**GXB KUNDB 测试案例&报告**

**目 录**

[目 录](#_Toc1167032594)

[1. 整体概述](#_Toc1171873311)

[2. 测试结论](#_Toc70365994)

[2.1 测试资源配置](#_Toc1110828889)

[2.1.1 硬件配置](#_Toc97295664)

[2.1.2 软件配置](#_Toc1667928860)

[2.2 测试网络拓补](#_Toc291279797)

[2.2.1 角色分布拓补](#_Toc1631399920)

[2.2.2 Shard分布拓补](#_Toc686428570)

[2.3 测试结论](#_Toc636038885)

[3. 功能性](#_Toc1049746110)

[3.1 安装运行](#_Toc645942545)

[3.1.1 验证KunDB部署功能](#_Toc494636803)

[3.2 重新启动](#_Toc1976949192)

[3.2.1 验证KunDB服务重新启动功能](#_Toc1156095626)

[3.3 系统配置管理](#_Toc869434074)

[3.3.1 验证KunDB系统配置功能](#_Toc550076276)

[3.4 模式管理](#_Toc508845374)

[3.4.1 验证KunDB创建模式功能](#_Toc1186397784)

[3.4.2 验证KunDB查看模式功能](#_Toc1010385105)

[3.4.3 验证KunDB修改模式功能](#_Toc232296532)

[3.4.4 验证KunDB删除模式功能](#_Toc1076098490)

[3.5 表对象管理](#_Toc1447808154)

[3.5.1 验证KunDB创建表对象功能](#_Toc2012242748)

[3.5.2 验证KunDB删除表对象功能](#_Toc790811061)

[3.5.3 验证KunDB修改表名称功能](#_Toc2127750316)

[3.5.4 验证KunDB修改表列名称功能](#_Toc892072313)

[3.5.5 验证KunDB修改表列类型功能](#_Toc1279968453)

[3.5.6 验证KunDB添加表字段功能](#_Toc925110938)

[3.5.7 验证KunDB删除表字段功能](#_Toc2067999946)

[3.6 视图对象管理](#_Toc1124932704)

[3.6.1 验证KunDB创建视图对象功能](#_Toc2092143532)

[3.6.2 验证KunDB删除视图对象功能](#_Toc1092389609)

[3.6.3 验证KunDB修改视图对象功能](#_Toc1195298698)

[3.6.4 验证KunDB修改视图名称功能](#_Toc1055488773)

[3.7 数据对象管理](#_Toc1189685274)

[3.7.1 验证KunDB插入数据功能](#_Toc715743910)

[3.7.2 验证KunDB更新数据功能](#_Toc1346768571)

[3.7.3 验证KunDB删除数据功能](#_Toc673601546)

[3.8 用户权限管理](#_Toc1402172480)

[3.8.1 验证KunDB支持创建用户功能](#_Toc1982807456)

[3.8.2 验证KunDB支持删除用户功能](#_Toc1723347657)

[3.8.3 验证KunDB支持授权用户数据库对象级权限功能](#_Toc2048115025)

[3.8.4 验证KunDB支持授权用户表对象级权限功能](#_Toc329960611)

[3.8.5 验证KunDB支持授权用户视图对象级权限功能](#_Toc1552813201)

[3.9 数据类型支持](#_Toc1056727003)

[3.9.1 验证KunDB支持基础数据类型功能](#_Toc1199394686)

[3.10 大对象支持](#_Toc2102889477)

[3.11 约束](#_Toc1565572377)

[3.11.1 验证KunDB支持主键约束功能](#_Toc238308822)

[3.11.2 验证KunDB支持删除主键约束功能](#_Toc965790934)

[3.11.3 验证KunDB支持多健值主键约束功能](#_Toc1797868909)

[3.11.4 验证KunDB支持唯一约束功能](#_Toc1314407313)

[3.11.5 验证KunDB支持删除唯一约束功能](#_Toc266115440)

[3.11.6 验证KunDB支持添加唯一约束功能](#_Toc1662628009)

[3.11.7 验证KunDB支持非空约束功能](#_Toc2105218374)

[3.11.8 验证KunDB支持删除非空约束功能](#_Toc246382108)

[3.11.9 验证KunDB支持添加非空约束功能](#_Toc407216674)

[3.11.10 验证KunDB支持外健约束功能](#_Toc1237703180)

[3.12 存储对象使用管理](#_Toc1171493046)

[3.12.1 验证KunDB支持存储过程使用动态SQL功能](#_Toc327732973)

[3.12.2 验证KunDB支持修改存储过程功能](#_Toc215152236)

[3.12.3 验证KunDB支持修改存储过程名称功能](#_Toc1116152930)

[3.12.4 验证KunDB支持删除存储过程功能](#_Toc1420122582)

[3.12.5 验证KunDB支持存储过程使用游标功能](#_Toc1410450934)

[3.13 触发器使用管理](#_Toc24158055)

[3.13.1 验证KunDB支持触发器使用动态SQL和游标功能](#_Toc462324208)

[3.13.2 验证KunDB支持修改触发器名称功能](#_Toc2126194844)

[3.13.3 验证KunDB支持删除触发器功能](#_Toc1370926626)

[3.14 分片操作](#_Toc1135925755)

[3.14.1 验证KunDB支持分片操作功能](#_Toc1380883676)

[3.15 函数管理](#_Toc1206250435)

[3.15.1 验证KunDB支持计算函数功能](#_Toc711789764)

[3.15.2 验证KunDB支持日期函数功能](#_Toc1281515053)

[3.15.3 验证KunDB支持字符函数功能](#_Toc1536211046)

[3.16 字符集编码](#_Toc117119317)

[3.16.1 验证KunDB支持GBK字符集功能](#_Toc190758408)

[3.16.2 验证KunDB支持GB18030字符集功能](#_Toc588122084)

[3.17 分区表管理](#_Toc72525146)

[3.17.1 验证KunDB支持HASH分区表功能](#_Toc1756330785)

[3.17.2 验证KunDB支持RANGE分区表功能](#_Toc826430907)

[3.17.3 验证KunDB支持LIST分区表功能](#_Toc1038316080)

[3.18 分布式事务管理](#_Toc1406716046)

[3.18.1 验证KunDB支持分布式事务全部回滚原子性功能](#_Toc2140838220)

[3.18.2 验证KunDB支持分布式事务全部提交原子性功能](#_Toc1304431520)

[3.18.3 验证KunDB支持分布式事务一致性功能](#_Toc921860407)

[3.18.4 验证KunDB支持分布式事务隔离性功能](#_Toc2098572946)

[3.18.5 验证KunDB支持分布式事务持久性功能](#_Toc1550813628)

[3.18.6 验证KunDB支持分布式事务原子性功能](#_Toc1329077082)

[3.18.7 验证KunDB支持分布式事务独立性功能](#_Toc1188792478)

[3.19 索引](#_Toc574823026)

[3.19.1 验证KunDB支持主键索引功能](#_Toc1656810055)

[3.19.2 验证KunDB支持删除主键索引功能](#_Toc1403944714)

[3.19.3 验证KunDB支持普通索引功能](#_Toc1690975956)

[3.19.4 验证KunDB支持多健值索引功能](#_Toc929448989)

[3.19.5 验证KunDB支持删除普通索引功能](#_Toc666912000)

[3.19.6 验证KunDB支持唯一索引功能](#_Toc1715134011)

[3.19.7 验证KunDB支持删除唯一索引功能](#_Toc1391773198)

[3.19.8 验证KunDB支持多健值索引可提升查询功能](#_Toc645623196)

[3.19.9 验证KunDB支持图形化管理索引功能](#_Toc938576990)

[3.20 数据均匀分布](#_Toc380215305)

[3.20.1 验证KunDB支持表的数据均匀分布到每个存储节点能力](#_Toc2026506872)

[3.21 图形化工具](#_Toc2144827425)

[3.22 配套工具](#_Toc1092005069)

[3.23 MVCC](#_Toc1160538277)

[3.23.1 验证KunDB支持MVCC功能](#_Toc1533554823)

[3.24 在线升级](#_Toc1209124386)

[3.24.1 验证KunDB支持在线升级功能](#_Toc1351296685)

[4. 性能效率](#_Toc2121676908)

[4.1 分布式读写](#_Toc1281649532)

[4.1.1 验证TPC-C 1000仓数据的装载时间和正确性](#_Toc960143822)

[4.2 TPC-C模拟测试](#_Toc800624167)

[4.2.1 验证1000并发用户的TPC-C模拟测试的tpmC和数据一致性](#_Toc172481964)

[4.2.2 验证3000并发用户的TPC-C模拟测试的tpmC和数据一致性](#_Toc219376220)

[4.3 海量数据复杂查询、Join查询性能](#_Toc793978739)

[4.3.1 验证KunDB系统海量数据复杂查询、Join查询的性能](#_Toc1476913484)

[5. 可靠性](#_Toc1141236628)

[5.1 168小时稳定性测试](#_Toc745068037)

[5.1.1 验证500并发用户、500仓数据的168小时TPC-C模拟测试](#_Toc880243464)

[5.2 模拟节点断电测试](#_Toc322830062)

[5.2.1 验证KunDB系统节点断电情况下的运行情况](#_Toc1933860516)

[5.3 模拟节点断网测试](#_Toc1455066490)

[5.3.1 验证KunDB系统节点断网情况下的运行情况](#_Toc1979640117)

[5.4 全断网测试](#_Toc1190321582)

[5.4.1 验证KunDB系统节点全断电情况下的运行情况](#_Toc998558798)

[5.5 全断电测试](#_Toc761605458)

[5.5.1 验证KunDB系统节点全断网情况下的运行情况](#_Toc1857233583)

[5.6 逻辑备份还原](#_Toc566209161)

[5.6.1 验证KunDB系统逻辑备份还原功能](#_Toc5895008)

[5.7 完全备份还原](#_Toc355373131)

[5.7.1 验证KunDB系统完全备份还原功能](#_Toc1504786151)

[5.8 增量备份还原](#_Toc386110313)

[5.8.1 验证KunDB系统增量备份还原功能](#_Toc234396356)

[5.9 数据同步](#_Toc1502129928)

[5.9.1 验证KunDB系统支持两个数据库之间的数据同步功能](#_Toc1478115382)

[5.10 恢复点目标RPO](#_Toc1394934633)

[5.10.1 验证KunDB故障恢复后业务可正常进行功能](#_Toc888201104)

[6. 信息安全性](#_Toc539756120)

[6.1 身份鉴别](#_Toc598747671)

[6.1.1 验证KunDB系统用户身份进行鉴别功能](#_Toc862394364)

[6.1.2 验证KunDB系统用户身份鉴别信息加密存储功能](#_Toc1821405652)

[6.1.3 验证KunDB系统用户身份鉴别失败次数告警功能](#_Toc1558891493)

[6.1.4 验证KunDB系统定义用户登录失败最大次数](#_Toc1663018531)

[6.2 访问控制](#_Toc1993887616)

[6.2.1 验证KunDB支持用户数据库对象级访问控制功能](#_Toc1778267714)

[6.2.2 验证KunDB支持用户视图对象级访问控制功能](#_Toc309513622)

[6.2.3 验证KunDB支持用户表对象级访问控制功能](#_Toc1323317452)

[6.3 安全审计](#_Toc772020694)

[6.3.1 验证KunDB系统对用户身份认证操作的安全审计功能](#_Toc1054581659)

[6.3.2 验证KunDB系统对SQL操作的安全审计功能](#_Toc56077268)

[6.3.3 验证KunDB系统对配置操作的安全审计功能](#_Toc1094850756)

[6.3.4 验证KunDB安全审计实时报警功能](#_Toc840958527)

[7. 易用性](#_Toc1511143758)

[8. 可维护性](#_Toc927007225)

[8.1 错误检测与提示](#_Toc2031280110)

[8.1.1 验证对用户错误操作/命令的检查和提示功能](#_Toc362218908)

[8.1.2 验证对非法数据输入的检查和提示功能](#_Toc1688612683)

[8.2 日志信息](#_Toc1741030045)

[8.2.1 验证对KunDB系统记录实时日志功能](#_Toc928428070)

[8.3 性能监控](#_Toc1694507692)

[8.3.1 验证对KunDB系统CPU使用率监控功能](#_Toc2096403176)

[8.3.2 验证对KunDB系统内存的监控功能](#_Toc285730573)

[8.3.3 验证对KunDB系统磁盘空间使用率的监控功能](#_Toc2080618005)

[8.3.4 验证对KunDB系统Computer Node运行内存状态监控功能](#_Toc183315884)

[8.3.5 验证对KunDB系统Computer Node Mysql Server运行状态监控功能](#_Toc1787860502)

[8.3.6 验证对KunDB系统各服务运行状态监控功能](#_Toc1411249740)

[8.3.7 验证对KunDB系统KunGate运行内存状态监控功能](#_Toc1578250518)

[8.3.8 验证对KunDB系统KunGate服务执行缓存统计功能](#_Toc528577958)

[8.3.9 验证对KunDB系统KunGate查询统计功能](#_Toc1951005860)

[8.3.10 验证对KunDB系统KunGate事务统计功能](#_Toc29514541)

[8.3.11 验证对KunDB系统KunGate会话统计功能](#_Toc1390972322)

[8.3.12 验证对KunDB系统Kuntablet运行内存状态监控功能](#_Toc1624927865)

[8.3.13 验证对KunDB系统Kuntablet服务执行缓存统计功能](#_Toc1588406034)

[8.3.14 验证对KunDB系统Kuntablet查询统计功能](#_Toc906507205)

[8.3.15 验证对KunDB系统KunTablet MySQL Server运行状态监控功能](#_Toc1471331833)

[8.4 运维告警功能](#_Toc1219190100)

[8.4.1 验证对KunDB系统节点CPU使用过大告警功能](#_Toc1216020827)

[8.4.2 验证对KunDB系统节点内存使用过大告警功能](#_Toc647165638)

[8.4.3 验证对KunDB系统节点磁盘剩余空间过低告警功能](#_Toc1991210794)

[8.4.4 验证对KunDB系统节点失联告警功能](#_Toc123118838)

[8.4.5 验证对KunDB系统30分钟内慢查询次数过多告警功能](#_Toc703242906)

[8.4.6 验证对KunDB系统ComputeNode CPU 占用过多告警功能](#_Toc938577902)

[8.4.7 验证对KunDB系统ComputeNode 内存过大告警功能](#_Toc964077366)

[8.4.8 验证对KunDB系统Kundb 认证失败告警功能](#_Toc66903017)

[8.4.9 验证对KunDB系统越权操作告警功能](#_Toc1865585127)

[8.4.10 验证对KunDB系统锁超时警功能](#_Toc847873828)

[8.4.11 验证对KunDB系统相关服务不可用告警功能](#_Toc429121925)

[8.4.12 验证对KunDB系统KunGate CPU占用过多告警功能](#_Toc1406714163)

[8.4.13 验证对KunDB系统KunGate 内存过大告警功能](#_Toc441420225)

[8.4.14 验证对KunDB系统KunGate连接数过多告警功能](#_Toc1357549995)

[8.4.15 验证对KunDB系统KunTablet CPU占用过多告警功能](#_Toc953738207)

[8.4.16 验证对KunDB系统KunTablet 内存过大告警功能](#_Toc390339753)

[8.4.17 验证对KunDB系统Mysql 主从延时过高告警功能](#_Toc1643280569)

[8.5 数据库多副本同步](#_Toc886872564)

[8.5.1 验证对KunDB系统数据库多副本同步功能](#_Toc573655638)

[9. 可移植性](#_Toc1283657423)

[10. 兼容性](#_Toc150638656)

[11. 可扩展性](#_Toc4422508)

[11.1 弹性热扩展](#_Toc1812235381)

[11.1.1 验证KunDB系统弹性热扩展功能](#_Toc2101644517)

[11.2 在线缩容](#_Toc33937049)

[11.2.1 验证KunDB系统支持在线缩容功能](#_Toc1055724055)

[12. 附录1 BenchmarkSQL使用方法](#_Toc1579088734)

[13. 附录2 KunLoader使用方法](#_Toc1622343083)

[14. 附录3 KunDR使用方法](#_Toc1962231260)

[15. 附录4 CPU、内存、磁盘占用工具使用方法](#_Toc902936919)

[16. 附件5 LoadeRunner运行脚本WebService执行程序](#_Toc694049536)

[17. 附件6 增量备份相关工具和操作](#_Toc1030768439)

[17.1 增量备份与数据库同步测试案例中测试表的DDL:](#_Toc1550102557)

[17.2 增量备份与数据库同步测试案例中测试表数据生成脚本](#_Toc537776682)

