----------------------------------------------------+
  | Warning | 7514 | Concurrency control rule 100 is not found in
  table |
  | Warning | 7514 | Concurrency control rule 100 is not found in
  cache |
  +---------+------
  +-----------------------------------------------------+
  ```

- **show_ccl_rule**

  查看内存中已启用规则。命令如下：

  ```
  dbms_ccl.show_ccl_rule();
  ```

  示例：

  ```
  mysql> call dbms_ccl.show_ccl_rule();
  +------+--------+--------+-------+-------+-------
  +-------------------+---------+---------+----------+----------+
  | ID   | TYPE   | SCHEMA | TABLE | STATE | ORDER | CONCURRENCY_COUNT
   | MATCHED | RUNNING | WAITTING | KEYWORDS |
  +------+--------+--------+-------+-------+-------
  +-------------------+---------+---------+----------+----------+
  |   17 | SELECT | test   | t     | Y     | N     |                30
   |       0 |       0 |       0 |          |
  |   16 | SELECT |        |       | Y     | N     |                20
   |       0 |       0 |       0 | key1     |
  |   18 | SELECT |        |       | Y     | N     |                10
   |       0 |       0 |       0 |          |
  ```

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

关于MATCHED、RUNNING和WAITTING的说明如下。

参数	说明
MATCHED	规则匹配成功次数。
RUNNING	此规则下正在并发执行的线程数。
WAITTING	此规则下正在等待执行的线程数。

- flush_ccl_rule

  如果您直接操作了表concurrency_control修改规则，规则不能立即生效，您需要让规则重新生效。命令如下：

  ```
  dbms_ccl.flush_ccl_rule();
  ```

  示例：

  ```
  mysql> update mysql.concurrency_control set CONCURRENCY_COUNT = 15
  where Id = 18;
  Query OK, 1 row affected (0.00 sec)
  Rows matched: 1  Changed: 1  Warnings: 0

  mysql> call dbms_ccl.flush_ccl_rule();
  Query OK, 0 rows affected (0.00 sec)
  ```

功能测试

- 测试规则

  设计如下三条规则对应三个维度：

  ```
  call dbms_ccl.add_ccl_rule('SELECT', 'test', 'sbtest1', 3, '');  //
  SELECT命令操作表sbtest1并发数为3
  call dbms_ccl.add_ccl_rule('SELECT', '', '', 2, 'sbtest2');       //
  SELECT命令关键字sbtest2并发数为2
  call dbms_ccl.add_ccl_rule('SELECT', '', '', 2, '');            //
  SELECT命令并发数为2
  ```

- 测试场景

  使用sysbench进行测试，场景如下：

  - 64 threads
  - 4 tables
  - select.lua

- **测试结果**

  查看规则并发数情况如下：

```
mysql> call dbms_ccl.show_ccl_rule();
+------+--------+--------+---------+-------+-------
+-------------------+---------+---------+----------+----------+
| ID   | TYPE   | SCHEMA | TABLE   | STATE | ORDER | CONCURRENC
Y_COUNT | MATCHED | RUNNING | WAITTING | KEYWORDS |
+------+--------+--------+---------+-------+-------
+-------------------+---------+---------+----------+----------+
|   20 | SELECT | test   | sbtest1 | Y     | N     |
  3 |     389 |       3 |         9 |          |
|   21 | SELECT |        |         | Y     | N     |
  2 |     375 |       2 |        14 | sbtest2  |
|   22 | SELECT |        |         | Y     | N     |
  2 |     519 |       2 |        34 |          |
+------+--------+--------+---------+-------+-------
+-------------------+---------+---------+----------+----------+
3 rows in set (0.00 sec)
```

  查看RUNNING列，符合预期的并行数量。

4 Statement Outline

生产环境中，SQL语句的执行计划经常会发生改变，导致数据库不稳定。阿里云利用Optimizer Hint和Index Hint让MySQL稳定执行计划，该方法称为Statement Outline，并提供了工具包（DBMS_OUTLN）便于您快捷使用。

前提条件

实例版本为RDS for MySQL 8.0。

功能设计

Statement Outline支持官方MySQL 8.0的所有hint类型，分为如下两类：

- Optimizer Hint

  根据作用域和hint对象，分为Global level hint、Table/Index level hint、Join order hint等。详情请参见MySQL官网。

- Index Hint

  根据Index Hint的类型和范围进行分类。详情请参见MySQL官网

Statement Outline表介绍

AliSQL内置了一个系统表（outline）保存hint，系统启动时会自动创建该表，无需您手动创建。这里提供表的创建语句供您参考：

```
CREATE TABLE `mysql`.`outline` (
  `Id` bigint(20) NOT NULL AUTO_INCREMENT,
  `Schema_name` varchar(64) COLLATE utf8_bin DEFAULT NULL,
  `Digest` varchar(64) COLLATE utf8_bin NOT NULL,
  `Digest_text` longtext COLLATE utf8_bin,
  `Type` enum('IGNORE INDEX','USE INDEX','FORCE INDEX','OPTIMIZER')
CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
  `Scope` enum('','FOR JOIN','FOR ORDER BY','FOR GROUP BY') CHARACTER
SET utf8 COLLATE utf8_general_ci DEFAULT '',
  `State` enum('N','Y') CHARACTER SET utf8 COLLATE utf8_general_ci NOT
 NULL DEFAULT 'Y',
  `Position` bigint(20) NOT NULL,
  `Hint` text COLLATE utf8_bin NOT NULL,
  PRIMARY KEY (`Id`)
) /*!50100 TABLESPACE `mysql` */ ENGINE=InnoDB
DEFAULT CHARSET=utf8 COLLATE=utf8_bin STATS_PERSISTENT=0 COMMENT='
Statement outline'
```

参数说明如下。

参数	说明
Id	Outline ID。

参数	说明
Schema_name	数据库名。
Digest	Digest_text进行hash计算得到的64字节的hash字符串。
Digest_text	SQL语句的特征。
Type	· Optimizer Hint中，hint类型的取值为OPTIMIZER。 · Index Hint中，hint类型的取值为USE INDEX、FORCE INDEX或IGNORE INDEX。
Scope	仅Index Hint需要提供，分为如下三类： · FOR GROUP BY · FOR ORDER BY · FOR JOIN 空串表示所有类型的Index Hint。
State	本规则是否启用。
Position	· Optimizer Hint中，Position表示Query Block，因为所有的Optimizer Hint必须作用到 Query Block上，所以，Position从1开始，hint作用在语句的第几个关键字上，Position就是几。 · Index Hint中，Position表示表的位置，也是从1开始，hint作用在第几个表上，Position就是几。
Hint	· Optimizer Hint中，Hint表示完整的hint字符串，例如/*+ MAX_EXECUTION_TIME(1000) */。 · Index Hint中，Hint表示索引名字的列表，例如ind_1,ind_2。

管理Statement Outline

为了便捷地管理Statement Outline，AliSQL在DBMS_OUTLN中定义了六个本地存储规则。详细说明如下：

· add_optimizer_outline

增加Optimizer Hint。命令如下：

