|  |  |
| --- | --- |
| 客户名称 | 汇付天下有限公司（MySQL） |
| 问题描述 | 升级KB之后发现附件无法上传，卡在100%两分钟后异常中断 |
| 处理过程 | 升级最新的KB之后发现此问题，而且每次的逻辑都是必现的。  每次上传附件时，插入的逻辑都会被锁住，锁超时之后抛出如下异常：  Lock wait timeout exceeded; try restarting transaction  跟踪数据库的SQL日志发现如下的事务逻辑：    其中：  ①的部分是会话3删除sourceid=33280的旧数据  ②的事务是会话12插入sourceid=32881新数据  截图部分的逻辑原来是在存储过程中实现，最新的KB版本中已去除存储过程，改用代码的逻辑进行操作数据，先删除再进行插入，但是这两个操作代码中并没有全部放到一个数据库会话单元中进行，此类操作在事务隔离级别是read-committed（不可重复读）时没有问题，但是在repeatable-read（可重复读）的模式下，就引出了间隙锁的异常，在此隔离级别下为了解决幻读的问题，引入间隙锁，在修改数据时会对一定范围的数据增加锁，而不只是增加被修改数据的行级锁。  会话1  delete from test where id = 1; --nocommt  会话2  insert into test values(2,'tom'); --locked  使用如下命令进行事务隔离级别修改：  SET GLOBAL TRANSACTION ISOLATION LEVEL READ-COMMITTED;  重启一次Ecology程序服务，问题即可解决。  总结：  此参数在我们程序安装适配过程已明确规定要求使用READ-COMMITTED，而部分客户在安装或者后面数据迁移时没有注重此类参数的修改，导致上面的问题；在程序初始化和后期的迁移过程中，一定要按照我们的手册进行。 |
| 客户名称 | 广州飞特物流有限公司（MySQL） |
| 问题描述 | 使用最新的安装包，在MySQL初始化时异常 |
| 处理过程 | 项目反馈最新的安装包初始化异常，抛出的报错如下：    尝试使用数据库命令直接执行此操作，抛出一样的异常；    后经MySQL官方手册比对发现，系MySQL8.0低版本的BUG导致的问题，此问题出现在8.0.12-8.0.13两个版本中，以上两个版本的数据库，在创建触发器时不区分大小写，而删除时却是区分大小写的，导致删除时有此触发器，但删除过程找不到，抛出触发器不存在的异常；更换数据库至最新版即可解决此问题。  总结：  最新的手册中已更换了数据库的下载连接，在安装部署程序的过程一定要使用配套的手册进行安装，此问题较低的Ecology程序中没有异常，最新版因为加入了新的脚本，请使用配套的手册进行部署。 |
| 客户名称 | 河南科隆集团有限公司（MySQL） |
| 问题描述 | 数据库安装好之后本地可以连接，远程连接卡死 |
| 处理过程 | 原来的SQL删除逻辑如下：  DELETE FROM historymsg WHERE id ON IN (SELECT MAX(id) id FROM historymsg GROUP BY msgid);  DELETE FROM Social\_historyMsgRight WHERE id ON IN (SELECT MAX(id) id FROM Social\_historyMsgRight GROUP BY userid,msgid)  改造后的SQL删除逻辑：  CREATE TABLE historymsg\_bak LIKE historymsg;  CREATE TABLE tmp\_historymsg\_id AS SELECT MAX(id) id FROM historymsg GROUP BY msgid;  CREATE INDEX idx\_id ON tmp\_historymsg\_id(id);  INSERT INTO historymsg\_bak SELECT a.\* FROM historymsg a INNER JOIN tmp\_historymsg\_id b ON a.id = b.id;  RENAME TABLE historymsg TO historymsg\_210122;  RENAME TABLE historymsg\_bak TO historymsg;  DROP TABLE tmp\_historymsg\_id;  CREATE TABLE Social\_historyMsgRight\_bak LIKE Social\_historyMsgRight;  CREATE TABLE tmp\_Social\_historyMsgRight\_id AS SELECT MAX(id) id FROM Social\_historyMsgRight GROUP BY userid,msgid;  CREATE INDEX idx\_id ON tmp\_Social\_historyMsgRight\_id(id);  INSERT INTO Social\_historyMsgRight\_bak SELECT a.\* FROM Social\_historyMsgRight a INNER JOIN tmp\_Social\_historyMsgRight\_id b ON a.id = b.id;  RENAME TABLE Social\_historyMsgRight TO Social\_historyMsgRight\_210122;  RENAME TABLE Social\_historyMsgRight\_bak TO Social\_historyMsgRight;  DROP TABLE tmp\_Social\_historyMsgRight\_id;  总结：  因为客户原标数据将近3000万，大表进行group by外层再加一个not in操作，数据库的buffer cache严重不足，会请求数据库的交换分区甚至磁盘进行额外操作，此类操作速度极低而且会临时占用较大的磁盘空间，有把磁盘撑满的风险；改造之后的逻辑是先把要保留的数据id拿到存放到一个中间表中，然后单独把要保留的数据存入其他表，将原标与处理后的表名进行互换即可，后面确认无异常再删除备份的表。 |