[17.3 增量备份与数据库同步测试案例中测试表数据操作脚本](#_Toc1153887277)

[17.4 增量备份与数据库同步测试案例中测试表数据检查sql](#_Toc105861816)

[18. 附件7 数据同步相关工具和操作](#_Toc1476354585)

[18.1 创建增量备份任务的操作步骤](#_Toc2117964643)

[19. 附件8 manager集群部署两套kundb产品](#_Toc172764833)

[19.1 manager集群部署两套kundb产品的操作步骤](#_Toc1194456064)

# 整体概述

本报告的目的是详细描述星环分布式事务型数据库KunDB 1.2.1版本对于工信部提供的《产品质量测试大纲试大纲分布式数据库管理系统（事务处理型）》(以下称大纲)的满足程度，报告中包含大纲中要求的测试用例，测试脚本，测试数据，测试结果以及相关的测试方法，测试工具等。

其中，报告中第3~9章节分别验证大纲中第七章(测试内容和要求)中一至九节所描述的内容，需要注意的是，大纲中第七章第一节24项(配套工具)、第五节（易用性）、第七节(可移植性)、第八章节(兼容性)因为测试内容的特殊性，并不适合使用具体的测试用例进行描述，仅进行质量的定性描述，具体对应关系如下：

|  |  |  |
| --- | --- | --- |
| 报告内容 | 覆盖大纲内容 | 备注 |
| 第3章 功能性 | 第七章第一节 功能性 |  |
| 第4章 性能效率 | 第七章第二节 功能效率 |  |
| 第5章 可靠性 | 第七章第三节 可靠性 |  |
| 第6章 信息安全性 | 第七章第四节 信息安全性 |  |
| 第7章 易用性 | 第七章第五节 易用性 | 仅定性描述测试结果 |
| 第8章 可维护性 | 第七章第六节 维护性 |  |
| 第9章 可移植性 | 第七章第七节 可移植性 | 仅定性描述测试结果 |
| 第10章 兼容性 | 第七章第八节 兼容性 | 仅定性描述测试结果 |
| 第11章 可扩展性 | 第七章第九节 可扩展性 |  |

对于涉及到的测试工具的使用方法，以及相关的复杂操作步骤，在附件1~附件8中会有详细的描述。同时，需要强调的是，本报告中涉及到的部分测试工具，相关场景的测试方法等并非是强制性的，来源于星环公司内部的测试体系，如果有其他方法可达到同样目的，均可在测试中采用

# 测试结论

## 测试资源配置

### 硬件配置

|  |  |  |
| --- | --- | --- |
| **资源类型** | **资源配置** | **数量** |
| **长城服务器** | CPU：64\*core aarch64 FT2000PLUS  Memory：256 G  Disk：1\*SSD 745G  8\*机械盘 900G 两块Raid1,其余6块两两Raid0 | 6台 |
|  |  |  |

表1 硬件配置

### 软件配置

|  |  |  |
| --- | --- | --- |
| **软件类型** | **软件配置** | **数量** |
| **操作系统** | Linux version 4.19.0-arm64-server |  |
| **中间件** | 东方通应用服务器软件TongWeb V6.1 |  |
| **测试工具** | benchmarksql-5.0 |  |

表2 软件配置

## 测试网络拓补

### 角色分布拓补

每个节点均按照相同的磁盘规划进行分区配置，其中具体的分区类型、容量、和挂载点的信息如**表3 各个节点磁盘分区**所示。

|  |  |  |  |
| --- | --- | --- | --- |
| **磁盘** | **类型** | **容量** | **挂载点** |
| **DISK1** | 2\*机械盘 RAID1 | 900G | / |
| **DISK2** | 2\*机械盘 RAID0 | 1.8T | /mnt/disk1 |
| **DISK3** | 2\*机械盘 RAID0 | 1.8T | /mnt/disk2 |
| **DISK4** | 1\*SSD | 800G | /mnt/nvme1 |

表3 各个节点磁盘分区

每个节点上的角色配置，以及角色使用的磁盘分区信息如**表4 各个节点角色分布**所示，其中每个角色的定义，请参见文档**《KunDBManual.docx》**所述。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Master** | **Replica1** | **Replica2** | **Other** |
| **Node1** | Shard1(Master) | Shard5(Replica1) | Shard4(Replica2) | KunGate (1/6)  KunCtld (1/1) Orchestrator (1/1) |
| **Node2** | Shard2(Master) | Shard1(Replica1) | Shard5(Replica2) | KunGate (1/6) |
| **Node3** | Shard3(Master) | Shard2(Replica1) | Shard1(Replica2) | KunGate (1/6) Compute Node (1/4) |
| **Node4** | Shard4(Master) | Shard3(Replica1) | Shard2(Replica2) | KunGate (1/6) Compute Node (1/4) |
| **Node5** | Shard5(Master) | Shard4(Replica1) | Shard3(Replica2) | KunGate (1/6) KunCtld (1/2) Compute Node (1/4) |
| **Node6** |  |  |  | KunGate (1/6) LicenseProxy Compute Node (1/4) |

表4 各个节点角色分布

### Shard分布拓补

该测试供需创建5个Shard，每个Shard有一个Master角色，2个Replica角色。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Shard1** | **Shard2** | **Shard3** | **Shard4** | **Shard5** |
| **Master** | Node1 | Node2 | Node3 | Node4 | Node5 |
| **Replica1** | Node2 | Node3 | Node4 | Node5 | Node1 |
| **Replica2** | Node3 | Node4 | Node5 | Node1 | Node2 |

表5 各个节点角色分布

## 测试结论

经测试，KunDB 1.2.1产品完全**符合大纲要求**，具体内容如**表6 测试总体结论**所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 质量项 | 质量状态 | 不满足说明 | 备注 |
| 功能性 | 满足要求 | 无 | 1. 配套星环图形化工具Waterdrop 2. 配套星环数据加载工具KunDR 3. 配套星环数据同步工具TDT |
| 性能效率 | 1. TPC-C 1000仓数据的装载时间为11m 2. TPC-C 1000仓的1000并发模拟测试TPMC为20w左右 3. TPC-C 1000仓的3000并发模拟测试TPMC为20w左右 | 无 |  |
| 可靠性 | 满足要求 | 无 |  |
| 信息安全性 | 满足要求 | 无 |  |
| 易用性 | 满足要求 | 无 |  |
| 维护性 | 满足要求 | 无 |  |
| 可移植性 | 满足要求 | 无 | 支持飞腾、华为平台架构 |
| 兼容性 | 满足要求 | 无 | 1. 支持的中间件有东方通应用服务器软件TongWeb V6.1 2. 支持上层业务通过JDBC连接并使用数据库 3. 支持上层业务通过ODBC连接并使用数据库 |
| 可扩展性 | 满足要求 | 无 |  |

表6 测试总体结论

# 功能性

## 安装运行

### 验证KunDB部署功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB部署功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1、通过Transwarp Manager上传KunDB的安装包  2、根据安装文档《KunDBManual.docx》中安装部署的步骤，依次进行KunDB产品的安装 |
| 预期结果 | 1. KunDB的安装包上传成功，在Transwarp Manager的应用市场中可查看到KunDB的相关产品信息。 2. KunDB产品安装成功，通过MySQL客户端可成功连接KunDB。 |
| 测试脚本 | --MySQL客户端登录命令  *mysql -h<kungate-host> -P<port> -u<kungate\_username> -p<kungate\_password> --ssl-ca=/etc/kundb1/conf/ca-cert.pem --ssl-cert=/etc/kundb1/conf/kungate-client-cert.pem --ssl-key=/etc/kundb1/conf/kungate-client-key.pem* |
| 实际结果 | 1. KunDB的安装包上传成功，结果如下：      1. MySQL客户端登录成功，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 重新启动

### 验证KunDB服务重新启动功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB服务重新启动功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1、通过Transwarp Manager重新启动KunDB服务。 |
| 预期结果 | 1、KunDB服务可正常被重启，通过登录后台服务器，可以查询到角色服务的状态变化。 |
| 测试脚本 |  |
| 实际结果 | 1、KunDB服务重启成功，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 系统配置管理

### 验证KunDB系统配置功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB部署功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1、通过Transwarp Manager修改KunGate的audit\_filter的参数。  2、修改完成后，通过Transwarp Manager重新配置KunDB的服务。 |
| 预期结果 | 1、KunGate的audit\_filter可正确修改成功  2、KunDB重新配置服务成功 |
| 测试脚本 |  |
| 实际结果 | 1、KunGate的audit\_filte参数配置修改成功，结果如下：    2、KunDB重新配置服务成功，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 模式管理

### 验证KunDB创建模式功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB创建模式功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB已经建立链接 |
| 测试步骤 | 1. 通过Waterdrop工具，在KunDB系统中创创建一个新的模式schematest，SQL语句见测试脚本中描述 2. 显示模式列表，SQL语句见测试脚本中描述 |
| 预期结果 | 1. 模式schematest可成功创建。 2. 模式schematest在模式列表中存在。 |
| 测试脚本 | ***1、非图形化操作***  *--创建模式SQL语句*  *create schema if not exists schematest;*  *--显示模式信息SQL语句*  *show schemas;*  ***2、图形化操作***   1. *在左侧目录树的模式图标处右键，选择“新建数据库”*      1. *输入模式的名字，单击确定*      1. *在节点图标处右键，进行刷新*      1. *若建多个模式，可选择一个设定为激活* |
| 实际结果 | ***1、非图形化操作***  1、模式创建成功，结果如下：    2、模式schematest在模式列表中存在，结果如下：    ***2、图形化操作***  1、模式创建成功，在目录树下显示：     1. 创建多个模式后，模式激活成功 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB查看模式功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB查看模式功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1. 通过Waterdrop工具，在KunDB系统中创创建一个新的模式schematest，SQL语句见测试脚本中描述 2. 查看模式schematest的信息，SQL语句见测试脚本中描述 |
| 预期结果 | 1. 模式schematest可成功创建。 2. 模式schematest结果信息显示正确。 |
| 测试脚本 | ***1、非图形化操作***  *--查询模式信息SQL语句*  *create schema if not exists schematest;*  *--查询模式信息SQL语句*  *show create schema schematest;*  *2、****图形化操作***   1. *点击所建的模式左侧下拉箭头查看构建的模式结构* 2. *左键双击构建的模式，在右侧查看模式属性* |
| 实际结果 | ***1、非图形化操作***  1、模式创建成功，结果如下：    2、模式结构信息正确，结果如下：    **2、*图形化操作***   1. 目录树下模式结构信息正确：      1. 右侧模式属性信息展示正确： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB修改模式功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB修改模式功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1. 通过Waterdrop工具，在KunDB系统中创创建一个新的模式schematest，SQL语句见测试脚本中描述 2. 通过Waterdrop工具，修改模式schematest的CHARACTER SET属性，SQL语句见测试脚本中描述 3. 查看模式schematest的信息，SQL语句见测试脚本中描述 |
| 预期结果 | 1. 模式schematest可成功创建。 2. 模式schemates的属性修改成功 3. 模式schematest结果信息显示正确，属性值为修改后的属性。 |
| 测试脚本 | ***1、非图形化操作***  *--建表SQL语句*  *create schema if not exists schematest;*  *--查询模式信息SQL语句*  *show create schema schematest;*  *--修改模式SQL语句*  *alter schema schematest CHARACTER set = GBK ;*  ***2、图形化操作***  *修改字符信息*   1. *左键双击所建模式* 2. *在右侧属性栏中对默认字符集类别进行修改*      1. *在默认排序规则选项中对排序规则进行修改*      1. *使用ctrl +s 或软件右下角保存按钮进行保存，并执行语句* |
| 实际结果 | ***1、非图形化操作***  1、模式创建成功，结果如下：    2、模式schemates的属性修改成功，结果如下：     1. 模式schematest结果信息显示正确，属性值为修改后的属性，结果如下：     ***2、图形化操作***  字符集和排序规则修改成功： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB修改模式名称功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB修改模式名称功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1. 通过Waterdrop工具，在KunDB系统中创创建一个新的模式schematest，SQL语句见测试脚本中描述 2. 通过Waterdrop工具，修改模式schematest的名称为schematest1，SQL语句见测试脚本中描述 3. 查看模式schematest1的信息，SQL语句见测试脚本中描述 |
| 预期结果 | 1. 模式schematest可成功创建。 2. 模式schemates的名称修改成功 3. 模式schematest结果信息显示正确，属性值为修改后的属性。 |
| 测试脚本 | ***1、非图形化操作***  *--建表SQL语句*  *create schema if not exists schematest;*  *--查询模式信息SQL语句*  *show create schema schematest1;*  *--修改模式SQL语句*  *alter schema schematest rename to schematest1 ;*  ***2、图形化操作***  *在目录树下右键点击模式，选择重命名*    *修改模式名字后点击确定*    *在现实的命令行中执行重命名操作语句* |
| 实际结果 | ***1、非图形化操作***   1. 模式创建成功，结果如下：      1. 模式schemates的名称修改成功，结果如下：      1. 模式schematest1结果信息显示正确，结果如下：     ***2、图形化操作***  模式名称修改成功： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB删除模式功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB删除模式功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1. 通过Waterdrop工具，在KunDB系统中创创建一个新的模式schematest，SQL语句见测试脚本中描述 2. 显示模式列表，SQL语句见测试脚本中描述 3. 通过Waterdrop工具，删除模式schematest，SQL语句见测试脚本中描述 4. 显示模式列表，SQL语句见测试脚本中描述 |
| 预期结果 | 1. 模式schematest成功创建 2. 模式schematest在模式列表中存在 3. 模式schematest成功删除 4. 模式schematest在模式列表中不存在 |
| 测试脚本 | ***1、非图形化操作***  *--创建模式SQL语句*  *create schema if not exists schematest;*  *--显示模式信息SQL语句*  *show schemas;*  *--删除模式SQL语句*  *drop schema if exists schematest;*  ***2、图形化操作***  *右键点击右侧目录树下的模式，单击删除* |
| 实际结果 | ***非图形化操作***  1、模式创建成功，结果如下：    2、模式schematest在模式列表中存在，结果如下：    3、模式删除成功,结果如下：    4、模式schematest在模式列表中不存在，结果如下：    ***图形化操作***  所建的test 1 模式已经删除  &pfm111& |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 表对象管理