```
dbms_outln.add_optimizer_outline('<Schema_name>','<Digest>','<
query_block>','<hint>','<query>');
```

说明：

Digest和Query（原始SQL语句）可以任选其一。如果填写Query，DBMS_OUTLN会计算Digest和Digest_text。

示例：

```
CALL DBMS_OUTLN.add_optimizer_outline("outline_db", '', 1, '/*+
MAX_EXECUTION_TIME(1000) */',
                                      "select * from t1 where id = 1
");
```

- **add_index_outline**

  增加Index Hint。命令如下：

  ```
  dbms_outln.add_index_outline('<Schema_name>','<Digest>',<Position
  >,'<Type>','<Hint>','<Scope>','<Query>');
  ```

  说明：

  Digest和Query（原始SQL语句）可以任选其一。如果填写Query，DBMS_OUTLN会计算Digest和Digest_text。

  示例：

  ```
  call dbms_outln.add_index_outline('outline_db', '', 1, 'USE INDEX',
   'ind_1', '',
                                    "select * from t1 where t1.col1 =1
  and t1.col2 ='xpchild'");
  ```

- **preview_outline**

  查看匹配Statement Outline的情况，可用于手动验证。命令如下：

  ```
  dbms_outln.preview_outline('<Schema_name>','<Query>');
  ```

  示例：

  ```
  mysql> call dbms_outln.preview_outline('outline_db', "select * from
  t1 where t1.col1 =1 and t1.col2 ='xpchild'");
  +------------
  +-----------------------------------------------------------------
  +------------+------------+-------+--------------------+
  | SCHEMA     | DIGEST
              | BLOCK_TYPE | BLOCK_NAME | BLOCK | HINT
   |
  +------------
  +-----------------------------------------------------------------
  +------------+------------+-------+--------------------+
  | outline_db | b4369611be7ab2d27c85897632576a04bc08f50b928a1d735b
  62d0a140628c4c | TABLE      | t1         |     1 | USE INDEX (`ind_1
  `) |
  +------------
  +-----------------------------------------------------------------
  +------------+------------+-------+--------------------+
  ```

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

- **show_outline**

  展示Statement Outline在内存中命中的情况。命令如下：

```
dbms_outln.show_outline();
```

  示例：

```
mysql> call dbms_outln.show_outline();
+------+------------
+-------------------------------------------------------------------------
+-----------+-------+------
+-----------------------------------------------------+------
+----------
+-------------------------------------------------------------------------------
+
| ID   | SCHEMA     | DIGEST
                   | TYPE      | SCOPE | POS  | HINT
                                    | HIT  | OVERFLOW | DIGEST_TEXT

  |
+------+------------
+-------------------------------------------------------------------------
+-----------+-------+------
+-----------------------------------------------------+------
+----------
+-------------------------------------------------------------------------------
+
|   33 | outline_db | 36bebc61fce7e32b93926aec3fdd790dad5d8951
07e2d8d3848d1c60b74bcde6 | OPTIMIZER |       |    1 | /*+ SET_VAR(
foreign_key_checks=OFF) */                |    1 |        0 | SELECT
 * FROM `t1` WHERE `id` = ?
          |
|   32 | outline_db | 36bebc61fce7e32b93926aec3fdd790dad5d8951
07e2d8d3848d1c60b74bcde6 | OPTIMIZER |       |    1 | /*+ MAX_EXECUT
ION_TIME(1000) */                         |    2 |        0 | SELECT
 * FROM `t1` WHERE `id` = ?
          |
|   34 | outline_db | d4dcef634a4a664518e5fb8a21c6ce9b79fccb44
b773e86431eb67840975b649 | OPTIMIZER |       |    1 | /*+ BNL(t1,t2)
 */                                     |    1 |        0 | SELECT `
t1` . `id` , `t2` . `id` FROM `t1` , `t2`
        |
|   35 | outline_db | 5a726a609b6fbfb76bb8f9d2a24af913a2b9d07f
015f2ee1f6f2d12dfad72e6f | OPTIMIZER |       |    2 |  /*+ QB_NAME(
subq1) */                                   |    2 |        0 | SELECT
 * FROM `t1` WHERE `t1` . `col1` IN ( SELECT `col1` FROM `t2` )
          |
|   36 | outline_db | 5a726a609b6fbfb76bb8f9d2a24af913a2b9d07f
015f2ee1f6f2d12dfad72e6f | OPTIMIZER |       |    1 | /*+ SEMIJOIN
(@subq1 MATERIALIZATION, DUPSWEEDOUT) */  |    2 |        0 | SELECT
 * FROM `t1` WHERE `t1` . `col1` IN ( SELECT `col1` FROM `t2` )
          |
|   30 | outline_db | b4369611be7ab2d27c85897632576a04bc08f50b
928a1d735b62d0a140628c4c | USE INDEX |       |    1 | ind_1
                                          |    3 |        0 | SELECT
 * FROM `t1` WHERE `t1` . `col1` = ? AND `t1` . `col2` = ?
          |
|   31 | outline_db | 33c71541754093f78a1f2108795cfb45f8b15ec5
d6bff76884f4461fb7f33419 | USE INDEX |       |    2 | ind_2
```