| 客户名称 | 上海东福网络科技有限公司（MySQL） |
| 问题描述 | 服务器CPU告警，数据库中存在并发高的的慢SQL |
| 处理过程 | 该SQL的查询逻辑耗时主要在排序分页上，SQL精简之后的逻辑如下：    进行优化改造后的SQL如下：    优化的思路如下：  类似这种分页+排序的SQL，第一种书写的逻辑，在使用createdate和createtime进行排序时，需要通过主键回表查询带出其他附带的字段信息，虽然可以利用到索引，但是这种逻辑并非最高效的，尤其再分页越靠后的时候，随着偏移量加大，需要拿到内存中的数据就更多，查询耗时就更久。而第二种SQL的书写方法，requestid和createdate，createtime字段上均有索引，在进行排序和分页时，只需要检索索引即可完成（MySQL的覆盖索引概念），只获取到分页之后的requestid值再与外部表进行inner join，查询速度会极大的提升，并且查询效率不会因为分页靠后而明显下降。  总结：  数据库的分页查询原始的写法并不是最优的，因为步长越大，数据库需要处理的数据量就越大，开销自然就不在一个量级，遇到一些数据量大，并发高的分页查询，请采用上面的优化后的写法进行改造。 |
| 客户名称 | 浙江省交通投资集团有限公司（MySQL） |
| 问题描述 | 客户反馈系统偶尔的卡顿，登入异常，打开流程慢 |
| 处理过程 | 通过慢日志分析对数据库的整体性能分析并进行了优化。  此处列举我们程序中常用的一个问题逻辑：  部分SQL片段如下    替换方法：    代码中较多的SQL发现开发人员习惯使用exists逻辑来过滤数据，但是在MySQL中，exists的性能并不是最高的，即使在字段存在索引的情况下，在结结果集比较大情况下，exists的检索速度远不如inner join的hash连接，而且过多的使用exists容易导致SQL的执行计划异常，而inner join逻辑相对更加直接，简化。  总结：  数据库的表关联和判断有很多方法，我们研发的思路更偏向用exists来代替join，而数据库中exists的效率是远不及join的，在mysql数据库中，exists的写法始终会选择一个小表进行全表查询，当最小的表数据量也比较大时，性能问题就很大了；我们推荐的优先逻辑：join > exists > in。 |
| 客户名称 | 浙江省交通投资集团有限公司（MySQL） |
| 问题描述 | 客户反馈系统偶尔的卡顿，登入异常，打开流程慢 |
| 处理过程 | 通过慢日志分析对数据库的整体性能分析并进行了优化。  此处列举我们程序中常用的一个问题逻辑：  部分SQL片段如下    替换方法：    代码中较多的SQL发现开发人员习惯使用exists逻辑来过滤数据，但是在MySQL中，exists的性能并不是最高的，即使在字段存在索引的情况下，在结结果集比较大情况下，exists的检索速度远不如inner join的hash连接，而且过多的使用exists容易导致SQL的执行计划异常，而inner join逻辑相对更加直接，简化。  总结：  数据库的表关联和判断有很多方法，我们研发的思路更偏向用exists来代替join，而数据库中exists的效率是远不及join的，在mysql数据库中，exists的写法始终会选择一个小表进行全表查询，当最小的表数据量也比较大时，性能问题就很大了；我们推荐的优先逻辑：join > exists > in。 |
| 客户名称 | 北京悦刻科技有限公司（MySQL） |
| 问题描述 | 客户反馈数据库的CPU过高，监控到QPS长期在2000+ |
| 处理过程 | 根据客户反馈，到数据库服务器执行show processlist命令观察SQL情况      客户的数据库云平台监控QPS结果如下：    后来与知识组同事确认，上面截图的SQL是OSS的定时任务，每小时执行一次，但明显此处执行的频率大很多，后将此表先进行重命名操作：  rename table imagefile to imagefile\_bak;  rename table imagefile\_bak to imagefile;  让程序的定时任务报错，跳出了循环，再次观察数据库的QPS已降低至100以内：    总结：  无论是程序还是数据库，在进行循环操作时，如for和while等，切记一定要设置跳出循环的条件，综合考虑可能遇到的各种奇葩情况，比如参数传入null或者空字符串怎么处理，传入非法数值怎么判断异常等，否则一旦形成死循环就是致命的，只有人为的干预结束才可以解决问题。 |
| 客户名称 | 广州若羽臣科技股份有限公司（MySQL） |
| 问题描述 | 门户的日程加载慢，请帮忙看看是不是性能问题 |