### 验证KunDB创建表对象功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB创建表对象功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在数据库dbtest1中执行show create table tb\_test1语句，可查看表的结构信息 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、显示的表结构信息，与建表语句中的一致。 |
| 测试脚本 | ***1. 非图形化操作***  *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *--查询表结构SQL语句*  *show create table tb\_test1;*  ***2. 图形化操作***  *1.右键点击“新建 表”*    *2.展开所建表，在“列”分栏下面右键点击“新建列”*    3.*根据需要设置列名和字段类型*     1. *使用 ctrl + s 或按钮进行保存，并执行语句* |
| 实际结果 | ***1.非图形化操作***  1、表创建成功，结果如下：    2、表结构信息正确，结果如下：    ***2.图形化操作***  表创建成功： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB删除表对象功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB删除表对象功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，删除表tb\_test1  3、通过Waterdrop工具，在数据库dbtest1中执行show create table tb\_test1语句 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表tb\_test1被成功删除  3、查询显示表不存在 |
| 测试脚本 | 1. ***非图形化操作***   *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *--删除表SQL语句*  *drop table if exists tb\_test1;*  *--查询表结构SQL语句*  *show create table tb\_test1;*   1. ***图形化操作*** 2. *在右侧目录树右键单击所建表，选择删除*      1. *单击“是”* |
| 实际结果 | ***1.非图形操作***  1、表创建成功，结果如下：  2、表删除成功，结果如下：    3、查询表结构信息，显示表不存在，结果如下：    ***2.图形操作***   1. 所建表已不显示在在右侧目录树中，表明已经删除 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB修改表名称功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB修改表名称功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，修改表tb\_test1的名称为tb\_test2  3、通过Waterdrop工具，在数据库dbtest1中执行show create table tb\_test2语句 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、表tb\_test1名称被成功修改，Waterdrop工具可显示表名称被修改成功  3、表tb\_test2的结构信息可被正常显示，与建表语句中的一致 |
| 测试脚本 | 1. ***非图形操作***   *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *--修改表名称SQL语句*  *rename table tb\_test1 to tb\_test2*  *--查询表结构SQL语句*  *show create table tb\_test1;*  *show create table tb\_test2;*   1. ***图形操作*** 2. *右键单击已保存并创建成功的表，选择“重命名”*      1. *修改名字后单击确定*      1. *执行重命名语句* |
| 实际结果 | ***1.非图形操作***  1、表创建成功，结果如下：    2、表名称修改成功，结果如下：    3、查询表结构信息正确，结果如下：    ***2. 图形操作***  1、所建表重命名成功且表结构信息完整 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB修改表列名称功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB修改表列名称功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，修改表tb\_test1的第一列名称，修改为a\_new  3、通过Waterdrop工具，在数据库dbtest1中执行show create table tb\_test1语句 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、表tb\_test1第一列名称被成功修改，Waterdrop工具可显示该列名称修改成功  3、表tb\_test1的结构信息可被正常显示，其中第一例的名称显示为修改后的值 |
| 测试脚本 | 1. ***非图形操作***   *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *--修改表列名称SQL语句*  *alter table tb\_test1 change column `b` `b\_new` int(255) not null;*  *--查询表结构SQL语句*  *show create table tb\_test1;*   1. ***图形操作*** 2. *右键单击所建成功的列，选择重命名*      1. *对所选列进行重命名并点击确定*      1. *ctrl+s或右下角按钮进行保存操作，并执行脚本语句* |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表列名称修改成功，结果如下：    3、查询表结构正确，结果如下：     1. ***图形操作***   表列名修改成功： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB修改表列类型功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB修改表列类型功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，修改表tb\_test1的第一列类型，修改为bigint  3、通过Waterdrop工具，在数据库dbtest1中执行show create table tb\_test1语句 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、表tb\_test1第一列类型被成功修改  3、表tb\_test1的结构信息可被正常显示，其中第一例的类型显示为修改后的值 |
| 测试脚本 | 1. ***非图形化操作***   *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *--修改表类型SQL语句*  *alter table tb\_test1 modify column `b` bigint not null ;*  *--查询表结构SQL语句*  *show create table tb\_test1;*   1. ***图形化操作*** 2. *在目录树左键双击想要修改的表列，在右侧属性栏中对类型进行修改*        1. ctrl +s进行保存并执行语句 |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表列名称修改成功，结果如下：    3、查询表结构信息正确，结果如下：     1. ***图形操作*** 2. 双击改后的表列，在右侧属性中可以看到已经修改成功： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB添加表字段功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB添加表字段功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，向表tb\_test1的添加一列new\_col，SQL语句见测试脚本中描述  3、通过Waterdrop工具，在数据库dbtest1中执行show create table tb\_test1语句 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表tb\_test1添加列信息成功  3、表tb\_test1的结构信息可被正常显示，其中新列new\_col的信息显示正确 |
| 测试脚本 | 1. ***非图形操作***   *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *--添加列SQL语句*  *alter table tb\_test1 add column `new\_col` varchar(255) NULL after `a` ;*  *--查询表结构SQL语句*  *show create table tb\_test1;*   1. ***图形操作*** 2. *右键单击目录树的所建表，选择新建列*      1. *对新建的表列进行设置，并单击确定*      1. *ctrl + s或右下角按钮进行保存，并执行语句* |
| 实际结果 | ***-- 非图形操作结果***  1、表创建成功，结果如下：    2、表列添加成功，结果如下：    3、查询表结构信息正确，结果如下：    ***-- 图形操作结果***  右侧目录树看出新建表列已生成 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB删除表字段功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB删除表字段功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，删除表tb\_test1的第一列a，SQL语句见测试脚本中描述  3、通过Waterdrop工具，在数据库dbtest1中执行show create table tb\_test1语句 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表tb\_test1第一列删除成功  3、表tb\_test1的结构信息可被正常显示，其中新列new\_col的信息显示正确 |
| 测试脚本 | 1. ***非图形操作***   *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *--删除列SQL语句*  *alter table tb\_test1 drop column `b`;*  *--查询表结构SQL语句*  *show create table tb\_test1;*   1. ***图形操作*** 2. *右键单击想要删除的列，点击“删除”*      1. *删除后，ctrl+s或右下角按钮进行保存，并执行语句* |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表列删除成功，结果如下：    3、查询表结构信息正确，结果如下：     1. ***图形操作***   右侧目录树的表中已成功删除所加的表列column 3： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 视图对象管理

* + - 1. **验证KunDB创建视图对象功能**

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB创建视图对象功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在数据库dbtest1中执行创建视图的SQL语句，创建视图view\_test  3、通过Waterdrop工具，在数据库dbtest1中执行show create view view\_test SQL语句，查询创建的视图信息 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、视图view\_test被创建成功。  3、视图view\_test结构信息可被正常查询和显示 |
| 测试脚本 | 1. *非图形化操作*   *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--创建视图SQL语句*  *create or replace view dbtest1.view\_test as select \* from dbtest1.tb\_test1;*  *--查询视图结构SQL语句*  *show create view dbtest1.view\_test;*   1. *图形化操作*   *-- 选中视图，右键点击“新建 视图”*    *-- 在视图配置页面中填写视图定义*    *-- 点击ctrl + s，保存执行* |
| 实际结果 | - 非图形化操作结果  1、表创建成功，结果如下：    2、视图被创建成功，结果如下：    3、查询视图结构信息正确，结果如下：    - 图形化操作结果  左侧目录树中将出现新建的视图 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

* + - 1. **验证KunDB删除视图对象功能**

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB删除视图对象功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在数据库dbtest1中执行创建视图的SQL语句，创建视图view\_test  3、通过Waterdrop工具，在数据库dbtest1中执行show create view view\_test SQL语句，查询创建的视图信息  4、通过Waterdrop工具，在数据库dbtest1中执行删除视图的SQL语句  5、通过Waterdrop工具，在数据库dbtest1中执行show create view view\_test SQL语句，查询创建的视图信息 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、视图view\_test被创建成功。  3、视图view\_test结构信息可被正常查询和显示  4、视图view\_test删除成功  5、查询视图view\_test显示视图不存在 |
| 测试脚本 | 1. *非图形化操作*   *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--删除视图SQL语句*  *drop view if exists dbtest1.view\_test;*  *--查询视图结构SQL语句*  *show create view dbtest1.view\_test;*   1. *图形化操作*   *-- 选中目标视图，右键点击“删除”* |
| 实际结果 | -- 非可视化操作结果  1、表创建成功，结果如下：    2、视图被创建成功，结果如下：    3、查询视图结构信息正确，结果如下：    4、视图删除成功，结果如下：    5、查询视图结构信息，报视图不存在，结果如下：    -- 可视化操作结果  左侧目录树中的视图消失。 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

* + - 1. **验证KunDB修改视图对象功能**

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB修改视图对象功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在数据库dbtest1中执行创建视图的SQL语句，创建视图view\_test  3、通过Waterdrop工具，在数据库dbtest1中执行show create view view\_test SQL语句，查询创建的视图信息  4、通过Waterdrop工具，在数据库dbtest1中执行修改视图的SQL语句  5、通过Waterdrop工具，在数据库dbtest1中执行show create view view\_test SQL语句，查询创建的视图信息 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、视图view\_test被创建成功。  3、视图view\_test结构信息可被正常查询和显示  4、视图view\_test修改成功  5、查询视图view\_test结构信息，显示为修改后的内容 |
| 测试脚本 | *1.非可视化操作*  *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--修改视图SQL语句*  *alter view dbtest1.view\_test as select \* from dbtest1.tb\_test1 limit 10;*  *--查询视图结构SQL语句*  *show create view dbtest1.view\_test;*  *2.可视化操作*  *- 选中目标视图，右键点击“编辑视图”或者双击视图，进入定义修改页面。修改视图的定义。*    *- 完成后点击ctrl + s，保存并执行修改* |
| 实际结果 | - 非可视化操作结果  1、表创建成功，结果如下：    2、视图被创建成功，结果如下：    3、查询视图结构信息正确，结果如下：    4、视图修改成功，结果如下：    5、查询视图结构信息正确，结果如下：    - 可视化操作结果  视图修改成功后，选中目标视图，右键点击“编辑视图”或者双击视图，进入定义查看页面，可以看到内容已经发生变化。 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

* + - 1. **验证KunDB修改视图名称功能**

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB修改视图名称功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在数据库dbtest1中执行创建视图的SQL语句，创建视图view\_test  3、通过Waterdrop工具，在数据库dbtest1中执行show create view view\_test SQL语句，查询创建的视图信息  4、通过Waterdrop工具，在数据库dbtest1中执行修改视图的SQL语句，修改视图名称为view\_test1  5、通过Waterdrop工具，在数据库dbtest1中执行show create view view\_test1 SQL语句，查询修改后的视图信息 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、视图view\_test被创建成功。  3、视图view\_test结构信息可被正常查询和显示  4、视图view\_test名称被成功修改为view\_test1  5、查询视图view\_test1结构信息，可正确显示 |
| 测试脚本 | 1. *非可视化操作步骤*   *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--修改视图名称SQL语句*  *rename table view\_test to view\_test1;*  *--查询视图结构SQL语句*  *show create view dbtest1.view\_test1;*   1. *可视化操作步骤*   *- 选中目标视图，右键点击“重命名”，然后修改名称。*    *- 点击“确定”预览，点击“执行”。* |
| 实际结果 | - 非可视化操作结果  1、表创建成功，结果如下：    2、视图被创建成功，结果如下：    3、查询视图结构信息正确，结果如下：    4、视图名称修改成功，结果如下：    5、查询视图结构信息正确，结果如下：    - 可视化操作结果  重命名视图名称后，左侧目录树生效。 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 数据对象管理

### 验证KunDB插入数据功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB插入数据功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在数据库dbtest1中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，查询插入的数据，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、表记录插入成功。  3、表记录查询成功，与插入的数据一致 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--插入记录SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test','2020-3-10 14:00:00','name','data',30,100);*  *--查询表数据SQL语句*  *select \* from tb\_test1 where id=1;* |
| 实际结果 | 1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表记录查询正确，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB更新数据功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB更新数据功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表tb\_test1中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，更新该行数据，SQL语句见测试脚本中描述  4、通过Waterdrop工具，查询该行数据，数据为更新后的数据，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表记录插入成功  3、表记录更新成功  4、表记录查询成功，与插入的数据一致 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--插入记录SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test','2020-3-10 14:00:00','name','data',30,100);*  *--更新记录SQL语句*  *update tb\_test1 set zjhm = 'test1', age=40, price=110 where id =1;*  *--查询表数据SQL语句*  *select \* from tb\_test1 where id=1;* |
| 实际结果 | 1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表记录更新成功，结果如下：    4、表记录查询正确，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB删除数据功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB删除数据功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表tb\_test1中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，删除该行数据，SQL语句见测试脚本中描述  4、通过Waterdrop工具，查询该行数据，数据为空，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表记录插入成功  3、表记录删除成功  4、表记录查询成功，查询不到该记录 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--删除记录SQL语句*  *delete from tb\_test1 where id=1;*  *--查询表数据SQL语句*  *select \* from tb\_test1 where id=1;* |
| 实际结果 | 1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表记录删除成功，结果如下：    4、表记录查询，查询不到该记录，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 用户权限管理

### 验证KunDB支持创建用户功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持创建用户功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB，登录命令见测试脚本描述 |
| 预期结果 | 1. Gudian Server 用户user\_test1创建成功 2. 数据库用户user\_test1创建成功 3. 通过用户user\_test1登录KunDB成功 |
| 测试脚本 | *--建表SQL语句*  *create user if not exists user\_test1;*  *--用户登录命令*  *mysql -h172.16.130.123 -P15307 -uuser\_test1 -p123 --ssl-ca=/etc/kundb1/conf/ca-cert.pem --ssl-cert=/etc/kundb1/conf/kungate-client-cert.pem --ssl-key=/etc/kundb1/conf/kungate-client-key.pem* |
| 实际结果 | 1. Gudian Server 用户user\_test1创建成功 2. 数据库用户user\_test1创建成功，结果如下：      1. 通过用户user\_test1登录KunDB成功，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持删除用户功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持删除用户功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB 4. 通过超级用户vt\_app登录KunDB,删除用户user\_test1，通过Gudian Server服务页面，删除用户user\_test1，SQL语句见测试脚本描述 5. 再次通过用户user\_test1登录KunDB |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功  3、通过用户user\_test1登录KunDB成功  4、数据库用户user\_test1删除成功，Gudian Server用户也删除成功  5、通过用户user\_test1登录KunDB失败 |
| 测试脚本 | *--创建用户SQL语句*  *create user user\_test1;*  *--删除用户SQL语句*  *drop user if exists user\_test1;* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，结果如下：    3、通过用户user\_test1登录KunDB成功，结果如下：     1. 数据库用户user\_test1删除成功，结果如下：      1. 通过用户user\_test1登录KunDB失败，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持授权用户数据库对象级权限功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持授权用户数据库对象级权限功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、KunDB中已存在数据库dbtest1  5、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB，并在数据库dbtest1中创建表*tb\_test1，SQL语句见测试脚本描述* 4. 通过超级用户vt\_app登录KunDB,给用户user\_test1赋予数据库dbtest1所有的操作权限，SQL语句见测试脚本描述 5. 再次通过用户user\_test1登录KunDB，并在数据库dbtest1中创建表*tb\_test1，SQL语句见测试脚本描述* |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功  3、通过用户user\_test1登录KunDB成功，建表失败，报没有权限错误  4、给用户user\_test1赋予数据库dbtest1所有的权限操作成功  5、通过用户user\_test1登录KunDB成功，建表成功 |
| 测试脚本 | *--创建用户SQL语句*  *create user user\_test1;*  *--删除用户SQL语句*  *drop user if exists user\_test1;*  *--授权用户SQL语句*  *grant create,drop,alter,select,insert,update,delete on dbtest1.\* to user\_test1;*  *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，结果如下：    3、通过用户user\_test1登录KunDB成功，建表失败，报没有权限错误，结果如下：    4、给用户user\_test1赋予数据库dbtest1所有的权限操作成功，结果如下：  5、通过用户user\_test1登录KunDB成功，建表成功，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持授权用户表对象级权限功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持授权用户表对象级权限功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、KunDB中已存在数据库dbtest1，且数据库存在表*tb\_test1*  5、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB，并在数据库dbtest1中表*tb\_test1插入记录，SQL语句见测试脚本描述* 4. 通过超级用户vt\_app登录KunDB,给用户user\_test1赋予表*tb\_test1*所有的操作权限，SQL语句见测试脚本描述 5. 再次通过用户user\_test1登录KunDB，并在数据库dbtest1中表*tb\_test1插入记录，SQL语句见测试脚本描述* |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功  3、通过用户user\_test1登录KunDB成功，数据插入失败，报没有权限错误  4、给用户user\_test1赋予表*tb\_test1*所有的权限操作成功  5、通过用户user\_test1登录KunDB成功，数据插入成功 |
| 测试脚本 | *--创建用户SQL语句*  *create user user\_test1;*  *--删除用户SQL语句*  *drop user if exists user\_test1;*  *--授权用户SQL语句*  *grant create,drop,alter,select,insert,update,delete on dbtest1.tb\_test1 to user\_test1;*  *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *--插入记录SQL语句*  *insert into tb\_test1(a,b,c) values(1,2,'test');* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，结果如下：    3、通过用户user\_test1登录KunDB成功，数据插入失败，报没有权限错误，结果如下：    4、给用户user\_test1赋予表*tb\_test1*所有的权限操作成功，结果如下：    5、通过用户user\_test1登录KunDB成功，数据插入成功，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持授权用户视图对象级权限功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持授权用户视图对象级权限功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、KunDB中已存在数据库dbtest1，且数据库存在表tb\_test1和视图view\_test  5、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，并创建视图view\_test,SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB，并在数据库dbtest1中查询视图view\_test*，SQL语句见测试脚本描述* 4. 通过超级用户vt\_app登录KunDB,给用户user\_test1赋予视图view\_test所有的操作权限，SQL语句见测试脚本描述 5. 再次通过用户user\_test1登录KunDB，并在数据库dbtest1中查询视图view\_test*，SQL语句见测试脚本描述* |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功,视图view\_test创建成功  3、通过用户user\_test1登录KunDB成功，查询视图失败，报没有权限错误  4、给用户user\_test1赋予视图view\_test所有的权限操作成功  5、通过用户user\_test1登录KunDB成功，查询视图成功 |
| 测试脚本 | *--创建用户SQL语句*  *create user user\_test1;*  *--删除用户SQL语句*  *drop user if exists user\_test1;*  *--授权用户视图权限SQL语句*  *grant all privileges on dbtest1.\* to user\_test1;*  *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *--创建视图SQL语句*  *create or replace view dbtest1.view\_test as select \* from dbtest1.tb\_test1;*  *--查询视图SQL语句*  *select \* from view\_test;* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，视图view\_test创建成功，结果如下：      3、通过用户user\_test1登录KunDB成功，查询视图失败，报没有权限错误，结果如下：    4、给用户user\_test1赋予查询视图所有的权限操作成功，结果如下：    5、通过用户user\_test1登录KunDB成功，查询视图成功，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 数据类型支持