```
                                                |    1 |         0 | SELECT
 * FROM `t1` , `t2` WHERE `t1` . `col1` = `t2` . `col1` AND `t2` . `
col2` = ? |
+------+------------
+-------------------------------------------------------------------------
+-----------+-------+------
+-----------------------------------------------------------+------
+----------
+-------------------------------------------------------------------------------------
+
7 rows in set (0.00 sec)
```

关于HIT和OVERFLOW的说明如下。

参数	说明
HIT	此Statement Outline命中的次数。
OVERFLOW	此Statement Outline没有找到Query block或相应的表的次数。

- del_outline

  删除内存和表中的某一条Statement Outline。命令如下：

  ```
  dbms_outln.del_outline(<Id>);
  ```

  示例：

  ```
  mysql> call dbms_outln.del_outline(32);
  ```

  说明：

  如果删除的规则不存在，系统会报相应的警告，您可以使用`show warnings;`查看警告内容。

  ```
  mysql> call dbms_outln.del_outline(1000);
  Query OK, 0 rows affected, 2 warnings (0.00 sec)

  mysql> show warnings;
  +---------+------+-----------------------------------------------+
  | Level   | Code | Message                                       |
  +---------+------+-----------------------------------------------+
  | Warning | 7521 | Statement outline 1000 is not found in table |
  | Warning | 7521 | Statement outline 1000 is not found in cache |
  +---------+------+-----------------------------------------------+
  ```

```
 2 rows in set (0.00 sec)
```

- **flush_outline**

  如果您直接操作了表outline修改Statement Outline，您需要让Statement Outline重新生效。命令如下：

```
dbms_outln.flush_outline();
```

  示例：

```
mysql> update mysql.outline set Position = 1 where Id = 18;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> call dbms_outln.flush_outline();
Query OK, 0 rows affected (0.01 sec)
```

功能测试

验证Statement Outline是否有效果，有如下两种方法：

- **通过preview_outline进行预览。**

```
mysql> call dbms_outln.preview_outline('outline_db', "select * from
t1 where t1.col1 =1 and t1.col2 ='xpchild'");
+------------
+------------------------------------------------------------------
+------------+------------+-------+--------------------+
| SCHEMA     | DIGEST
             | BLOCK_TYPE | BLOCK_NAME | BLOCK | HINT
 |
+------------
+------------------------------------------------------------------
+------------+------------+-------+--------------------+
| outline_db | b4369611be7ab2d27c85897632576a04bc08f50b928a1d735b
62d0a140628c4c | TABLE      | t1         |     1 | USE INDEX (`ind_1
`) |
+------------
+------------------------------------------------------------------
+------------+------------+-------+--------------------+
1 row in set (0.01 sec)
```

- **直接使用explain查看。**

```
mysql> explain select * from t1 where t1.col1 =1 and t1.col2 ='
xpchild';
+----+-------------+-------+------------+------+---------------
+-------+---------+-------+------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key
   | key_len | ref   | rows | filtered | Extra       |
+----+-------------+-------+------------+------+---------------
+-------+---------+-------+------+----------+-------------+
|  1 | SIMPLE      | t1    | NULL       | ref  | ind_1         |
ind_1 | 5       | const |    1 |   100.00 | Using where |
+----+-------------+-------+------------+------+---------------
+-------+---------+-------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> show warnings;
+-------+------
+-------------------------------------------------------------------------------
+
| Level | Code | Message


                                                                        |
+-------+------
+-------------------------------------------------------------------------------
+
| Note  | 1003 | /* select#1 */ select `outline_db`.`t1`.`id` AS `
id`,`outline_db`.`t1`.`col1` AS `col1`,`outline_db`.`t1`.`col2` AS `
col2` from `outline_db`.`t1` USE INDEX (`ind_1`) where ((`outline_db
`.`t1`.`col1` = 1) and (`outline_db`.`t1`.`col2` = 'xpchild')) |
+-------+------
+-------------------------------------------------------------------------------
+
1 row in set (0.00 sec)
```

5 Recycle Bin

由于DDL语句无法回滚，开发或运维人员如果误操作（例如DROP TABLE）可能会导致数据丢失。阿里云支持回收站（Recycle Bin）功能，临时将删除的表转移到回收站，还可以设置保留的时间，方便您找回数据，同时提供了工具包（DBMS_RECYCLE）便于您快捷使用。

Recycle Bin参数

Recycle Bin设计了如下五个参数。

参数	说明
recycle_bin	是否打开回收站功能，包括session级别和global级别。
recycle_bin_retention	回收站保留时间，单位：秒。默认为604800，即一周。
recycle_scheduler	是否打开回收站的异步清理任务线程。
recycle_scheduler_interval	回收站异步清理任务线程的轮询间隔，单位：秒。默认为30。
recycle_scheduler_purge_table_print	是否打印异步清理现场工作的详细日志。

Recycle Bin介绍

- 回收/清理机制
  - 回收机制

    执行DROP TABLE/DATABASE语句时，只保留相关的表对象，并移动到专门的recycle bin目录中。其它对象的删除策略如下：

    ■ 如果是与表无关的对象，根据操作语句决定是否保留，不做回收。

    ■ 如果是表的附属对象，可能会修改表数据的，做删除处理，例如Trigger和Foreign key。但Column statistics不做清理，随表进入回收站。
  - 清理机制

    回收站会启动一个后台线程，来异步清理超过recycle_bin_retention时间的表对象。在清理回收站表的时候，如果遇到大表，会再启动一个后台线程异步删除大表。

- 权限

  RDS for MySQL实例启动时，会初始化一个名为__recycle_bin__的数据库，作为回收站使用的专有数据库。__recycle_bin__是系统级数据库，您无法直接进行修改和删除。

  对于回收站内的表，虽然您无法直接执行drop table语句，但是可以使用call dbms_recycle.purge_table('<TABLE>');进行清理。

  
  说明:
  账号在原表和回收站表都需要具有DROP权限。

- 回收站表命名规则

  Recycle Bin会从不同的数据库回收到统一的__recycle_bin__数据库中，所以需要保证目标表表名唯一，所以定义了如下命名格式：