| 处理过程 | 1.开发定位加载慢，主要卡在SQL上；  2.通过操作前台页面，数据库中show processlist;跟踪当前慢SQL，定位到是如下SQL比较慢；    3.进一步查看执行计划，慢主要在not exists涉及的子查询，耗时在全表扫描上，故添加优化索引解决，如下：  CREATE INDEX idx\_wpf\_workplanid\_hu ON workplanFinish(workplanid,hasFinish,userid); |
| 客户名称 | 吉利科技集团有限公司（MySQL） |
| 问题描述 | 人员主表和部门表单独还原到测试系统的方法 |
| 处理过程 | 1.以下是导出表数据结构及还原的命令语法：  1.1.导出表的数据及结构  mysqldump database -u username -p password -h ip地址 --tables table\_name1 table\_name2 table\_name3 > 文件路径.sql  1.2.还原操作：  mysql -u username -p password -h ip地址 所要还原的目标库名 < hrm\_tb\_data.sql  2.有了上面的语法，根据客户mysql实际环境进行更改，例如：  2.1.正式数据库服务器上表结构数据备份（自动包含有表上的触发器）  /usr/local/mysql/bin/mysqldump ecology -uecology -h127.0.0.1 -p --tables hrmresource hrmdepartment hrmsubcompany > hrm\_tb\_data.sql  2.2.测试数据库服务器上针对上面的备份文件进行还原操作  /usr/local/mysql/bin/mysql -uecology -h127.0.0.1 -p ecology < hrm\_tb\_data.sql  3.综合以上，可以作为MySQL中单独同步几张正式环境中的表到测试环境中的参考方法。 |
| 客户名称 | 三菱重工叉车（大连）有限公司（SQLServer） |
| 问题描述 | 协助清理ecology\_biz\_log表2020-04-29之前的人力资源模块数据 |
| 处理过程 | 1.在SSMS数据库客户端根据原有的ecology\_biz\_log表，提取表及索引的创建DDL（注意检查下该表上是否有触发器，有触发器的话也要提取下，并在SQL插入前禁用，插入后启用，本例是表上没有触发器的情况），替换为ecology\_biz\_log\_tmp0226表，先执行表的创建；  2.插入要保留的全量数据  set identity\_insert ecology\_biz\_log\_tmp0226 on  INSERT INTO ecology\_biz\_log\_tmp0226 (  id,  operateDate,  operateTime,  userid,  usertype,  targetId,  targetName,  logType,  logSmallType,  operateType,  clientIp,  oldValues,  newValues,  description,  params,  logTypeLabel,  logSmallTypeLabel,  belongtype,  belongTypeLabel,  belongTypeTargetId,  belongTypeTargetName,  isDetail,  mainId,  belongMainId,  groupId,  groupNameLabel,  operateAuditType,  isArchived,  deviceType  )  SELECT  id,  operateDate,  operateTime,  userid,  usertype,  targetId,  targetName,  logType,  logSmallType,  operateType,  clientIp,  oldValues,  newValues,  description,  params,  logTypeLabel,  logSmallTypeLabel,  belongtype,  belongTypeLabel,  belongTypeTargetId,  belongTypeTargetName,  isDetail,  mainId,  belongMainId,  groupId,  groupNameLabel,  operateAuditType,  isArchived,  deviceType  FROM  ecology\_biz\_log  WHERE NOT (  logtype BETWEEN 3  AND 4  AND operateDate <= '2020-04-29'  )set identity\_insert ecology\_biz\_log\_tmp0226 off  3.执行刚刚提取的创建索引SQL  CREATE NONCLUSTERED INDEX [i\_20191121\_03] ON [dbo].[ecology\_biz\_log\_tmp0226]([logType] ASC,[operateDate] ASC,[logSmallType] ASC)  4.插入增量数据的同时，对表进行重命名，也就是一起操作执行  4.1.获取ecology\_biz\_log\_tmp0226表上的最大id值  SELECT MAX(id) FROM ecology\_biz\_log\_tmp0226  4.2.