### 验证KunDB支持基础数据类型功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持基础数据类型(整型、浮点型、字符型、日期类型等)功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中字段包含整型、浮点型、字符型、日期类型等，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表tb\_test1中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，查询插入的数据，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、表记录插入成功。  3、表记录查询成功，与插入的数据一致 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--插入记录SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test','2020-3-10 14:00:00','name','data',30,100);*  *--查询表数据SQL语句*  *select \* from tb\_test1 where id=1;* |
| 实际结果 | 1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表记录查询正确，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 大对象支持

### 验证KunDB支持大对象数据类型功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持大对象数据类型(CLOB、BOLB)功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中字段包含CLOB、BOLB数据类型，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表tb\_test1中插入一条记录，包含CLOB、BOLB类型列的值，SQL语句见测试脚本中描述  3、通过Waterdrop工具，查询插入的数据，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、表记录插入成功。  3、表记录查询成功，与插入的数据一致 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--插入记录SQL语句，load\_file文件根据实际情况修改*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`,`avatar\_small`,`extra\_info`)*  *values(1,'test','2020-3-10 14:00:00','name','data',30,100,load\_file("/home/transwarp/test\_pic.png"),'big text');*  *--查询表数据SQL语句*  *select id,avatar\_small,extra\_info from tb\_test1 where id=1;* |
| 实际结果 | 1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表记录查询正确，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 约束

### 验证KunDB支持主键约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持主键约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中存在主键，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表tb\_test1中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，在表tb\_test1中再插入一条记录，主键值相同，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、表记录插入成功。  3、表记录插入失败，报主键冲突 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--插入记录SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test','2020-3-10 14:00:00','name','data',30,100);* |
| 实际结果 | ***1.非图形操作：***  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表记录插入失败，报主键冲突，结果如下：    ***2.图形操作：***  1.创建表时创建主键    2. 主键创建成功      3. 插入数据成功    4. 插入记录失败，主键冲突 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持删除主键约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持删除主键约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中存在主键，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表tb\_test1中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，删除表tb\_test1的主键约束，SQL语句见测试脚本中描述  4、通过Waterdrop工具，在表tb\_test1中再插入一条记录，主键值相同，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表记录插入成功  3、表tb\_test1主键删除成功  4、表记录插入成功 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--插入记录SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test','2020-3-10 14:00:00','name','data',30,100);*  *--删除表主键约束SQL语句*  *alter table tb\_test1 drop primary key;* |
| 实际结果 | **1. 非图形化操作**  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表主键约束删除成功，结果如下：    4、表记录插入成功，结果如下：    **2.图形化操作**  1.删除主键    2. 删除成功    3. 插入相同数据成功 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持禁用/启用主键约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持禁用/启用主键约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中存在主键，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表tb\_test1中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，删除表tb\_test1的主键约束，SQL语句见测试脚本中描述  4、通过Waterdrop工具，在表tb\_test1中再插入一条记录，主键值相同，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表记录插入成功  3、表tb\_test1主键删除成功  4、表记录插入成功 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--插入记录SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test','2020-3-10 14:00:00','name','data',30,100);*  *--删除表主键约束SQL语句*  *alter table tb\_test1 drop primary key;* |
| 实际结果 | **1. 非图形化操作**  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表主键约束禁用成功，结果如下：      4、表记录插入成功，结果如下：    **2.图形化操作**  1.删除主键    2. 删除成功    3. 插入相同数据成功 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持多健值主键约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持多健值主键约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中存在多健值主键，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表tb\_test1中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，在表tb\_test1中再插入一条记录，主键值全部相同，SQL语句见测试脚本中描述  4、通过Waterdrop工具，在表tb\_test1中再插入一条记录，主键值不全相同，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表记录插入成功  3、表记录插入失败，报主键冲突  4、表记录插入成功 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`,`age`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--插入记录SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test','2020-3-10 14:00:00','name','data',30,100);* |
| 实际结果 | **1.非图形化操作**  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表记录插入失败，报主键冲突，结果如下：    **2. 图形化操作**  1. 创建主键    2. 创建成功    3. 插入数据成功    4. 插入相同数据，主键冲突 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持唯一约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持唯一约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表*constraint\_test*，表中存在主键、唯一约束，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表*constraint\_test*中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，在表*constraint\_test*中再插入一条记录，唯一约束列值相同，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表*constraint\_test*可成功创建。  2、表记录插入成功。  3、表记录插入失败，报唯一约束冲突 |
| 测试脚本 | *--建表SQL语句*  *create table if not exists constraint\_test(id int not null,name varchar(200) unique,additional\_message varchar(200) ,primary key(id))partition by hash(id) USING `binary\_md5`;*  *--插入记录SQL语句*  *insert into constraint\_test values(1,'xiaohong','xiaoxuesheng');*  *insert into constraint\_test values(2,'xiaohong','xiaoxuesheng');* |
| 实际结果 | **非图形化操作**  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表记录插入失败，报唯一约束冲突，结果如下：    **2. 图形化操作**  1.创建唯一约束    2. 创建成功    3.插入数据成功    4.插入相同数据失败 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持删除唯一约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持删除唯一约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表*constraint\_test*，表中存在主键、唯一约束，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表*constraint\_test*中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，删除表*constraint\_test*中唯一约束，SQL语句见测试脚本中描  4、通过Waterdrop工具，在表*constraint\_test*中再插入一条记录，原唯一约束列值相同，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表*constraint\_test*可成功创建  2、表记录插入成功  3、表*constraint\_test*唯一约束删除成功  4、表记录插入失败，报唯一约束冲突 |
| 测试脚本 | *--建表SQL语句*  *create table if not exists constraint\_test(id int not null,name varchar(200) unique,additional\_message varchar(200) ,primary key(id))partition by hash(id) USING `binary\_md5`;*  *--删除唯一约束SQL语句*  *ALTER TABLE constraint\_test DROP INDEX name;*  *--插入记录SQL语句*  *insert into constraint\_test values(1,'xiaohong','xiaoxuesheng');*  *insert into constraint\_test values(2,'xiaohong','xiaoxuesheng');* |
| 实际结果 | **1. 非图形化操作**  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表唯一约束删除成功，结果如下：    4、表记录插入成功，结果如下:    **2.图形化操作**  1.删除唯一约束    2.删除成功    3.插入相同数据成功 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持添加唯一约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持添加唯一约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表*constraint\_test*，表中存在主键、唯一约束，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表*constraint\_test*中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，在表*constraint\_test*中在添加一个唯一约束，SQL语句见测试脚本中描  4、通过Waterdrop工具，在表*constraint\_test*中再插入一条记录，新添加的唯一约束列值相同，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表*constraint\_test*可成功创建  2、表记录插入成功  3、表*constraint\_test*新增唯一约束成功  4、表记录插入失败，报唯一约束冲突 |
| 测试脚本 | *--建表SQL语句*  *create table if not exists constraint\_test(id int not null,name varchar(200) unique,additional\_message varchar(200) ,primary key(id))partition by hash(id) USING `binary\_md5`;*  *--添加唯一约束SQL语句*  *ALTER TABLE constraint\_test ADD CONSTRAINT UN\_ID UNIQUE(additional\_message);*  *--插入记录SQL语句*  *insert into constraint\_test values(1,'xiaohong','xiaoxuesheng');*  *insert into constraint\_test values(2,'xiaohong1','xiaoxuesheng');* |
| 实际结果 | **1. 非图形化操作**  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表唯一约束添加成功，结果如下：    4、表记录插入失败，报唯一约束冲突，结果如下:    **2. 图形化操作**  1.创建唯一约束    2. 创建成功    3.插入数据成功    4.插入相同数据失败 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持禁用/启用唯一约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持禁用/启用唯一约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中存在主键，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表tb\_test1中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，删除表tb\_test1的主键约束，SQL语句见测试脚本中描述  4、通过Waterdrop工具，在表tb\_test1中再插入一条记录，主键值相同，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表记录插入成功  3、表tb\_test1主键删除成功  4、表记录插入成功 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--插入记录SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test','2020-3-10 14:00:00','name','data',30,100);*  *--删除表主键约束SQL语句*  *alter table tb\_test1 drop primary key;* |
| 实际结果 | **1. 非图形化操作**  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表主键约束禁用成功，结果如下：      4、表记录插入成功，结果如下：    **2.图形化操作**  1.禁用主键  2. 删除成功    3. 插入相同数据成功 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持非空约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持非空约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表*constraint\_test*，表中存在主键、非空约束，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表*constraint\_test*中插入一条记录，非空约束列的值为null，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表*constraint\_test*可成功创建  2、表记录插入失败，报非空约束冲突 |
| 测试脚本 | *--建表SQL语句*  *drop table if exists constraint\_test;*  *create table if not exists constraint\_test(id int not null,name varchar(200) unique,*  *additional\_message varchar(200) not null,primary key(id))partition by hash(id) USING `binary\_md5`;*  *--插入记录SQL语句*  *insert into constraint\_test values(1,'xiaohong',null);* |
| 实际结果 | 1**. 非图形化操作**  1、表创建成功，结果如下：    2、表记录插入失败，报非空约束冲突，结果如下：    **2. 图形化操作**  1. 创建表    2. 插入非空数据    3. 插入null数据错误 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持删除非空约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持删除非空约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表*constraint\_test*，表中存在主键、非空约束，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表*constraint\_test*中插入一条记录，非空约束列的值为null，SQL语句见测试脚本中描述  3、通过Waterdrop工具，删除表*constraint\_test*的非空约束  4、通过Waterdrop工具，在表*constraint\_test*中插入一条记录，原非空约束列的值为null，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表*constraint\_test*可成功创建  2、表记录插入失败，报非空约束冲突  3、表*constraint\_test*非空约束删除成功  4、表记录插入成功 |
| 测试脚本 | *--建表SQL语句*  *drop table if exists constraint\_test;*  *create table if not exists constraint\_test(id int not null,name varchar(200) unique,*  *additional\_message varchar(200) not null,primary key(id))partition by hash(id) USING `binary\_md5`;*  *--删除表非空约束SQL语句*  *ALTER TABLE constraint\_test MODIFY additional\_message VARCHAR(200);*  *--插入记录SQL语句*  *insert into constraint\_test values(1,'xiaohong',null);* |
| 实际结果 | **1. 非图形化操作**  1、表创建成功，结果如下：    2、表记录插入失败，报非空约束冲突，结果如下：    3、表的非空约束删除成功，结果如下：    4、表记录插入成功，结果如下：    **2. 图形化操作**  1. 创建表    2. 插入null失败    3.删除null约束      4. 插入null数据 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持添加非空约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持添加非空约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表*constraint\_test*，表中存在主键、非空约束，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表*constraint\_test*中新添加一个非空约束，SQL语句见测试脚本中描述  3、通过Waterdrop工具，在表*constraint\_test*中插入一条记录，新增非空约束列的值为null，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表*constraint\_test*可成功创建  2、表*constraint\_test*新增非空约束成功  3、表记录插入失败，报非空约束冲突 |
| 测试脚本 | *--建表SQL语句*  *drop table if exists constraint\_test;*  *create table if not exists constraint\_test(id int not null,name varchar(200) unique,*  *additional\_message varchar(200) not null,*  *additional\_text varchar(200),*  *primary key(id))partition by hash(id) USING `binary\_md5`;*  *--新增表非空约束SQL语句*  *ALTER TABLE constraint\_test MODIFY additional\_text VARCHAR(200) NOT NULL;*  *--插入记录SQL语句*  *insert into constraint\_test values(1,'xiaohong',null);* |
| 实际结果 | **1. 非图形化操作**  1、表创建成功，结果如下：    2、表的非空约束新增成功，结果如下：    3、表记录插入失败，报非空约束冲突，结果如下：    **2. 图形化操作**  1. 非图形化操作  1. 创建表    2. 插入null数据    3. 增加null约束    4. 插入null数据失败 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持外健约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持主非空约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表child、parent，parent表中存在主键，child表中存在外健，为parent的主键，SQL语句见测试脚本中描述  2、通过Waterdrop工具，向表parent中插入一些数据，SQL语句见测试脚本中描述  3、通过Waterdrop工具，在表child中插入几条记录，外健约束列的值在parent表中存在，SQL语句见测试脚本中描述  4、通过Waterdrop工具，在表child中插入一条记录，外健约束列的值在parent表中不存在，SQL语句见测试脚本中描述  5、通过Waterdrop工具，删除表parent中的记录，该记录主键仍然被child表中数据外健引用，SQL语句见测试脚本中描述  6、通过Waterdrop工具，更新表parent中的记录，该记录主键仍然被child表中数据外健引用，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表记录插入成功  3、表记录插入成功  4、表记录插入失败，报外健冲突  5、表记录删除失败，报外健冲突  6、表记录更新失败，报外健冲突 |
| 测试脚本 | *--建表SQL语句*  *drop table if exists child;*  *drop table if exists parent;*  *create table parent(id INT not null, primary key (id));*  *create table child(*  *`id` int(0) not null,*  *`parent\_id` int(0) not null,*  *primary key (`id`),*  *constraint `test` foreign key (`parent\_id`) references `parent` (`id`) on delete restrict on update restrict);*  *--parent表插入记录SQL语句*  *INSERT INTO parent(id) VALUES(1),(2),(3),(4);*  *INSERT INTO parent(id) VALUES(5),(6),(7),(8),(9);*  *--child表插入数据记录SQL，外健存在*  *insert into child(id,parent\_id) VALUES(101,1),(102,1),(103,1);*  *insert into child(id,parent\_id) VALUES(201,2),(301,3),(401,4),(501,5),(601,6),(701,7),(801,8);*  *--parent表删除记录SQL语句*  *delete from parent where id=1;*  *--parent表更新记录SQL语句*  *delete from parent where id=1;* |
| 实际结果 | 1、表创建成功，结果如下：      2、parent表插入成功，结果如下：    3、child表记录插入成功，结果如下：    4、child表记录插入失败，报外健冲突，结果如下：    5、parent表记录删除失败，报外健冲突，结果如下：    6、parent表记录更新失败，报外健冲突，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持禁用/启用外健约束功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持禁用/启用外健约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中存在主键，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表tb\_test1中插入一条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，删除表tb\_test1的主键约束，SQL语句见测试脚本中描述  4、通过Waterdrop工具，在表tb\_test1中再插入一条记录，主键值相同，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表记录插入成功  3、表tb\_test1主键删除成功  4、表记录插入成功 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--插入记录SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test','2020-3-10 14:00:00','name','data',30,100);*  *--删除表主键约束SQL语句*  *alter table tb\_test1 drop primary key;* |
| 实际结果 | **1. 非图形化操作**  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、表主键约束禁用成功，结果如下：    4、表记录插入成功，结果如下：    **2.图形化操作**  1.删除主键    2. 删除成功    3. 插入相同数据成功 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

* + - 1. **验证KunDB支持check约束功能**

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持主check约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表t0，存在check约束，SQL语句见测试脚本中描述  2、通过Waterdrop工具，向表t0中插入一些数据，存在check约束不通过的数据，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表记录中check约束不通过的数据插入失败，其他的插入成功 |
| 测试脚本 | 1. ***非图形操作***   *--建表SQL语句*  *drop table if exists t0;*  *create table t0(id int, name varchar(20), age int constraint less20 check(age < 20) , age1 int constraint gt30 check(age1 > 30) ) partition by hash(id) using binary\_md5;*  *--表插入记录SQL语句*  *insert into t0(id, name, age, age1) values(1,'a',19, 31);*  *insert into t0(id, name, age, age1) values(2,'b',20, 32);*  *insert into t0(id, name, age, age1) values(3,'c',19, 30);*  *insert into t0(id, name, age, age1) values(4,'d',20, 30);*   1. ***图形操作***   *1.创建表和列，并保存*    *2. 右键单击检查约束，选择“新建 检查约束”*    *3.根据需求设置check约束的名称，并输入check约束表达式，单击确定*    *4.control+s或按钮保存，执行语句*    *5.右键单击创建好的check约束，点击编辑约束进行属性查看*    *6.插入数据（5，10，15，25）对check约束进行验证并目录树刷新* |
| 实际结果 | ***1.非图形操作***  1、表创建成功，结果如下：    2、表记录中check约束不通过的数据插入失败，其他的插入成功，结果如下：      ***2.图形操作***  1、check约束创建成功，在目录树下显示    2.、右侧属性栏展示成功：    3、 符合check约束条件的数据插入成功，不符合的数据未插入 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