  ```
  "__" + <Storage Engine> + <SE private id>
  ```

  参数说明如下。

参数	说明
Storage Engine	存储引擎名称。
SE private id	存储引擎为每一个表生成的唯一值。例如在InnoDB引擎中就是table id。

- 独立回收

  回收的设置只会影响该实例本身，不会影响到binlog复制到的节点（备实例、只读实例和灾备实例）上。例如我们可以在主实例上设置回收，保留7天；在备实例上设置回收，保留14天。

  
  说明:
  回收站保留周期不同，将导致实例的空间占用差别比较大。

注意事项

- 如果回收站数据库和待回收的表跨了文件系统，执行drop table语句将会搬迁表空间文件，耗时较长。
- 如果Tablespace为General，可能会存在多个表共享同一个表空间的情况，当回收其中一张表的时候，不会搬迁相关的表空间文件。

前提条件

实例版本为RDS for MySQL 8.0。

管理Recycle Bin

AliSQL在DBMS_RECYCLE中定义了两个管理接口。详细说明如下：

- show_tables

  展示回收站中所有临时保存的表。命令如下：

  ```
  call dbms_recycle.show_tables();
  ```

  示例：

  ```
  mysql> call dbms_recycle.show_tables();
  +-----------------+--------------+---------------+-------------
  +---------------------+---------------------+
  | SCHEMA          | TABLE         | ORIGIN_SCHEMA | ORIGIN_TABLE |
  RECYCLED_TIME       | PURGE_TIME          |
  +-----------------+--------------+---------------+-------------
  +---------------------+---------------------+
  | __recycle_bin__ | __innodb_1063 | product_db    | t1           |
  2019-08-08 11:01:46 | 2019-08-15 11:01:46 |
  | __recycle_bin__ | __innodb_1064 | product_db    | t2           |
  2019-08-08 11:01:46 | 2019-08-15 11:01:46 |
  | __recycle_bin__ | __innodb_1065 | product_db    | parent       |
  2019-08-08 11:01:46 | 2019-08-15 11:01:46 |
  | __recycle_bin__ | __innodb_1066 | product_db    | child        |
  2019-08-08 11:01:46 | 2019-08-15 11:01:46 |
  +-----------------+--------------+---------------+-------------
  +---------------------+---------------------+
  4 rows in set (0.00 sec)
  ```

参数	说明
SCHEMA	回收站的数据库名。
TABLE	进入回收站后的表名。
ORIGIN_SCHEMA	原数据库名。
ORIGIN_TABLE	原表名。
RECYCLED_TIME	回收时间。
PURGE_TIME	预计从回收站删除的时间。

- purge_table

  手动清理回收站中的表。命令如下：

  ```
  call dbms_recycle.purge_table('<TABLE>');
  ```

  说明：

  - TABLE为进入回收站后的表名。

- **账号在原表和回收站表都需要具有DROP权限。**

示例：

```
mysql> call dbms_recycle.purge_table('__innodb_1063');
Query OK, 0 rows affected (0.01 sec)
```

6 Thread Pool

为了发挥出RDS的最佳性能，阿里云提供线程池（Thread Pool）功能，将线程和会话分离，在拥有大量会话的同时，只需要少量线程完成活跃会话的任务即可。

优势

MySQL默认的线程使用模式是会话独占模式，每个会话都会创建一个独占的线程。当有大量的会话存在时，会导致大量的资源竞争，大量的系统线程调度和缓存失效也会导致性能急剧下降。

阿里云RDS的线程池实现了不同类型SQL操作的优先级及并发控制机制，将连接数始终控制在最佳连接数附近，使RDS数据库在高连接大并发情况下始终保持高性能。线程池的优势如下：

- 当大量线程并发工作时，线程池会自动调节并发的线程数量在合理的范围内，从而避免线程调度工作过多和大量缓存失效。
- 大量的事务并发执行时，线程池会将语句和事务分为不同的优先级，分别控制语句和事务的并发数量，从而减少资源竞争。
- 线程池给予管理类的SQL语句更高的优先级，保证这些语句优先执行。这样在系统负载很高时，新建连接、管理、监控等操作也能够稳定执行。
- 线程池给予复杂查询SQL语句相对较低的优先级，并且有最大并发数的限制。这样可以避免过多的复杂SQL语句将系统资源耗尽，导致整个数据库服务不可用。

前提条件

实例版本为RDS for MySQL 8.0。

使用Thread Pool

Thread Pool设计了如下三个参数，您可以在控制台进行修改。详情请参见#unique_39。

参数	说明
thread_pool_enabled	是否开启线程池功能。取值： · ON · OFF 默认值：OFF。
thread_pool_size	分组的数量，默认值：4。线程池中线程的被平均的分到多个组中进行管理。

参数	说明
thread_pool_oversubscribe	每个组中允许的活跃线程的数量，默认值：16。活跃线程是指正在执行SQL语句的线程，但是不包括以下两种情形： · SQL语句在等待磁盘IO； · SQL语句在等待事务提交。

查询Thread Pool状态

您可以通过如下命令查询Thread Pool状态：