获取的最大id值放于下方高亮部分，这里是732908，实现增量插入  set identity\_insert ecology\_biz\_log\_tmp0226 on  INSERT INTO ecology\_biz\_log\_tmp0226 (  id,  operateDate,  operateTime,  userid,  usertype,  targetId,  targetName,  logType,  logSmallType,  operateType,  clientIp,  oldValues,  newValues,  description,  params,  logTypeLabel,  logSmallTypeLabel,  belongtype,  belongTypeLabel,  belongTypeTargetId,  belongTypeTargetName,  isDetail,  mainId,  belongMainId,  groupId,  groupNameLabel,  operateAuditType,  isArchived,  deviceType  )  SELECT  id,  operateDate,  operateTime,  userid,  usertype,  targetId,  targetName,  logType,  logSmallType,  operateType,  clientIp,  oldValues,  newValues,  description,  params,  logTypeLabel,  logSmallTypeLabel,  belongtype,  belongTypeLabel,  belongTypeTargetId,  belongTypeTargetName,  isDetail,  mainId,  belongMainId,  groupId,  groupNameLabel,  operateAuditType,  isArchived,  deviceType  FROM  ecology\_biz\_log  WHERE NOT (  logtype BETWEEN 3  AND 4  AND operateDate <= '2020-04-29'  )  AND id > 732908 set identity\_insert ecology\_biz\_log\_tmp0226 OFF  EXEC sys.sp\_rename @objname = N'ecology\_biz\_log', @newname = 'ecology\_biz\_log\_bak\_20210226',@objtype = 'object'  EXEC sys.sp\_rename @objname = N'ecology\_biz\_log\_tmp0226', @newname = 'ecology\_biz\_log',@objtype = 'object'  5.以上可作为SQLServer中清理超大表数据的参考方法。 |
| 客户名称 | 浙江国际招（投）标公司（MySQL） |
| 问题描述 | 建模查询列表客户觉得卡顿 |
| 处理过程 | 1.开发定位主要是三个接口涉及的SQL执行比较慢，如下截图：    2.通过分析SQL的执行计划，发现这三个sql都主要慢在uf\_xiangmutaizhang这张自定义建模表上，故添加优化索引如下，优化后基本都在0.6s以内：  CREATE INDEX idx\_uf\_xiangmutaizhang\_fyc ON uf\_xiangmutaizhang(formmodeid,ywdl,cjr);  DROP INDEX idx\_sfzxm ON uf\_xiangmutaizhang;  CREATE INDEX idx\_sfzxm\_zxm ON uf\_xiangmutaizhang(sfzxm,zxm);  3.所以日常在通过建模自定义表后，一定要结合业务，针对和自定义表相关的条件字段、表关联字段加上合适的索引。 |
| 客户名称 | 广东佛盈智慧大数据科技有限公司（SQLServer） |
| 问题描述 | 数据库还原报错：因为数据库正在使用，所以无法获得对数据库的独占访问权 |
| 处理过程 | 1.客户反馈，数据库还原时，报如下截图的错：    2.sqlserver数据库还原前是要获取该库的独占访问权的，由于一直还有其它进程连接该库，所以无法第一时间获取独占，有的时候尽管我们停了OA服务，但依然无法第一时间获取到要还原目标库的独占权就是这个原因；  3.故可以通过创建一个存储过程，在还原前kill掉所有和这个库有关的连接；  4.通常我们选择在master系统库下创建存储过程，如下SQL：  USE [master]  GO  CREATE PROCEDURE [dbo].[pr\_KillSpid] @dbname sysname  AS  DECLARE @s NVARCHAR(1000)  DECLARE spid\_cursor CURSOR LOCAL  FOR  SELECT N'kill '+CAST(spid AS VARCHAR)  FROM master.dbo.sysprocesses  WHERE dbid=DB\_ID(@dbname) AND spid > 50  OPEN spid\_cursor  FETCH NEXT FROM spid\_cursor INTO @s  WHILE @@fetch\_status=0  BEGIN  EXEC(@s)  FETCH NEXT FROM spid\_cursor INTO @s  END  CLOSE spid\_cursor  DEALLOCATE spid\_cursor  GO  5.正式还原参考操作如下： |