* + - 1. **验证KunDB支持修改check约束功能**

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持主check约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表t0，存在check约束，SQL语句见测试脚本中描述  2、通过Waterdrop工具，向表t0中插入一些数据，存在check约束不通过的数据，SQL语句见测试脚本中描述  3、通过waterdrop工具，修改表t0的check约束，修改成功  4、通过Waterdrop工具，向表t0中插入一些数据，存在check约束不通过的数据，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表记录中check约束不通过的数据插入失败，其他的插入成功  3、表t0的check约束修改成功  4、表记录中按照修改后的check约束不通过的数据插入失败，其他的插入成功 |
| 测试脚本 | 1. ***非图形操作***   *--建表SQL语句*  *drop table if exists t0;*  *create table t0(id int, name varchar(20), age int constraint less20 check(age < 20) , age1 int constraint gt30 check(age1 > 30) ) partition by hash(id) using binary\_md5;*  *--表插入记录SQL语句*  *insert into t0(id, name, age, age1) values(1,'a',19, 31);*  *insert into t0(id, name, age, age1) values(2,'b',20, 32);*  *insert into t0(id, name, age, age1) values(3,'c',19, 30);*  *insert into t0(id, name, age, age1) values(4,'d',20, 30);*  *--修改check约束条件*  *alter table t0 modify constraint gt30 enable novalidate;*  *ALTER TABLE dbtest1.t0 MODIFY CONSTRAINT gt30 TO CHECK(age1 >20) ;*   1. ***图形操作***   *1.创建表和列，并保存*    *2. 右键单击检查约束，选择“新建 检查约束”*    *3.根据需求设置check约束的名称，并输入约束表达式，单击确定*    *4.control+s或按钮保存，执行语句*    *5.右键单击创建好的check约束，点击编辑约束进行属性查看*    *6.插入数据（5，10，15，25）对check约束进行验证并目录树刷新*  *7.在属性栏中对表达式进行修改*    *8.control+s或按钮进行保存，并执行语句*    *9. 插入新数据（25，35）进行验证并目录树刷新* |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表记录中check约束不通过的数据插入失败，其他的插入成功，结果如下：      3、表t0的check约束修改成功，结果如下：     1. 表记录中按照修改后的check约束不通过的数据插入失败，其他的插入成功，结果如下：       ***2.图形操作***  1、check约束创建成功，在目录树下显示    2.、右侧属性栏展示成功：    3、 符合check约束条件的数据插入成功，不符合的数据未插入    4、check约束修改成功    5、符合check约束的数据插入成功，不符合数据未插入，证明check约束修改成功 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

* + - 1. **验证KunDB支持启动/禁用check约束功能**

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持禁用check约束功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表t0，存在check约束，SQL语句见测试脚本中描述  2、通过Waterdrop工具，向表t0中插入一些数据，存在check约束不通过的数据，SQL语句见测试脚本中描述  3、通过waterdrop工具，禁用表t0的check约束，禁用成功  （注：disable novalidate 和disable validate语句功能一致，均为不做任何检查）  4、通过Waterdrop工具，向表t0中插入一些数据，存在check约束不通过的数据，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表记录中check约束不通过的数据插入失败，其他的插入成功  3、表t0的check约束禁用成功  4、表禁用的check约束不在起作用，按照未禁用的check约束不通过的数据插入失败，其他的插入成功 |
| 测试脚本 | ***1、非图形操作***  *--建表SQL语句*  *drop table if exists t0;*  *create table t0(id int, name varchar(20), age int constraint less20 check(age < 20) , age1 int constraint gt30 check(age1 > 30) ) partition by hash(id) using binary\_md5;*  *--表插入记录SQL语句*  *insert into t0(id, name, age, age1) values(1,'a',19, 31);*  *insert into t0(id, name, age, age1) values(2,'b',20, 32);*  *insert into t0(id, name, age, age1) values(3,'c',19, 30);*  *insert into t0(id, name, age, age1) values(4,'d',20, 30);*  *--禁用约束*  *ALTER TABLE dbtest1.t0 MODIFY CONSTRAINT gt30 disable validate ;*  （注：disable validate 和disable novalidate语句功能一致，均为不做任何检查）  ***2、图形操作***  *1.创建表和列，并保存*    *2. 右键单击检查约束，选择“新建 检查约束”*    *3.根据需求设置check约束的名称，并输入约束表达式，单击确定*    *4.control+s或按钮保存，执行语句*    *5.右键单击创建好的check约束，点击编辑check约束进行属性查看*    *6.插入数据（5，10，15，25）对check约束进行验证并目录树刷新*  *7.在属性中将激活栏取消勾选，并保存*  （注：disable validate 和disable novalidate语句功能一致，均为不做任何检查）    *8.插入新数据（25，30）并刷新进行验证* |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表记录中check约束不通过的数据插入失败，其他的插入成功，结果如下：      3、表t0的check约束禁用成功，结果如下：    4、表禁用的check约束不再起作用，按照未禁用的check约束，不通过未禁用约束的数据插入失败，其他的插入成功，结果如下：      ***2.图形操作***  1、check约束创建成功，在目录树下显示    2.、右侧属性栏展示成功：    3、 符合check约束条件的数据插入成功，不符合的数据未插入    4、检查约束禁用成功，保存刷新后属性栏已无激活勾选    5、插入不符合原约束条件的新数据成功，证明检查约束禁止成功 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 存储对象使用管理

### 验证KunDB支持存储过程使用动态SQL功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持存储过程的调用、使用动态SQL、游标功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，创建存储过程NewProc，存在动态SQL，SQL语句见测试脚本中描述  3、通过Waterdrop工具，调用存储过程NewProc |
| 预期结果 | 1、表tb\_test1可成功创建  2、存储过程NewProc创建成功  3、存储过程调用成功，表tb\_test1中存在被存储过程插入的数据 |
| 测试脚本 | 1. *非可视化操作*   *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--创建存储过程NewProc SQL语句*  *delimiter $$*  *create PROCEDURE `NewProc`(IN `id\_in` BIGINT(20), IN `zjhm\_in` varchar(30), IN `name\_in` varchar(30))*  *BEGIN*  *declare SQL\_FOR\_INSERT varchar(500);*  *set SQL\_FOR\_INSERT = "insert into tb\_test1(`id`,`name`,`zjhm`) values(?,?,?) ";*  *set @sql = SQL\_FOR\_INSERT;*  *PREPARE stmt FROM @sql;*  *set @parm1 = id\_in;*  *set @parm2 = name\_in;*  *set @parm3 = zjhm\_in;*  *EXECUTE stmt USING @parm1 , @parm2, @parm3;*  *deallocate prepare stmt;*  *END*  *$$*  *delimiter;*  *--调用存储过程语句*  *call NewProc(99, "name", "zjhm");*   1. *可视化操作*   *- 选中存储过程列表，点击“新建存储过程”*    *- 定义存储过程名称*    *- 在“源码”填写存储过程定义*    *- ctrl+s保存改动，并点击执行*    *- 可以选中目标存储过程，查看分区内容* |
| 实际结果 | - 非可视化操作结果  1、表创建成功，结果如下：    2、存储过程创建成功，结果如下：    3、存储过程调用成功，结果如下：      - 可视化操作结果  可以从左侧目录树中查看到创建的存储过程。 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装ji文档《KunDBManual.docx》中描述 |

* + - 1. **验证KunDB支持修改存储过程功能**

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持修改存储过程功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，创建存储过程NewProc，存在动态SQL，SQL语句见测试脚本中描述  3、通过Waterdrop工具，修改存储过程NewProc，修改comment为NewPro\_test，SQL语句见测试脚本中描述  4、通过Waterdrop工具，调用存储过程NewProc |
| 预期结果 | 1、表tb\_test1可成功创建  2、存储过程NewProc创建成功  3、存储过程NewProc修改成功  4、存储过程调用成功，表tb\_test1中存在被存储过程插入的数据 |
| 测试脚本 | 1. *非可视化操作*   *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--创建存储过程NewProc SQL语句*  *delimiter $$*  *create PROCEDURE `NewProc`(IN `id\_in` BIGINT(20), IN `zjhm\_in` varchar(30), IN `name\_in` varchar(30))*  *BEGIN*  *declare SQL\_FOR\_INSERT varchar(500);*  *set SQL\_FOR\_INSERT = "insert into tb\_test1(`id`,`name`,`zjhm`) values(?,?,?) ";*  *set @sql = SQL\_FOR\_INSERT;*  *PREPARE stmt FROM @sql;*  *set @parm1 = id\_in;*  *set @parm2 = name\_in;*  *set @parm3 = zjhm\_in;*  *EXECUTE stmt USING @parm1 , @parm2, @parm3;*  *deallocate prepare stmt;*  *END*  *$$*  *delimiter;*  *--调用存储过程语句*  *call NewProc(99, "name", "zjhm");*  *--修改存储过程语句*  *delimiter $$*  *create or replace PROCEDURE `NewProc`(IN `id\_in` BIGINT(20), IN `zjhm\_in` varchar(30), IN `name\_in` varchar(30))*  *BEGIN*  *declare SQL\_FOR\_INSERT varchar(500);*  *set SQL\_FOR\_INSERT = "insert into tb\_test1(`id`,`name`,`zjhm`) values(?,?,?) ";*  *set @sql = SQL\_FOR\_INSERT;*  *PREPARE stmt FROM @sql;*  *set @parm1 = id\_in+1;*  *set @parm2 = name\_in;*  *set @parm3 = zjhm\_in;*  *EXECUTE stmt USING @parm1 , @parm2, @parm3;*  *deallocate prepare stmt;*  *END*  *$$*  *delimiter;*   1. *可视化操作过程*   *- 选中目标存储过程，右键点击编辑存储过程*    *- 在源码编辑框中修改存储过程内容*    *- 完成后点击按下ctrl+s，保存改动并执行* |
| 实际结果 | -- 非可视化操作结果  1、表创建成功，结果如下：    2、存储过程创建成功，结果如下：    3、存储过程修改成功，结果如下：    4、存储过程调用成功，结果如下：      -- 可视化操作结果  查看发现存储过程的内容已经修改 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持修改存储过程名称功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持修改存储过程名称功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，创建存储过程NewProc，SQL语句见测试脚本中描述  3、通过Waterdrop工具，调用存储过程NewProc，SQL语句见测试脚本中描述  4、通过Waterdrop工具，修改存储过程NewProc名称为NewProc1，SQL语句见测试脚本中描述  5、通过Waterdrop工具，调用存储过程NewProc1，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、存储过程NewProc创建成功  3、存储过程调用成功，表tb\_test1中存在被存储过程插入的数据  4、存储过程名称修改成功  5、存储过程调用成功，表tb\_test1中存在被存储过程插入的数据 |
| 测试脚本 | 1. *非可视化操作*   *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--创建存储过程NewProc SQL语句*  *delimiter $$*  *create PROCEDURE `NewProc`(IN `id\_in` BIGINT(20), IN `zjhm\_in` varchar(30), IN `name\_in` varchar(30))*  *BEGIN*  *declare SQL\_FOR\_INSERT varchar(500);*  *set SQL\_FOR\_INSERT = "insert into tb\_test1(`id`,`name`,`zjhm`) values(?,?,?) ";*  *set @sql = SQL\_FOR\_INSERT;*  *PREPARE stmt FROM @sql;*  *set @parm1 = id\_in;*  *set @parm2 = name\_in;*  *set @parm3 = zjhm\_in;*  *EXECUTE stmt USING @parm1 , @parm2, @parm3;*  *deallocate prepare stmt;*  *END*  *$$*  *delimiter;*  *--调用存储过程SQL语句*  *call NewProc(99, "name", "zjhm");*  *call NewProc1(100, "name1", "zjhm1");*  *--修改存储过程名称SQL语句*  *alter procedure NewProc rename to NewProc1;*   1. 可视化操作   - 选中目标存储过程，右键选择重命名    - 填写重命名的名称    - 点击确定后，保存修改并执行 |
| 实际结果 | -- 非可视化操作结果  1、表创建成功，结果如下：    2、存储过程创建成功，结果如下：    3、存储过程调用成功，结果如下：      存储过程名称修改成功，结果如下：    存储过程调用成功：      -- 可视化操作结果，发现重名称成功 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持删除存储过程功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持删除存储过程功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，创建存储过程NewProc，SQL语句见测试脚本中描述  3、通过Waterdrop工具，调用存储过程NewProc  4、通过Waterdrop工具，删除存储过程NewProc，SQL语句见测试脚本中描述  5、通过Waterdrop工具，调用已删除的存储过程NewProc |
| 预期结果 | 1、表tb\_test1可成功创建  2、存储过程NewProc创建成功  3、存储过程调用成功，表tb\_test1中存在被存储过程插入的数据  4、存储过程NewProc删除成功  5、存储过程调用失败，报存储过程不存在 |
| 测试脚本 | 1. *非可视化操作*   *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--创建存储过程NewProc SQL语句*  *delimiter $$*  *create PROCEDURE `NewProc`(IN `id\_in` BIGINT(20), IN `zjhm\_in` varchar(30), IN `name\_in` varchar(30))*  *BEGIN*  *declare SQL\_FOR\_INSERT varchar(500);*  *set SQL\_FOR\_INSERT = "insert into test\_1(`id`,`name`,`zjhm`) values(?,?,?) ";*  *set @sql = SQL\_FOR\_INSERT;*  *PREPARE stmt FROM @sql;*  *set @parm1 = id\_in;*  *set @parm2 = name\_in;*  *set @parm3 = zjhm\_in;*  *EXECUTE stmt USING @parm1 , @parm2, @parm3;*  *deallocate prepare stmt;*  *END*  *$$*  *delimiter;*  *--调用存储过程语句*  *call NewProc(99, "name", "zjhm");*  *--删除存储过程语句*  *drop PROCEDURE IF EXISTS NewProc;*   1. *可视化操作*   *-- 选中目标存储过程，右键选择删除*    *-- 点击确定* |
| 实际结果 | -- 非可视化操作结果  1、表创建成功，结果如下：    2、存储过程创建成功，结果如下：    3、存储过程调用成功，结果如下：      4、存储过程删除成功，结果如下：    5、存储过程调用失败，报存储过程不存在错误，结果如下：    -- 可视化操作结果  该表被成功删除 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持存储过程使用游标功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持存储过程使用游标功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述  2、通过Waterdrop工具，向表tb\_test1中插入几条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，创建存储过程NewProc，存在游标，SQL语句见测试脚本中描述  4、通过Waterdrop工具，调用存储过程NewProc |
| 预期结果 | 1、表tb\_test1可成功创建  2、表tb\_test1数据记录插入成功  3、存储过程NewProc创建成功  4、存储过程调用成功，表tb\_test1中存在被存储过程插入的数据 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test','2020-3-10 14:00:00','name','data',30,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(2,'test','2020-3-10 14:00:00','name','data',30,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(3,'test','2020-3-10 14:00:00','name','data',30,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(4,'test','2020-3-10 14:00:00','name','data',30,100);*  *--创建存储过程NewProc SQL语句*  *delimiter $$*  *create procedure if not exists testcursor()*  *Begin*  *Declare stopflag int default 0;*  *Declare my\_uname varchar(20);*  *Declare uname\_cur cursor for select name from tb\_test1;*  *declare continue handler for not found set stopflag=1;*  *Open uname\_cur;*  *Fetch uname\_cur into my\_uname;*  *while( stopflag=0 ) do*  *begin*  *update tb\_test1 set `data`='cursor update';*  *end ;*  *Fetch uname\_cur into my\_uname;*  *End while;*  *Close uname\_cur;*  *End*  *$$*  *delimiter ;*  *--调用存储过程语句*  *call testcursor();* |
| 实际结果 | 1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、存储过程创建成功，结果如下：    4、存储过程调用成功，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 触发器使用管理