```
show status like "thread_pool%";
```

示例：

```
mysql> show status like "thread_pool%";
+-----------------------------+-------+
| Variable_name               | Value |
+-----------------------------+-------+
| thread_pool_active_threads | 1     |
| thread_pool_big_threads    | 0     |
| thread_pool_dml_threads    | 0     |
| thread_pool_idle_threads   | 19    |
| thread_pool_qry_threads    | 0     |
| thread_pool_total_threads  | 20    |
| thread_pool_trx_threads    | 0     |
| thread_pool_wait_threads   | 0     |
+-----------------------------+-------+
8 rows in set (0.00 sec)
```

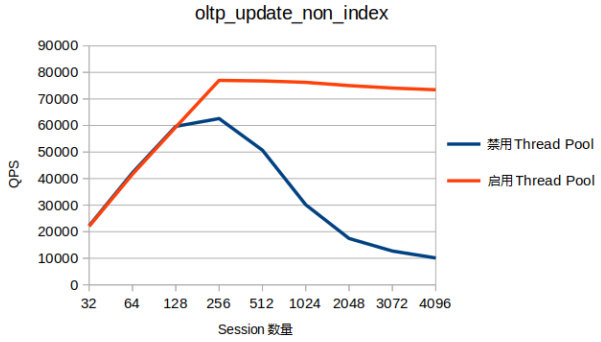
参数说明如下。

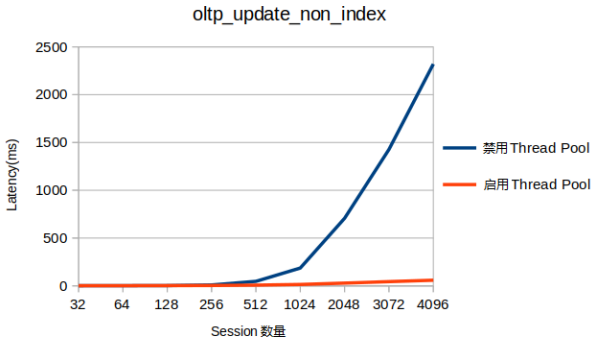
参数	说明
thread_pool_active_threads	线程池中的活跃线程数。
thread_pool_big_threads	线程池中正在执行复杂查询的线程数。复杂查询包括有子查询、聚合函数、group by、limit等的查询语句。
thread_pool_dml_threads	线程池中的在执行DML的线程数。
thread_pool_idle_threads	线程池中的空闲线程数。
thread_pool_qry_threads	线程池中正在执行简单查询的线程数。
thread_pool_total_threads	线程池中的总线程数。

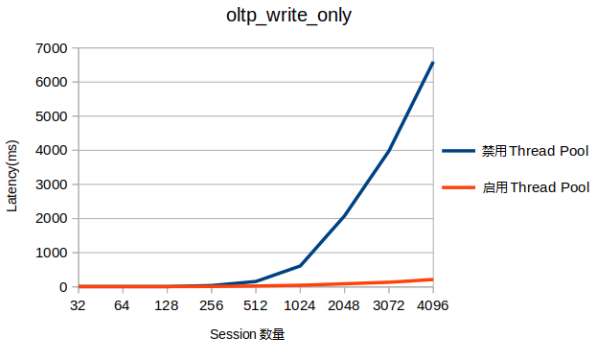
参数	说明
thread_pool_trx_threads	线程池中正在执行事务的线程数。
thread_pool_wait_threads	线程池中正在等待磁盘IO、事务提交的线程数。

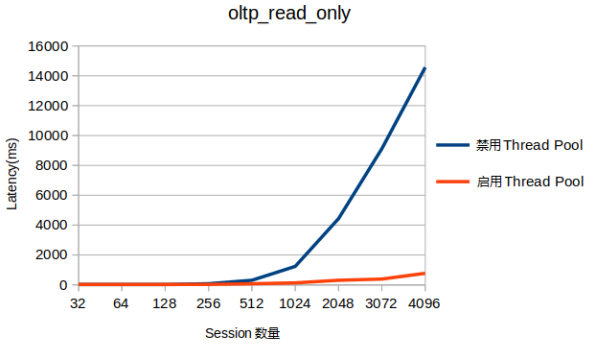
Sysbench测试

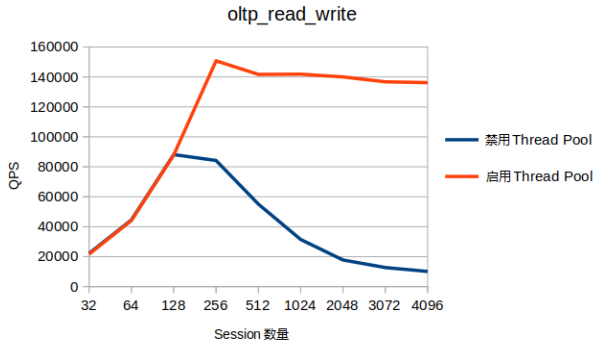
如下是开启线程池和不开启线程池的性能对比。从测试结果可以看出线程池在高并发的情况下有着明显的性能优势。

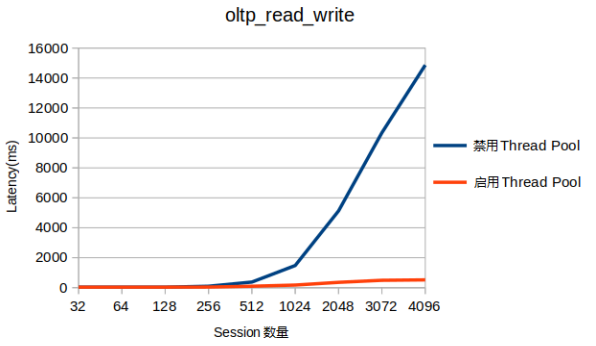


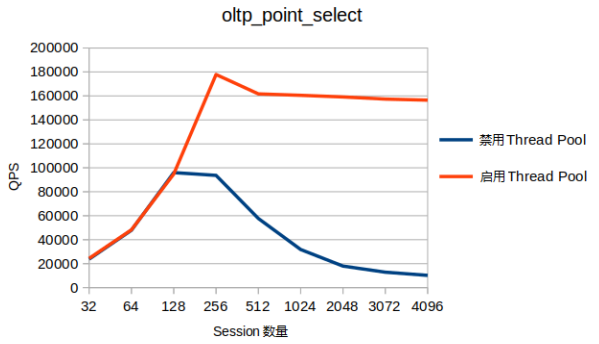


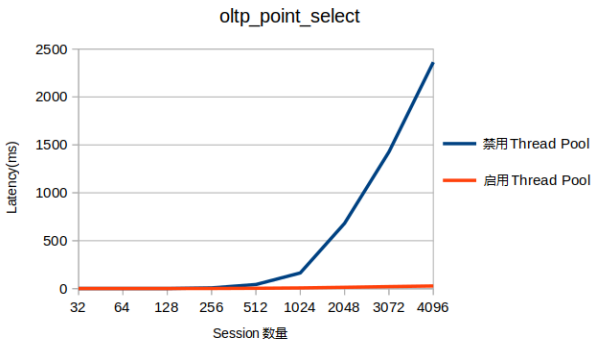





oltp_read_write
QPS
160000
140000
120000
100000
80000
60000
40000
20000
0
32 64 128 256 512 1024 2048 3072 4096
Session 数量
禁用Thread Pool
启用Thread Pool


oltp_read_write
Latency(ms)
16000
14000
12000
10000
8000
6000
4000
2000
0
32 64 128 256 512 1024 2048 3072 4096
Session 数量
禁用Thread Pool
启用Thread Pool


oltp_point_select
QPS
200000
180000
160000
140000
120000
100000
80000
60000
40000
20000
0
32 64 128 256 512 1024 2048 3072 4096
Session 数量
禁用Thread Pool
启用Thread Pool


oltp_point_select
Latency(ms)
2500
2000
1500
1000
500
0
32 64 128 256 512 1024 2048 3072 4096
Session 数量
禁用Thread Pool
启用Thread Pool

7 Sequence Engine

AliSQL提供了Sequence Engine，简化获取序列值的复杂度。

Sequence Engine介绍

在持久化数据库系统中，无论是单节点中的业务主键，还是分布式系统中的全局唯一值，亦或是多系统中的幂等控制，单调递增的唯一值是常见的需求。不同的数据库系统有不同的实现方法，例如MySQL提供的AUTO_INCREMENT，Oracle、SQL Server提供的SEQUENCE。

在MySQL数据库中，如果业务希望封装唯一值，例如增加日期、用户等信息，使用AUTO_INCREMENT的方法会带来很大不便，在实际的系统设计中，也存在不同的折中方法：

- 序列值由Application或者Proxy来生成，不过弊端很明显，状态带到应用端会增加扩容和缩容的复杂度。
- 序列值由数据库通过模拟的表来生成，但需要中间件来封装和简化获取唯一值的逻辑。

AliSQL提供了Sequence Engine，通过引擎的设计方法，尽可能地兼容其他数据库的使用方法，简化获取序列值复杂度。

Sequence Engine实现了MySQL存储引擎的设计接口，但底层的数据仍然使用现有的存储引擎，例如InnoDB或者MyISAM来保存持久化数据，兼容现有的第三方工具（例如Xtrabackup），所以Sequence Engine仅仅是一个逻辑引擎。

Sequence Engine通过Sequence Handler接口访问Sequence对象，实现NEXTVAL的滚动、缓存的管理等，最后透传给底层的基表数据引擎，实现最终的数据访问。

使用限制

- Sequence不支持子查询和join查询。
- 可以使用`SHOW CREATE TABLE`或者`SHOW CREATE SEQUENCE`来访问Sequence结构，但不能使用`SHOW CREATE SEQUENCE`访问普通表。
- 不支持建表的时候指定Sequence引擎，Sequence表只能通过创建Sequence的语法来创建。

创建Sequence

创建Sequence语句如下：