### 验证KunDB支持触发器使用动态SQL和游标功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持触发器使用动态SQL和游标功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1、tb\_test2，SQL语句见测试脚本中描述  2、通过Waterdrop工具，向表tb\_test1中插入几条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，创建触发器newtrigger，存在动态SQL语句和游标，SQL语句见测试脚本中描述  4、通过Waterdrop工具，向表中test\_2中插入数据，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表tb\_test1数据插入成功  3、触发器newtrigger创建成功  4、数据插入成功，同时触发器被调用 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *create table if not exists tb\_test2 like tb\_test1;*  --创建触发器newtrigger SQL语句  *drop trigger if exists newtrigger;*  *delimiter*  *$$*  *create trigger newtrigger before insert*  *on tb\_test2 for each row*  *begin*  *declare name\_test varchar(20);*  *declare test\_cursor cursor for select name from tb\_test1;*  *open test\_cursor;*  *fetch next from test\_cursor into name\_test;*  *update tb\_test1 set data='trigger change';*  *close test\_cursor;*  *END*  *$$*  *delimiter;*  *--向表tb\_test1插入数据SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test1','2020-3-10 14:00:00','name1','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(2,'test2','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(3,'test3','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(4,'test4','2020-3-10 14:00:00','name3','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(5,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *--向表tb\_test2插入数据SQL语句*  *insert into tb\_test2(id,name) values(1,"hello");* |
| 实际结果 | **1. 非图形化操作**  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、触发器创建成功，结果如下：    4、数据插入成功，同时触发器被调用，结果如下：      **2. 图形化操作**    1. 建表成功    2. 插入数据成功    3. 触发器创建成功    4.数据插入成功，触发器被调用 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持修改触发器名称功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持修改触发器名称功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1、tb\_test2，SQL语句见测试脚本中描述  2、通过Waterdrop工具，向表tb\_test1中插入几条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，创建触发器newtrigger，SQL语句见测试脚本中描述  4、通过Waterdrop工具，修改触发器newtrigger为newtrigger1，SQL语句见测试脚本中描述  5、通过Waterdrop工具，向表中test\_2中插入数据，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表tb\_test1数据插入成功  3、触发器newtrigger创建成功  4、触发器的名称修改成功  5、数据插入成功，同时触发器被调用 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *create table if not exists tb\_test2 like tb\_test1;*  --创建触发器newtrigger SQL语句  *drop trigger if exists newtrigger;*  *delimiter*  *$$*  *create trigger newtrigger before insert*  *on tb\_test2 for each row*  *begin*  *declare name\_test varchar(20);*  *declare test\_cursor cursor for select name from tb\_test1;*  *open test\_cursor;*  *fetch next from test\_cursor into name\_test;*  *update tb\_test1 set data='trigger change';*  *close test\_cursor;*  *END*  *$$*  *delimiter;*  --修改触发器的名称  *alter trigger newtrigger rename to newtrigger1;*  *--向表tb\_test1插入数据SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test1','2020-3-10 14:00:00','name1','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(2,'test2','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(3,'test3','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(4,'test4','2020-3-10 14:00:00','name3','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(5,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *--向表tb\_test2插入数据SQL语句*  *insert into tb\_test2(id,name) values(1,"hello");* |
| 实际结果 | 1. 非图形化操作  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、触发器创建成功，结果如下：    4、触发器的名称修改成功，结果如下：      5、数据插入成功，同时触发器被调用，结果如下：      2. 图形化操作  1. 修改触发器名称    2. 插入数据，调用触发器成功 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持修改触发器内容功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持修改触发器名称功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1、tb\_test2，SQL语句见测试脚本中描述  2、通过Waterdrop工具，向表tb\_test1中插入几条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，创建触发器newtrigger，SQL语句见测试脚本中描述  4、通过Waterdrop工具，修改触发器newtrigger的执行语句，SQL语句见测试脚本中描述  5、通过Waterdrop工具，向表中test\_2中插入数据，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表tb\_test1数据插入成功  3、触发器newtrigger创建成功  4、触发器的执行语句修改成功  5、数据插入成功，同时触发器被调用 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *create table if not exists tb\_test2 like tb\_test1;*  --创建触发器newtrigger SQL语句  *drop trigger if exists newtrigger;*  *delimiter*  *$$*  *create trigger newtrigger before insert*  *on tb\_test2 for each row*  *begin*  *declare name\_test varchar(20);*  *declare test\_cursor cursor for select name from tb\_test1;*  *open test\_cursor;*  *fetch next from test\_cursor into name\_test;*  *update tb\_test1 set data='trigger change';*  *close test\_cursor;*  *END*  *$$*  *delimiter;*  *--修改触发器*newtrigger的执行语句  *delimiter*  *$$*  *create or replace trigger newtrigger before insert*  *on tb\_test2 for each row*  *begin*  *declare name\_test varchar(20);*  *declare test\_cursor cursor for select name from tb\_test1;*  *open test\_cursor;*  *fetch next from test\_cursor into name\_test;*  *update tb\_test1 set data='trigger has been changed';*  *close test\_cursor;*  *END*  *$$*  *delimiter;*  *--向表tb\_test1插入数据SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test1','2020-3-10 14:00:00','name1','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(2,'test2','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(3,'test3','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(4,'test4','2020-3-10 14:00:00','name3','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(5,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *--向表tb\_test2插入数据SQL语句*  *insert into tb\_test2(id,name) values(1,"hello");* |
| 实际结果 | **1. 非图形化操作**  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、触发器创建成功，结果如下：    4、触发器的执行语句修改成功，结果如下：      5、数据插入成功，同时触发器被调用，结果如下：    **2. 图形化操作**  1. 修改触发器    2. 插入数据，且触发器调用成功 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持删除触发器功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持删除触发器功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1、tb\_test2，SQL语句见测试脚本中描述  2、通过Waterdrop工具，向表tb\_test1中插入几条记录，SQL语句见测试脚本中描述  3、通过Waterdrop工具，创建触发器newtrigger，SQL语句见测试脚本中描述  4、通过Waterdrop工具，删除触发器newtrigger，SQL语句见测试脚本中描述  5、通过Waterdrop工具，向表中test\_2中插入数据，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表tb\_test1数据插入成功  3、触发器newtrigger创建成功  4、触发器newtrigger删除成功  5、数据插入成功，没有触发器被调用，tb\_test1的数据未被修改 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *create table if not exists tb\_test2 like tb\_test1;*  --删除触发器newtrigger SQL语句  *drop trigger if exists newtrigger;*  --创建触发器newtrigger SQL语句  *delimiter*  *$$*  *create trigger newtrigger before insert*  *on tb\_test2 for each row*  *begin*  *declare name\_test varchar(20);*  *declare test\_cursor cursor for select name from tb\_test1;*  *open test\_cursor;*  *fetch next from test\_cursor into name\_test;*  *update tb\_test1 set data='trigger change';*  *close test\_cursor;*  *END*  *$$*  *delimiter;*  *--向表tb\_test1插入数据SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test1','2020-3-10 14:00:00','name1','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(2,'test2','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(3,'test3','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(4,'test4','2020-3-10 14:00:00','name3','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(5,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *--向表tb\_test2插入数据SQL语句*  *insert into tb\_test2(id,name) values(1,"hello");* |
| 实际结果 | **1. 非图形化操作**  1、表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、触发器创建成功，结果如下：    4、触发器newtrigger删除成功，结果如下：    5、数据插入成功，没有触发器被调用，tb\_test1的数据未被修改，结果如下：      **2. 图形化操作**  1. 删除触发器    2. 插入数据不再调用触发器 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 分片操作

### 验证KunDB支持分片操作功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持分片操作功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接，只连接一个KunGate |
| 测试步骤 | 1. 通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，SQL语句见测试脚本中描述 2. 通过Waterdrop工具，向表tb\_test1中插入几条记录，SQL语句见测试脚本中描述   3、查询表test1的记录 |
| 预期结果 | 1、表tb\_test1创建成功,每个分片上均存在该表信息  2、表tb\_test1数据插入成功，每个分片上均存在部分数据  3、数据查询成功，数据和插入的数据一致 |
| 测试脚本 | ***1.非图形操作***  *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--向表tb\_test1插入数据SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test1','2020-3-10 14:00:00','name1','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(2,'test2','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(3,'test3','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(4,'test4','2020-3-10 14:00:00','name3','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(5,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(6,'test1','2020-3-10 14:00:00','name1','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(7,'test2','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(8,'test3','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(9,'test4','2020-3-10 14:00:00','name3','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(10,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *--查询数据SQL语句*  *select \* from tb\_test1;*  ***2、图形操作***   1. *在所建模式中新建表*      1. *在新建表的下拉列表中的“列”中新建列*      1. *表列建好后，右键单击下拉列表的分区选项，选择新建分区*      1. *选择想要分区的列、hash类型和分区函数，单击确定*   （注：只可在建表且未保存时进行分区操作，未分区的表保存后无法再进行表列分区，只能删除表重新创建；hash类型仅支持单一表列分区，不支持多表列hash分区，分区数与环境shed数一致）     1. *ctrl+s或右下角按钮进行保存，并执行语句*      1. *双击表，在右侧数据属性中插入数据并保存，对分区进行验证*      1. *分区保存完成后右键单击分区，选择查看分区，查看分区属性*      1. *在右侧栏中右键单击分区打开菜单，选择”编辑分区“查看分区属性以及分区数据* |
| 实际结果 | 1. ***非图形操作***   1、表tb\_test1创建成功,每个分片上均存在该表信息，结果如下：        2、表tb\_test1数据插入成功，每个分片上均存在部分数据，结果如下：        3、数据查询成功，数据和插入的数据一致，结果如下：     1. ***图形操作***   1、表数据插入成功，每个分片上均存在部分数据且涵盖全部数据，证明分区成功： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 函数管理

### 验证KunDB支持计算函数功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持计算函数功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创查询的SQL语句，存在计算型函数，SQL语句见测试脚本中描述 |
| 预期结果 | 1、查询执行成功，结果中计算函数的值正确 |
| 测试脚本 | *--计算函数查询语句*  *select 10%3,3\*1,3+1,3-1,-1,3/1,ABS(-1),ATAN(45),CEIL(1.2),CEILING(1.2),CONV('A',16,2);*  *select COS(PI()),COT(12),CRC32('MySQL'),DEGREES(PI()),EXP(2),FLOOR(1.23),HEX(1),LN(2),LOG(2);*  *select LOG2(2),LOG10(10),MOD(234,10),PI(),POW(2,2),RADIANS(90),RAND(),ROUND(-1.58),sign(3),sin(pi());*  *select SQRT(4),TAN(10),TRUNCATE(1.223,1);* |
| 实际结果 | 1、查询执行成功，结果中计算函数的值正确，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持日期函数功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持日期函数功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创查询的SQL语句，存在日期型函数，SQL语句见测试脚本中描述 |
| 预期结果 | 1、查询执行成功，结果中日期函数的值正确 |
| 测试脚本 | *--日期函数查询语句*  *select DATE\_ADD('2008-01-02', INTERVAL 31 DAY),ADDTIME('2007-12-31 23:59:59.999999', '1 1:1:1.000002'),CURDATE(),CURTIME();*  *select DATE('2003-12-31 01:02:03'),DATEDIFF('2007-12-31 23:59:59','2007-12-30'),DATE\_ADD('2018-05-01',INTERVAL 1 DAY),DATE\_SUB('2018-05-01',INTERVAL 1 month);*  *select DATE\_FORMAT('2009-10-04 22:23:00', '%W %M %Y'),DAYNAME('2007-02-03'),DAYOFMONTH('2007-02-03'),DAYOFWEEK('2007-02-03'),DAYOFWEEK('2007-02-03'),DAYOFWEEK('2007-02-03');*  *select FROM\_DAYS(730669),FROM\_UNIXTIME(1447430881),FROM\_UNIXTIME(1447430881,'%Y %D %M %h:%i:%s %x'),STR\_TO\_DATE('10.31.2003','%m.%d.%Y'),HOUR('10:05:03'),LAST\_DAY('2003-02-05');*  *select MAKEDATE(2011,31),MAKETIME(12,15,30),MICROSECOND('12:00:00.123456'),MINUTE('2008-02-03 10:05:03'),MONTH('2008-02-03'),MONTHNAME('2008-02-03'),NOW(),SLEEP(2),SYSDATE();*  *select PERIOD\_ADD(200801,2),PERIOD\_DIFF(200802,200703),QUARTER('2008-04-01'),SECOND('10:05:03'),SEC\_TO\_TIME(2378),STR\_TO\_DATE('01,5,2013','%d,%m,%Y'),STR\_TO\_DATE('a09:30:17','a%h:%i:%s');*  *select STR\_TO\_DATE('00/00/0000', '%m/%d/%Y'),SUBDATE('2008-01-02', INTERVAL 31 DAY),SUBTIME('2007-12-31 23:59:59.999999','1 1:1:1.000002'),TIME('2003-12-31 01:02:03');*  *select TIMEDIFF('2000:01:01 00:00:00','2000:01:01 00:00:00.000001'),TIMESTAMP('2003-12-31'),TIMESTAMPADD(MINUTE,1,'2003-01-02'),TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01');*  *select TIME\_FORMAT('100:00:00', '%H %k %h %I %l'),TIME\_TO\_SEC('22:23:00'),TO\_DAYS(950501),TO\_SECONDS(950501),UNIX\_TIMESTAMP(),UTC\_DATE(),UTC\_TIME(),UTC\_TIMESTAMP(),WEEK('2008-02-20');*  *select YEAR('2000-01-01'),YEARWEEK('2000-01-01'),WEEKDAY('2008-02-03 22:23:00'),WEEKOFYEAR('2008-02-20'),YEAR('1987-01-01'),YEARWEEK('1987-01-01');* |
| 实际结果 | 1、查询执行成功，结果中计算函数的值正确，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持字符函数功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持字符函数功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创查询的SQL语句，存在字符型函数，SQL语句见测试脚本中描述 |
| 预期结果 | 1、查询执行成功，结果中字符函数的值正确 |
| 测试脚本 | *--字符函数查询语句*  *--正则*  *SELECT 'Michael!' REGEXP '.\*','a' REGEXP '^[a-d]','fo\nfo' REGEXP '^fo$','fofo' REGEXP '^fo','fo\no' REGEXP '^fo\no$','fofo' REGEXP '^f.\*$','Baaaaaan' REGEXP '^Ba+n','Bn' REGEXP '^Ba?n','pi' REGEXP 'pi|apa','pi' REGEXP '^(pi)\*$',*  *'abcde' REGEXP 'a[bcd]{2}e','aXbc' REGEXP '[a-dXYZ]','justalnums' REGEXP '[[:alnum:]]+','a word a' REGEXP '[[:<:]]word[[:>:]]','1+2' REGEXP '1+2';*  *SELECT ASCII("2"),BIN(12),BIT\_LENGTH('TEXT'),CHAR(77,121,83,81,'76'),CHAR\_LENGTH('KUNDB'),CONCAT("Kun","DB"),CONCAT\_WS(',','First name','Second name','Last Name'),ELT(1, 'Aa', 'Bb', 'Cc', 'Dd'),EXPORT\_SET(5,'Y','N',',',4),*  *FIELD('Bb', 'Aa', 'Bb', 'Cc', 'Dd', 'Ff'),FIND\_IN\_SET('b','a,b,c,d'),TO\_BASE64('abc'), FROM\_BASE64(TO\_BASE64('abc')),HEX('abc'),INSTR('foobarbar', 'bar'),LCASE("KUNDB"),LEFT('foobarbar', 5),*  *LENGTH('text'),LOWER('QUADRATICALLY'),LPAD('hi',4,'??'),LTRIM(' barbar'),MAKE\_SET(1,'a','b','c'),OCT(12),ORD('2'),REPEAT('MySQL', 3),REPLACE('www.mysql.com', 'w', 'Ww'),*  *REVERSE('abc'),RIGHT('foobarbar', 4),RPAD('hi',5,'?'),RTRIM('barbar '),SOUNDEX('Hello'),SPACE(6),SUBSTRING('Quadratically',5),SUBSTRING\_INDEX('www.mysql.com', '.', 2),TRIM(' bar '),UNHEX('4D7953514C'),*  *UNHEX('GG'),UPPER('Hej'),WEIGHT\_STRING('asdf'),'a' LIKE 'a ','bar' NOT LIKE '%baz%',STRCMP('text', 'text2'),CHARSET(BINARY 'a'), COLLATION(BINARY 'a'),USER(), CHARSET(USER()), COLLATION(USER()),QUOTE('Don\'t!');* |
| 实际结果 | 1、查询执行成功，结果中字符函数的值正确，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 字符集编码

### 验证KunDB支持GBK字符集功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持GBK字符集功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1设置字符集编码为GBK,具体设置方面见测试脚本  2、通过Waterdrop工具，并向表tb\_test1中插入几条记录，存在中文字符，SQL语句见测试脚本中描述  3、查询表test1的记录，查看中文数据信息 |
| 预期结果 | 1、GBK字符集编码的表创建成功  2、表tb\_test1创建并数据插入成功  3、数据查询成功，中文显示正确 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=gbk PARTITION BY HASH(id) USING `binary\_md5`;*  *--向表tb\_test1插入数据SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test1','2020-3-10 14:00:00','张三','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(2,'test2','2020-3-10 14:00:00','李四','data',40,100);*  *--查询数据SQL语句*  *select \* from tb\_test1;* |
| 实际结果 | 1、GBK字符集编码的表创建成功，结果如下：    2、表记录插入成功，结果如下：    3、数据查询成功，中文显示正确，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持GB18030字符集功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持GB 18030字符集功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1设置字符集编码为GB18030,具体设置方面见测试脚本  2、通过Waterdrop工具，并向表tb\_test1中插入几条记录，存在中文字符，SQL语句见测试脚本中描述  3、查询表test1的记录，查看中文数据信息 |
| 预期结果 | 1、GB18030字符集编码的表创建成功  2、表tb\_test1创建并数据插入成功  3、数据查询成功，中文显示正确 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=*GB18030 *PARTITION BY HASH(id) USING `binary\_md5`;*  *--向表tb\_test1插入数据SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test1','2020-3-10 14:00:00','张三','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(2,'test2','2020-3-10 14:00:00','李四','data',40,100);*  *--查询数据SQL语句*  *select \* from tb\_test1;* |
| 实际结果 | 1、GB18030字符集编码修改成功，结果如下：    2、表创建成功，记录插入成功，结果如下：      3、数据查询成功，中文显示正确，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 分区表管理

* + - 1. **验证KunDB支持HASH分区表功能**

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持HASH分区表功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1. 通过Waterdrop工具，在数据库dbtest1中创建HASH分区表tb\_test1，SQL语句见测试脚本中描述 2. 通过Waterdrop工具，向表tb\_test1中插入几条记录，SQL语句见测试脚本中描述 3. 通过Waterdrop工具，查询表tb\_test1的记录，以及各个分区中的数据分布 4. 通过Waterdrop工具，单独查询第一个分区的数据 5. 删除HASH分区表tb\_test1 |
| 预期结果 | 1、表tb\_test1创建成功,每个分片上均存在该表信息  2、表tb\_test1数据插入成功  3、数据查询成功，每个分区中的数据量总数和插入的数据一致  4、数据查询成功，只显示第一个分区的数据  5、表删除成功 |
| 测试脚本 | 1. ***非图形操作***   *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--向表tb\_test1插入数据SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test1','2020-3-10 14:00:00','name1','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(2,'test2','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(3,'test3','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(4,'test4','2020-3-10 14:00:00','name3','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(5,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(6,'test1','2020-3-10 14:00:00','name1','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(7,'test2','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(8,'test3','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(9,'test4','2020-3-10 14:00:00','name3','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(10,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *--查询数据SQL语句和分区数据*  *select \* from tb\_test1;*  *show partitions from tb\_test1;*  *--查询某个分区的数据*  *select \* from tb\_test1 partition(p0);*  *--删除表SQL语句*  *drop table if exists tb\_test1;*   1. ***图形操作***   *1.在所建模式中新建表*    *2.在新建表的下拉列表中的“列”中新建列*    *3.表列建好后，右键单击下拉列表的分区选项，选择新建分区*    *4.选择想要分区的列、hash类型和分区函数，单击确定*  （注：只可在建表且未保存时进行分区操作，未分区的表保存后无法再进行表列分区，只能删除表重新创建；hash类型仅支持单一表列分区，不支持多表列hash分区，分区数量与环境shed数一致）    *5 .ctrl+s或右下角按钮进行保存，并执行语句*    *6. 双击表，在右侧数据属性中插入数据并保存，对分区进行验证*    *7.分区保存完成后右键单击分区，选择查看分区，查看分区属性*    *8.在右侧栏中右键单击分区打开菜单，选择”编辑分区“查看分区属性以及分区数据*    *9. 在右侧栏中右键单击分区打开菜单，选择“删除“进行删除分区，ctrl+s或按钮进行保存* |
| 实际结果 | ***1.非图形操作：***  1、表tb\_test1创建成功,每个分片上均存在该表信息，结果如下：    2、表tb\_test1数据插入成功，结果如下：    3、数据查询成功，每个分区中的数据量总数和插入的数据一致：      4、数据查询成功，只显示第一个分区的数据    5、表删除成功，结果如下：    ***2.图形操作：***  1.保存后目录树分区中显示生成的分区：    2.进行查看分区操作后右侧显示分区：    3.右侧进行编辑分区操作后，成功显示分区属性和分区数据：    4.分区数据显示无误，证明分区有效    5.删除后右侧目录树分区不再显示，分区删除成功： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

* + - 1. **验证KunDB支持RANGE分区表功能**

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持RANGE分区表功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1. 通过Waterdrop工具，在数据库dbtest1中创建HASH分区表tb\_test1，SQL语句见测试脚本中描述 2. 通过Waterdrop工具，向表tb\_test1中插入几条记录，SQL语句见测试脚本中描述 3. 通过Waterdrop工具，查询表tb\_test1的记录，以及各个分区中的数据分布 4. 通过Waterdrop工具，单独查询第一个分区的数据 5. 删除RANGE分区表tb\_test1 |
| 预期结果 | 1、表tb\_test1创建成功,每个分片上均存在该表信息  2、表tb\_test1数据插入成功  3、数据查询成功，每个分区中的数据量总数和插入的数据一致  4、数据查询成功，只显示第一个分区的数据  5、表删除成功 |
| 测试脚本 | 1. ***非图形操作***   *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY RANGE(id)(*  *PARTITION p0 VALUES LESS THAN (6),*  *PARTITION p1 VALUES LESS THAN (MAXVALUE));*  *--向表tb\_test1插入数据SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test1','2020-3-10 14:00:00','name1','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(2,'test2','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(3,'test3','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(4,'test4','2020-3-10 14:00:00','name3','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(5,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(6,'test1','2020-3-10 14:00:00','name1','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(7,'test2','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(8,'test3','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(9,'test4','2020-3-10 14:00:00','name3','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(10,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *--查询数据SQL语句和分区数据*  *select \* from tb\_test1;*  *show partitions from tb\_test1;*  *--查询某个分区的数据*  *select \* from tb\_test1 partition(p0);*  *--删除表SQL语句*  *drop table if exists tb\_test1;*  ***2、图形操作***   1. *在所建模式中新建表*      1. *在新建表的下拉列表中的“列”中新建列*      1. *表列建好后，右键单击下拉列表的分区选项，选择新建分区*      1. *选择想要分区的列、RANGE类型和分区函数*   （注：只可在建表且未保存时进行分区操作，未分区的表保存后无法再进行表列分区，只能删除表重新创建）     1. *在划分详情处选择想要划分的数目，并点击生成划分*      1. *根据需要设置分区名称和表达式，表达式形如10表示为小于10；设置完后单击确定*   （注：划分数目大于1时每一个分区都需要设置表达式）     1. *ctrl+s或右下角按钮进行保存，并执行语句*      1. *双击表，在右侧数据属性中插入数据并保存，对分区进行验证*      1. *分区保存完成后右键单击分区，选择查看分区，查看分区属性*      1. *在右侧栏中右键单击分区打开菜单，选择”编辑分区“查看属性和分区数据*     *11.在右侧栏中右键单击分区打开菜单，选择“删除“进行删除分区，ctrl+s或按钮进行保存* |
| 实际结果 | 1. ***非图形操作：***   1、表tb\_test1创建成功,每个分片上均存在该表信息，结果如下：    2、表tb\_test1数据插入成功，结果如下：    3、数据查询成功，每个分区中的数据量总数和插入的数据一致，结果如下：      4、数据查询成功，只显示第一个分区的数据，结果如下：    5、表删除成功，结果如下：    ***2.图形操作：***  1、保存后目录树分区中显示生成的分区：    2、进行查看分区操作后右侧显示分区：    3、右侧进行编辑分区操作后，成功显示分区属性和分区数据：    4、分区数据显示成功，证明分区有效    5.、删除后右侧目录树分区不再显示，分区删除成功： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

* + - 1. **验证KunDB支持LIST分区表功能**

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持LIST分区表功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1. 通过Waterdrop工具，在数据库dbtest1中创建HASH分区表tb\_test1，SQL语句见测试脚本中描述 2. 通过Waterdrop工具，向表tb\_test1中插入几条记录，SQL语句见测试脚本中描述 3. 通过Waterdrop工具，查询表tb\_test1的记录，以及各个分区中的数据分布 4. 通过Waterdrop工具，单独查询第一个分区的数据 5. 删除RANGE分区表tb\_test1 |
| 预期结果 | 1、表tb\_test1创建成功,每个分片上均存在该表信息  2、表tb\_test1数据插入成功  3、数据查询成功，每个分区中的数据量总数和插入的数据一致  4、数据查询成功，只显示第一个分区的数据  5、表删除成功 |
| 测试脚本 | 1. ***非图形操作***   *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY LIST(id)(*  *PARTITION p0 VALUES(1,2,3,4,5),*  *PARTITION p1 VALUES(6,7,8,9,10),*  *PARTITION p2 VALUES(default));*  *--向表tb\_test1插入数据SQL语句*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(1,'test1','2020-3-10 14:00:00','name1','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(2,'test2','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(3,'test3','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(4,'test4','2020-3-10 14:00:00','name3','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(5,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(6,'test1','2020-3-10 14:00:00','name1','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(7,'test2','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(8,'test3','2020-3-10 14:00:00','name2','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(9,'test4','2020-3-10 14:00:00','name3','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(10,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(11,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *insert into tb\_test1(`id`,`zjhm`,`create\_time`,`name`,`data`,`age`,`price`) values(100,'test5','2020-3-10 14:00:00','name4','data',40,100);*  *--查询数据SQL语句和分区数据*  *select \* from tb\_test1;*  *show partitions from tb\_test1;*  *--查询某个分区的数据*  *select \* from tb\_test1 partition(p0);*  *--删除表SQL语句*  *drop table if exists tb\_test1;*  ***2、图形操作***  *1.在所建模式中新建表*    *2.在新建表的下拉列表中的“列”中新建列*    *3. 表列建好后，右键单击下拉列表的分区选项，选择新建分区*    *4. 选择想要分区的列、RANGE类型和分区函数*  （注：只可在建表且未保存时进行分区操作，未分区的表保存后无法再进行表列分区，只能删除表重新创建）    *5. 在划分详情处选择想要划分的数目，并点击生成划分*    *6. 根据需要设置分区名称和表达式，表达式形如10表示为LIST为10；设置完后单击确定*  （注：划分数目大于1时每一个分区都需要设置表达式）    *7. ctrl+s或右下角按钮进行保存，并执行语句*    *8. 双击表，在右侧数据属性中插入数据并保存，对分区进行验证*    *9. 分区保存完成后右键单击分区，选择查看分区，查看分区属性*    *10. 在右侧栏中右键单击分区打开菜单，选择”编辑分区“查看属性和分区数据*    *11. 在右侧栏中右键单击分区打开菜单，选择“删除“进行删除分区，ctrl+s或按钮进行保存* |
| 实际结果 | ***1.非图形操作***  1、表tb\_test1创建成功,每个分片上均存在该表信息，结果如下：    2、表tb\_test1数据插入成功，结果如下：     1. 数据查询成功，每个分区中的数据量总数和插入的数据一致，结果如下：       4、数据查询成功，只显示第一个分区的数据，结果如下：    5、表删除成功，结果如下：    ***2.图形操作：***  1、保存后目录树分区中显示生成的分区：    2、进行查看分区操作后右侧显示分区：    3、右侧进行编辑分区操作后，成功显示分区属性和分区数据：    4、分区数据显示成功，证明分区有效    5.、删除后右侧目录树分区不再显示，分区删除成功： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 分布式事务管理

### 验证KunDB支持分布式事务全部回滚原子性功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持分布式事务全部回滚原子性功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1,存在用户vt\_app均具备所有权限 |
| 测试步骤 | 1、通过vt\_app用户，在数据库dbtest1中创建表test\_11，SQL语句见测试脚本中描述  2、通过vt\_app用户，执行事务全部回滚操作，SQL语句见测试脚本中描述  3、通过vt\_app用户，执行查询事务操作，SQL语句见测试脚本中描述 |
| 预期结果 | 1. 表test\_11创建成功 2. 事务回滚操作执行成功 3. 查询事务执行成功，表test\_11中不存在任何数据 |
| 测试脚本 | *--建表SQL语句*  *create table test\_11 (c int primary key,d int);*  *--事务回滚操作*  *begin; --采用隐式事务机制的数据库，可以不用执行本语句*  *Insert into test\_11 values(1,1); --执行成功*  *Insert into test\_11 values(2,2); --执行成功*  *Rollback;--或rollback，根据各厂商语法执行，预期执行成功*  *--事务查询操作*  *Begin; --采用隐式事务启动的数据库，可以不用执行本语句*  *select \* from test\_11 ; --执行成功，返回空结果集*  *commit; --结束查询事务* |
| 实际结果 | 1. 表test\_11创建成功，结果如下：      1. 事务回滚操作执行成功，结果如下：      1. 查询事务执行成功，表test\_11中不存在任何数据，结果如下： |
| 备注 |  |

### 验证KunDB支持分布式事务全部提交原子性功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持分布式事务全部提交原子性功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1,存在用户vt\_app均具备所有权限  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过vt\_app用户，在数据库dbtest1中创建表test\_11，SQL语句见测试脚本中描述  2、通过vt\_app用户，执行事务全部提交操作，SQL语句见测试脚本中描述  3、通过vt\_app用户，执行查询事务操作，SQL语句见测试脚本中描述 |
| 预期结果 | 1. 表test\_11创建成功 2. 事务回滚操作执行成功 3. 查询事务执行成功，表test\_11中存在提交的数据 |
| 测试脚本 | *--建表SQL语句*  *create table test\_11 (c int primary key,d int);*  *--事务提交操作*  *Begin; --采用隐式事务机制的数据库，可以不用执行本语句*  *Insert into test\_11 values(1,1); --执行成功*  *Insert into test\_11 values(2,2); --执行成功*  *commit;--或commit，根据各厂商语法执行，预期执行成功*  *--事务查询操作*  *Begin; --采用隐式事务启动的数据库，可以不用执行本语句*  *select \* from test\_11 ; --执行成功，返回空结果集*  *commit; --结束查询事务* |
| 实际结果 | 1. 表test\_11创建成功，结果如下：      1. 事务提交操作执行成功，结果如下：      1. 查询事务执行成功，表test\_11中不存提交的数据，结果如下： |
| 备注 |  |

### 验证KunDB支持分布式事务一致性功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持分布式事务一致性功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1，存在三个用户分别为vt\_app,u1,u2,均具备所有权限 |
| 测试步骤 | 1、通过vt\_app用户，在数据库dbtest1中创建表test\_1，SQL语句见测试脚本中描述  2、通过u1用户，向表test\_1执行插入事务操作，通过u2用户，在u1用户插入事务提交前后，分别执行查询操作，SQL语句见测试脚本中描述 |
| 预期结果 | 1. 表test\_11创建成功 2. u1用户插入事务操作执行成功，u2 用户在u1插入事务提交前，查询不到数据，提交后可查询到提交的插入数据 |
| 测试脚本 | *--建表SQL语句*  *Create table test\_1(a int not null, b int, c char(8));*  *--一致性事务操作*   | *u1用户* | *u2用户* | | --- | --- | | *SQL>Begin work;*  *SQL>Insert into test\_1 values(1,11,'test1'),(2,12,'test2');* |  | |  | *SQL>select \* from test\_1;*    *---返回0行* | | *SQL>commit;*    *--- SQL operation complete.* |  | |  | *SQL>select \* from test\_1;*    *A           B           C*  *----------- ----------- --------*  *2          12 test2*  *1          11 test1*    *--- 查询出两条记录* | |
| 实际结果 | 1. 表test\_11创建成功，结果如下：      1. u1用户插入事务操作执行成功，u2 用户在u1插入事务提交前，查询不到数据，提交后可查询到提交的插入数据，结果如下：   IMG_256 |
| 备注 |  |