```
CREATE SEQUENCE [IF NOT EXISTS] schema.sequence_name
   [START WITH <constant>]
   [MINVALUE <constant>]
   [MAXVALUE <constant>]
   [INCREMENT BY <constant>]
   [CACHE <constant> | NOCACHE]
   [CYCLE | NOCYCLE]
```

```
  ;
```

参数说明如下。

参数	说明
START	Sequence的起始值。
MINVALUE	Sequence的最小值。
MAXVALUE	Sequence的最大值。  说明: 如果有参数NOCYCLE，到达最大值后会报如下错误： `ERROR HY000: Sequence 'db.seq' has been run out.`
INCREMENT BY	Sequence的步长。
CACHE/NOCACHE	缓存的大小，为了性能考虑，可以设置较大的缓存，但如果遇到实例重启，缓存内的值会丢失。
CYCLE/NOCYCLE	表示Sequence如果用完了后，是否允许从MINVALUE重新开始。取值： · CYCLE：允许； · NOCYCLE：不允许。

示例：

```
create sequence s
       start with 1
       minvalue 1
       maxvalue 9999999
       increment by 1
       cache 20
       cycle;
```

为了兼容MySQL Dump的备份方式，您也可以使用另外一种创建Sequence的方法，即创建Sequence表并插入一行初始记录。示例如下：

```
CREATE SEQUENCE schema.sequence_name (
  `currval` bigint(21) NOT NULL COMMENT 'current value',
  `nextval` bigint(21) NOT NULL COMMENT 'next value',
  `minvalue` bigint(21) NOT NULL COMMENT 'min value',
  `maxvalue` bigint(21) NOT NULL COMMENT 'max value',
  `start` bigint(21) NOT NULL COMMENT 'start value',
  `increment` bigint(21) NOT NULL COMMENT 'increment value',
  `cache` bigint(21) NOT NULL COMMENT 'cache size',
  `cycle` bigint(21) NOT NULL COMMENT 'cycle state',
  `round` bigint(21) NOT NULL COMMENT 'already how many round'
) ENGINE=InnoDB DEFAULT CHARSET=latin1

INSERT INTO schema.sequence_name VALUES(0,0,1,9223372036854775807,1,1,
10000,1,0);
```

```
COMMIT;
```

Sequence表介绍

由于Sequence是通过真正的引擎表来保存的，所以通过查询创建语句看到仍然是默认的引擎表。

示例如下：

```
SHOW CREATE [TABLE|SEQUENCE] schema.sequence_name;

CREATE SEQUENCE schema.sequence_name (
  `currval` bigint(21) NOT NULL COMMENT 'current value',
  `nextval` bigint(21) NOT NULL COMMENT 'next value',
  `minvalue` bigint(21) NOT NULL COMMENT 'min value',
  `maxvalue` bigint(21) NOT NULL COMMENT 'max value',
  `start` bigint(21) NOT NULL COMMENT 'start value',
  `increment` bigint(21) NOT NULL COMMENT 'increment value',
  `cache` bigint(21) NOT NULL COMMENT 'cache size',
  `cycle` bigint(21) NOT NULL COMMENT 'cycle state',
  `round` bigint(21) NOT NULL COMMENT 'already how many round'
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

查询语法

Sequence支持的查询语法如下：

```
 SELECT [nextval | currval | *] FROM seq;
 SELECT nextval(seq),currval(seq);
 SELECT seq.currval, seq.nextval from dual;
```

8 Performance Insight

Performance Insight是专注于实例负载监控、关联分析、性能调优的利器，帮助您迅速评估数据库负载，找到性能问题的源头，提升数据库的稳定性。

前提条件

实例版本为RDS for MySQL 5.7。

Performance Insight介绍

Performance Insight由如下两部分组成：

- Object statistics

  Object statistics查询表和索引的统计信息，包括如下两个表：

  - TABLE_STATISTICS：记录读取和修改的行。
  - INDEX_STATISTICS：记录索引的读取行。

- Performance point

  Performance point提供实例的详细性能信息，方便您更快更准确地量化SQL的开销。Performance point包括如下三个维度：

  - CPU：包括执行任务的总时间（Elapsed time）、CPU执行任务的时间（CPU time）等。
  - LOCK：包括服务器MDL锁时间、存储事务锁时间、互斥冲突（仅调试模式）、读写锁冲突等。
  - IO：数据文件读写时间、日志文件写入时间、逻辑读取、物理读取、物理异步读取等。

Object statistics使用方法

1. 确认参数OPT_TABLESTAT和OPT_INDEXSTAT的值为ON。示例如下：

```
mysql> show variables like "opt_%_stat";
  +---------------+-------+
  | Variable_name | Value |
  +---------------+-------+
  | opt_indexstat | ON    |
  | opt_tablestat | ON    |
  +---------------+-------+
```

说明:

如果参数找不到或参数值不为ON，请确认您的实例版本是否为MySQL 5.7。

2. **在information_schema数据库查询TABLE_STATISTICS表或INDEX_STATISTICS表，查看表和索引的统计信息。示例如下：**

```
mysql> select * from TABLE_STATISTICS limit 10;
  +--------------+-------------+----------+-------------
+----------------------+--------------+-------------
+--------------+
  | TABLE_SCHEMA | TABLE_NAME   | ROWS_READ | ROWS_CHANGED |
ROWS_CHANGED_X_INDEXES | ROWS_INSERTED | ROWS_DELETED | ROWS_UPDATED
 |
  +--------------+-------------+----------+-------------
+----------------------+--------------+-------------
+--------------+
  | mysql        | db           |         2 |            0 |
            0 |            0 |           0 |           0 |
  | mysql        | engine_cost  |         2 |            0 |
            0 |            0 |           0 |           0 |
  | mysql        | proxies_priv |         1 |            0 |
            0 |            0 |           0 |           0 |
  | mysql        | server_cost  |         6 |            0 |
            0 |            0 |           0 |           0 |
  | mysql        | tables_priv  |         2 |            0 |
            0 |            0 |           0 |           0 |
  | mysql        | user         |         7 |            0 |
            0 |            0 |           0 |           0 |
  | test         | sbtest1      |      1686 |          142 |
          184 |          112 |          12 |          18 |
  | test         | sbtest10     |      1806 |          125 |
          150 |          105 |           5 |          15 |
  | test         | sbtest100    |      1623 |          141 |
          182 |          110 |          10 |          21 |
  | test         | sbtest11     |      1254 |          136 |
          172 |          110 |          10 |          16 |
  +--------------+-------------+----------+-------------
+----------------------+--------------+-------------
+--------------+

  mysql> select * from INDEX_STATISTICS limit 10;
  +--------------+-------------+------------+-----------+
  | TABLE_SCHEMA | TABLE_NAME   | INDEX_NAME | ROWS_READ |
  +--------------+-------------+------------+-----------+
  | mysql        | db           | PRIMARY    |         2 |
  | mysql        | engine_cost  | PRIMARY    |         2 |
  | mysql        | proxies_priv | PRIMARY    |         1 |
  | mysql        | server_cost  | PRIMARY    |         6 |
  | mysql        | tables_priv  | PRIMARY    |         2 |
  | mysql        | user         | PRIMARY    |         7 |
  | test         | sbtest1      | PRIMARY    |      2500 |
  | test         | sbtest10     | PRIMARY    |      3007 |
  | test         | sbtest100    | PRIMARY    |      2642 |
  | test         | sbtest11     | PRIMARY    |      2091 |
  +--------------+-------------+------------+-----------+
```

参数说明如下。

参数	说明
TABLE_SCHEMA	**数据库名称。**
TABLE_NAME	**表名称。**

参数	说明
ROWS_READ	读的行数。
ROWS_CHANGED	修改的行数。
ROWS_CHANGED_X_INDEXES	索引修改的行数。
ROWS_INSERTED	插入的行数。
ROWS_DELETED	删除的行数。
ROWS_UPDATED	更新的行数。
INDEX_NAME	索引名称。

Performance point使用方法

1. 确认Performance point的相关参数。正常的示例如下：

```
mysql> show variables like "%performance_point%";
  +---------------------------------------+-------+
  | Variable_name                         | Value |
  +---------------------------------------+-------+
  | performance_point_dbug_enabled        | OFF   |
  | performance_point_enabled             | ON    |
  | performance_point_iostat_interval     | 2     |
  | performance_point_iostat_volume_size  | 10000 |
  | performance_point_lock_rwlock_enabled | ON    |
  +---------------------------------------+-------+
```

> **说明:**
> 如果参数找不到，请确认您的实例版本是否为MySQL 5.7。

2. 在performance_schema数据库查询events_statements_summary_by_digest_supplement表，查看排名前10的SQL语句。示例如下：

```
mysql> select * from events_statements_summary_by_digest_supplement
limit 10;
  +--------------------+---------------------------------
+-----------------------------------------+-------------+
  | SCHEMA_NAME        | DIGEST                          |
DIGEST_TEXT                             | ELAPSED_TIME | ......
  +--------------------+---------------------------------
+-----------------------------------------+-------------+
  | NULL               | 6b787dd1f9c6f6c5033120760a1a82de | SELECT
 @@`version_comment` LIMIT ?        |          932 |
  | NULL               | 2fb4341654df6995113d998c52e5abc9 | SHOW
SCHEMAS                                 |         2363 |
  | NULL               | 8a93e76a7846384621567fb4daa1bf95 | SHOW
VARIABLES LIKE ?                        |        17933 |
  | NULL               | dd148234ac7a20cb5aee7720fb44b7ea | SELECT
SCHEMA ( )                              |         1006 |
```

```
  | information_schema | 2fb4341654df6995113d998c52e5abc9 | SHOW
SCHEMAS                                |          2156 |
  | information_schema | 74af182f3a2bd265678d3dadb53e08da | SHOW
TABLES                                 |          3161 |
  | information_schema | d3a66515192fcb100aaef6f8b6e45603 | SELECT
 * FROM `TABLE_STATISTICS` LIMIT ?  |          2081 |
  | information_schema | b3726b7c4c4db4b309de2dbc45ff52af | SELECT
 * FROM `INDEX_STATISTICS` LIMIT ?  |          2384 |
  | information_schema | dd148234ac7a20cb5aee7720fb44b7ea | SELECT
SCHEMA ( )                             |           129 |
  | test               | 2fb4341654df6995113d998c52e5abc9 | SHOW
SCHEMAS                                |           342 |
  +--------------------+----------------------------------
+-----------------------------------------+--------------+
```

参数说明如下。

参数	说明
SCHEMA_NAME	数据库名称。
DIGEST	Digest_text进行hash计算得到的64字节的hash字符串。
DIGEST_TEXT	SQL语句的特征。
ELAPSED_TIME	实际运行时间。单位：μs。
CPU_TIME	CPU运行时间。单位：μs。
SERVER_LOCK_TIME	服务器锁定时间。单位：μs。
TRANSACTIO N_LOCK_TIME	存储事务锁定时间。单位：μs。
MUTEX_SPINS	互斥旋转次数。
MUTEX_WAITS	互斥等待次数。
RWLOCK_SPI N_WAITS	读写闩锁的自旋等待数。
RWLOCK_SPI N_ROUNDS	读写闩锁的旋转循环圈数。
RWLOCK_OS_WAITS	读写闩锁的操作系统等待数。
DATA_READS	数据文件读取次数。
DATA_READ_TIME	数据文件读取时间。单位：μs。
DATA_WRITES	数据文件写入次数。
DATA_WRITE_TIME	数据文件写入时间。单位：μs。
REDO_WRITES	日志文件写入次数。
REDO_WRITE_TIME	日志文件写入时间。单位：μs。

参数	说明
LOGICAL_READS	逻辑页读取次数。
PHYSICAL_READS	物理页读取次数。
PHYSICAL_ASYNC_READS	物理异步页读取次数。

3. 在information_schema数据库查询IO_STATISTICS表，查看最近的数据读写情况。示例如下：

```
mysql> select * from IO_STATISTICS limit 10;
  +---------------------+-----------+----------------+
  | TIME                | DATA_READ | DATA_READ_TIME | ......
  +---------------------+-----------+----------------+
  | 2019-08-08 09:56:53 |        73 |            983 |
  | 2019-08-08 09:56:57 |         0 |              0 |
  | 2019-08-08 09:59:17 |         0 |              0 |
  | 2019-08-08 10:00:55 |      4072 |          40628 |
  | 2019-08-08 10:00:59 |         0 |              0 |
  | 2019-08-08 10:01:09 |       562 |           5800 |
  | 2019-08-08 10:01:11 |       606 |           6910 |
  | 2019-08-08 10:01:13 |       609 |           6875 |
  | 2019-08-08 10:01:15 |       625 |           7077 |
  | 2019-08-08 10:01:17 |       616 |           5800 |
  +---------------------+-----------+----------------+
```

参数说明如下。

参数	说明
TIME	日期。
DATA_READ	数据读取次数。
DATA_READ_TIME	数据读取总时间。单位：μs。
DATA_READ_MAX_TIME	数据读取最长时间。单位：μs。
DATA_READ_BYTES	数据读取总大小。单位：byte。
DATA_WRITE	数据写入次数。
DATA_WRITE_TIME	数据写入总时间。单位：μs。
DATA_WRITE_MAX_TIME	数据写入最长时间。单位：μs。
DATA_WRITE_BYTES	数据写入总大小。单位：byte。

9 Purge Large File Asynchronously

AliSQL支持通过异步删除大文件的方式保证系统稳定性。

背景信息

使用InnoDB引擎时，直接删除大文件会导致POSIX文件系统出现严重的稳定性问题，因此InnoDB会启动一个后台线程来异步清理数据文件。当删除单个表空间时，会将对应的数据文件先重命名为临时文件，然后清除线程将异步、缓慢地清理文件。

说明:

AliSQL提供清除文件日志来保证DDL语句的原子性。

使用方法

1. 查看实例全局变量设置，示例如下：

```
mysql> SHOW GLOBAL VARIABLES LIKE '%data_file_purge%';
  +-----------------------------------------+-------+
  | Variable_name                           | Value |
  +-----------------------------------------+-------+
  | innodb_data_file_purge                  | ON    |
  | innodb_data_file_purge_all_at_shutdown  | OFF   |
  | innodb_data_file_purge_dir              |       |
  | innodb_data_file_purge_immediate        | OFF   |
  | innodb_data_file_purge_interval         | 100   |
  | innodb_data_file_purge_max_size         | 128   |
  | innodb_print_data_file_purge_process    | OFF   |
  +-----------------------------------------+-------+
```

参数说明如下。

参数	说明
innodb_data_file_purge	是否启用异步清除策略。
innodb_data_file_purge_all_at_shutdown	正常关机时全部清理。
innodb_data_file_purge_dir	临时文件目录。
innodb_data_file_purge_immediate	取消数据文件的链接但不清理。

参数	说明
innodb_data_file_purge_interval	清理时间间隔。单位：ms。
innodb_data_file_purge_max_size	每次清理单个文件大小的最大值。单位：MB。
innodb_print_data_file_purge_process	是否打印文件清理工作进程。

说明:

建议使用如下命令进行设置：

```
set global INNODB_DATA_FILE_PURGE = on;
set global INNODB_DATA_FILE_PURGE_INTERVAL = 100;
set global INNODB_DATA_FILE_PURGE_MAX_SIZE = 128;
```

2. **使用如下命令查看清理进度：**

```
select * from information_schema.innodb_purge_files;
```