### 验证KunDB支持分布式事务隔离性功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持分布式事务隔离性功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过vt\_app用户，在数据库dbtest1中创建表test\_1，SQL语句见测试脚本中描述  2、通过u1、u2用户，向表test\_1执行插入事务操作，SQL语句见测试脚本中描述  3、查询隔离级别并记录 |
| 预期结果 | 1. 表test\_11创建成功 2. u1、u2不能查询到其他用户未提交的数据 |
| 测试脚本 | *--建表SQL语句*  *Create table test\_1(a int not null, b int, c char(8));*  *--事务隔离性操作*   |  |  | | --- | --- | | *u1用户* | *u2用户* | | *SQL>begin work;* | *SQL>begin work;* | | *SQL>insert into test\_1 values(3,13,'test3');*    *---插入一条语句* | *SQL>insert into test\_1 values(3,33,'test333');*    *---插入一条语句* | | *SQL>select \* from test\_1 where a=3;*  *A           B           C*  *----------- ----------- --------*  *3          13 test3*    *---查询出一条数据*  *---无法查询出用户2插入的数据* | *SQL>select \* from test\_1 where a=3;*  *A           B           C*  *----------- ----------- --------*  *3          33 test333*    *---查询出一条数据*  *---无法查询出用户1插入的数据* | | *SQL>commit;* | *SQL>commit work;* | | *SQL>select \* from test\_1;*    *A           B           C*  *----------- ----------- --------*  *2          12 test2*  *1          11 test1*  *3          13 test3*  *3          33 test333* | *SQL>select \* from test\_1;*    *A           B           C*  *----------- ----------- --------*  *2          12 test2*  *1          11 test1*  *3          13 test3*  *3          33 test333* | |
| 实际结果 | 1、表test\_11创建成功，结果如下：    2、u1、u2不能查询到其他用户未提交的数据，结果如下：  IMG_257  3、隔离级别如下： |
| 备注 |  |

### 验证KunDB支持分布式事务持久性功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持分布式事务持久性功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1,存在用户vt\_app均具备所有权限  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过vt\_app用户，在数据库dbtest1中创建表test\_11，SQL语句见测试脚本中描述  2、通过vt\_app用户，执行事务全部提交操作，SQL语句见测试脚本中描述  3、退出vt\_app用户再重新登录，执行查询操作，SQL语句见测试脚本中描述 |
| 预期结果 | 1. 表test\_11创建成功 2. 事务提交操作执行成功 3. 查询执行成功，表test\_11中存在提交的数据 |
| 测试脚本 | *--建表SQL语句*  *create table test\_11 (c int primary key,d int);*  *--事务提交操作*  *Begin; --采用隐式事务机制的数据库，可以不用执行本语句*  *Insert into test\_11 values(1,1); --执行成功*  *Insert into test\_11 values(2,2); --执行成功*  *commit;--或commit，根据各厂商语法执行，预期执行成功*  *--查询SQL操作*  *select \* from test\_11 ;* |
| 实际结果 | 1、表test\_11创建成功，结果如下：  2、事务提交操作执行成功，结果如下：     1. 查询执行成功，表test\_11中存提交的数据，结果如下： |
| 备注 |  |

### 验证KunDB支持分布式事务原子性功能

|  |  |  |
| --- | --- | --- |
| 测试目的 | 验证KunDB支持分布式事务原子性功能 |  |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1,具有3个shard，且无副本，存在用户vt\_app均具备所有权限  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |  |
| 测试步骤 | 1、通过vt\_app用户，在数据库dbtest1中创建表test\_11，SQL语句见测试脚本中描述  2、通过vt\_app用户，在数据库dbtest1中创建存储过程*trans\_atomicity()*，用于执行批量向表test\_11插入1万条数据的事务，SQL语句见测试脚本中描述  3、通过Waterdrop工具，使用vt\_app用户，采用loadbalance的方式，连接3个KunGate，执行调用存储过程*trans\_atomicity()的操作*，SQL语句见测试脚本中描述  4、在存储过程执行过程中，拔掉一个节点的网线 |  |
| 预期结果 | 1. 表test\_11创建成功 2. 存储过程创建成功 3. 存储过程调用成功 4. 拔掉网线后，存储过程中事务执行失败，所有数据均不提交 |  |
| 测试脚本 | *--建表SQL语句*  *create table test\_11 (c int primary key,d int) partition by hash(c) USING `binary\_md5`;*  *--批量插入1万条数据的事务*  *--通过使用JDBC执行存储过程来模拟*  *create procedure trans\_atomicity()*  *begin*  *declare i int default 0;*  *while i < 10000 do*  *insert into test\_11 values(i,i);*  *set i = i + 1;*  *end while;*  *end;*  *--查询SQL操作*  *select \* from test\_11 ;*  *--调用存储过程*  **call** trans\_atomicity(); |  |
| 实际结果 | 1、表test\_11创建成功，结果如下：    2、存储过程创建成功，结果如下：     1. 存储过程调用成功，结果如下：      1. 拔掉网线后，存储过程中事务执行失败，所有数据均不提交，结果如下： |  |
| 备注 |  |  |

### 验证KunDB支持分布式事务独立性功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持分布式事务独立性功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1,具有2个shard，存在用户vt\_app均具备所有权限  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、在一个节点上建立一个会话1，在会话1中设置事务隔离级别为读提交，执行建表t1,和插入数据的SQL操作，SQL语句见测试脚本中描述  2、在另一个节点上新开一个会话2，查询表t1的数据，SQL语句见测试脚本中描述  3、在原来的查询会话1中，进行commit  4、在会话2中再次查询表t1的数据  5、新建一个查询会话3，在会话3中设置事务隔离级别为读未提交，并执行建表t2及插入数据的SQL语句，SQL语句见测试脚本中描述  6、在另一个节点新建一个查询会话窗口 4,在会话 4 中执行查询表t2数据的SQL语句，SQL语句见测试脚本中描述  7、新建一个查询会话5，在会话5中设置事务隔离级别为读可序列化，并执行建表t3的SQL语句，并执行查询数据事务，SQL语句见测试脚本中描述  8、在另一个节点新建一个查询会话窗口6,在会话 6中插入表3数据的SQL语句，SQL语句见测试脚本中描述  9、在会话5中执行commit操作  10、新建一个查询会话7，在会话7中设置事务隔离级别为可重复读，并执行建表t4和插入数据的SQL语句，SQL语句见测试脚本中描述  11、在另一个节点新建一个查询会话窗口8,在会话8中设置事务隔离级别为可重复读，在会话 8中查询表4数据的SQL语句，SQL语句见测试脚本中描述  12、通过会话7，更新表4中插入的数据记录，SQL语句见测试脚本中描述  13、通过会话8，再次执行插入表4数据的SQL语句，SQL语句见测试脚本中描述  14、通过会话8进行commit,再次执行查询 |
| 预期结果 | 1. 建表t1和插入数据的SQL操作执行成功 2. 查询返回结果集为空 3. Commit成功 4. 再次查询t1数据，返回结果集数据为会话1提交的数据 5. 建表t2和插入数据的SQL操作执行成功 6. 查询t2数据，返回结果集数据为会话3插入但未交的数据 7. 建表t3的SQL操作执行成功 8. 表t3中插入数据的操作被阻塞 9. 会话5的commit操作执行成功，且会话6中插入操作也执行成功 10. 建表4和数据插入成功 11. 会话8查询数据成功，返回正确的结果集 12. 数据更新成功 13. 会话8查询数据成功，依然是更新前的数据 14. Commit成功，会话8查询返回结果集为更新后的结果 |
| 测试脚本 | *--建表t1和插入数据SQL语句*  *create table t1 (c1 int,c2 varchar(10)) partition by hash(c1) USING `binary\_md5`;*  *insert into t1 values(1,'aa');*  *--建表t2和插入数据SQL语句*  *create table t2 (c1 int,c2 varchar(10)) partition by hash(c1) USING `binary\_md5`;*  *insert into t2 values(1,'test1');*  *--建表t3和插入数据SQL语句*  *create table t3 (c1 int,c2 varchar(10)) partition by hash(c1) USING `binary\_md5`;*  *insert into t3 values(1,'test3aaa');*  *--建表t4和插入数据、更新数据的SQL语句*  *create table t4 (c1 int,c2 varchar(10)) partition by hash(c1) USING `binary\_md5`;*  *insert into t4 values(1,'aa');*  *update t4 set c2= 'bb' where c1=1;*  *--设置事务隔离级别为读未提交*  *SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;*  *--设置事务隔离级别为读提交*  *SET TRANSACTION ISOLATION LEVEL READ COMMITTED;*  *--设置事务隔离级别为读可串行化*  *SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;*  *--设置事务隔离级别为可重复读*  *SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;*  *--查询表t1的数据*  *select \* from t1;*  *--查询表t2的数据*  *select \* from t2;*  *--查询表t2的数据*  *select \* from t3;* |
| 实际结果 | 1、建表t1和插入数据的SQL操作执行成功，结果如下：    2、查询返回结果集为空，结果如下：    3、Commit成功，结果如下：    4、再次查询t1数据，返回结果集数据为会话1提交的数据，结果如下：    5、建表t2和插入数据的SQL操作执行成功，结果如下：    6、查询t2数据，返回结果集数据为会话3插入但未交的数据，结果如下：    7、建表t3的SQL操作执行成功，结果如下：    8、表t3中插入数据操作被阻塞，结果如下：    9、会话5的commit操作执行成功，且会话6中插入操作也执行成功，结果如下：      10、建表4和数据插入成功，结果如下：    11、会话8查询数据成功，返回正确的结果集,结果如下：    12、数据更新成功，结果如下：    13、会话8查询数据成功，依然是更新前的数据,结果如下：    14、commit成功，会话8查询返回结果集为更新后的结果,结果如下： |
| 备注 |  |

## 索引

### 验证KunDB支持主键索引功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持主键索引功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中存在主键，SQL语句见测试脚本中描述  2、通过Waterdrop工具，查询表tb\_test1中索引信息，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建。  2、表tb\_test1的索引信息可被正常查询。 |
| 测试脚本 | 1. ***非图形操作***   *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--查询索引SQL语句*  *show index from tb\_test1;*   1. ***图形操作***   *1.右键点击“新建 表”*    *2.展开所建表，在“列”分栏下面右键点击“新建列”*    3.*根据需要设置列名和字段类型*      *4.使用 ctrl + s 或按钮进行保存，并执行语句*    *5.在目录树右键单击约束图标，选择新建约束*    *6.根据需要设置主键索引名称并选取表列*    *7.执行保存操作* |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表tb\_test1的索引信息可被正常查询，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述   1. ***图形操作***   1、左侧目录树约束栏显示所创建的主键PRIMARY    2、左键双击所建主键，右侧属性栏显示主键索引属性，表明创建成功 |

### 验证KunDB支持删除主键索引功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持删除主键索引功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中存在主键，SQL语句见测试脚本中描述  2、通过Waterdrop工具，查询表tb\_test1中索引信息，SQL语句见测试脚本中描述  3、通过Waterdrop工具，删除表tb\_test1中索引信息，SQL语句见测试脚本中描述  4、通过Waterdrop工具，查询表tb\_test1中索引信息，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表tb\_test1的索引信息可被正常查询  3、表tb\_test1可成功删除  4、表tb\_test1的索引信息查询不到 |
| 测试脚本 | ***1、非图形操作***  *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--查询索引SQL语句*  *show index from tb\_test1;*  *--删除表主键约束SQL语句*  *alter table tb\_test1 drop primary key;*  ***2、图形操作***  *1.右键点击“新建 表”*    *2.展开所建表，在“列”分栏下面右键点击“新建列”*    3.*根据需要设置列名和字段类型*      *4.使用 ctrl + s 或按钮进行保存，并执行语句*    *5.在目录树右键单击约束图标，选择新建约束*    *6.根据需要设置主键索引名称并选取表列*    *7.执行保存操作*    *8.右键单击目录树约束栏下的主键，选择”删除“*    9.ctrl +s 或按钮保存，并执行语句 |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表tb\_test1的索引信息可被正常查询，结果如下：    3、表tb\_test1可成功删除，结果如下：    4、表tb\_test1的索引信息查询不到，结果如下：     1. ***图形操作***   1、左侧目录树约束栏显示所创建的主键PRIMARY    2、左键双击所建主键，右侧属性栏显示主键索引属性，表明创建成功    3、删除操作并保存后，右侧目录树约束处不再出现PRIMARY主键索引 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持普通索引功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持普通索引功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1. 通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中存在主键，SQL语句见测试脚本中描述 2. 通过Waterdrop工具，在数据库dbtest1中执行创建普通索引*idx\_common*的SQL语句，SQL语句见测试脚本中描述   3、通过Waterdrop工具，查询表tb\_test1中索引信息，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表tb\_test1的普通索引*idx\_common*被正常创建成功  3、表tb\_test1的索引信息可被正常查询，包括主键索引和普通索引 |
| 测试脚本 | ***1、非图形操作***  *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--查询索引SQL语句*  *show index from tb\_test1;*  *--创建普通索引SQL语句*  *CREATE INDEX idx\_common ON tb\_test1 (name);*  ***2、图形操作***  *1.右键点击“新建 表”*    *2.展开所建表，在“列”分栏下面右键点击“新建列”*    3.*根据需要设置列名和字段类型*      *4.使用 ctrl + s 或按钮进行保存，并执行语句*    *5. 目录树右键单击进行新建索引*    *6.根据需要选择索引列和普通索引的类型、*    *7.ctrl+s进行保存* |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表tb\_test1的普通索引*idx\_common*被正常创建成功，结果如下：     1. 表tb\_test1的索引信息可被正常查询，包括主键索引和普通索引，结果如下：      1. ***图形操作***   1、表newtable的普通索引*newtable\_idx1*被正常创建成功，目录树右侧属性栏显示结果如下：    2、右侧属性栏显示结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持多健值索引功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持唯一索引功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表*index\_test*，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表*index\_test*中创建多健值唯一索引*unique\_key*，SQL语句见测试脚本中描述  3、通过Waterdrop工具，查询表*index\_test*中索引信息，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表*index\_test*可成功创建。  2、表*index\_test*的多健值唯一索引*unique\_key创建成功*  3、表*index\_test*的索引信息可被正常查询，存在创建的多健值唯一索引 |
| 测试脚本 | ***1、非图形操作***  *--建表SQL语句*  *drop table if exists index\_test;*  *create table if not exists index\_test(id int,name varchar(20),age int,address varchar(20)) partition by hash(name) USING `binary\_md5`;*  *--添加唯一索引SQL语句*  *create unique index unique\_key on index\_test(id,name);*  *--查询索引SQL语句*  *show index from index\_test;*  ***2、图形操作***  *1.右键点击“新建 表”*    *2.展开所建表，在“列”分栏下面右键点击“新建列”*    3.*根据需要设置列名和字段类型*      *4.使用 ctrl + s 或按钮进行保存，并执行语句*    *5. 目录树右键单击约束，进行新建约束*    *6.类型选择唯一索引，并选中多个表列*    7.ctrl+s或按钮保存，并执行语句 |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表*index\_test*的多健值唯一索引*unique\_key创建成功*，结果如下：    3、表*index\_test*的索引信息可被正常查询，存在创建的多健值唯一索引，结果如下：     1. ***图形操作***   1、左侧目录树索引栏显示所创建的多键值唯一索引newtable\_un  &pfm111&  2、左键双击所建主键，右侧属性栏显示多键值唯一索引属性，表明创建成功 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持删除普通索引功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持删除普通索引功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1. 通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中存在主键，SQL语句见测试脚本中描述 2. 通过Waterdrop工具，在数据库dbtest1中执行创建普通索引*idx\_common*的SQL语句，SQL语句见测试脚本中描述 3. 通过Waterdrop工具，查询表tb\_test1中索引信息，SQL语句见测试脚本中描述 4. 通过Waterdrop工具，删除数据库dbtest1中普通索引*idx\_common*，SQL语句，SQL语句见测试脚本中描述 5. 通过Waterdrop工具，查询表tb\_test1中索引信息，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表tb\_test1可成功创建  2、表tb\_test1的普通索引*idx\_common*被正常创建成功  3、表tb\_test1的索引信息可被正常查询，包括主键索引和普通索引  4、表tb\_test1的普通索引*idx\_common*被正常删除成功  5、表tb\_test1的索引信息可被正常查询，其中普通索引已被删除 |
| 测试脚本 | 1. ***非图形操作***   *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--查询索引SQL语句*  *show index from tb\_test1;*  *--创建普通索引SQL语句*  *CREATE INDEX idx\_common ON tb\_test1 (name);*  *--删除普通索引SQL语句*  *drop index idx\_common on tb\_test1;*   1. ***图形操作***   *1.右键点击“新建 表”*    *2.展开所建表，在“列”分栏下面右键点击“新建列”*    3.*根据需要设置列名和字段类型*      *4.使用 ctrl + s 或按钮进行保存，并执行语句*    *5. 目录树右键单击进行新建索引*    *6.根据需要选择索引列和普通索引的类型、*    *7.ctrl+s进行保存*    *8. 普通索引创建成功后，右键单击目录树已生成的索引，选择删除*    *9. ctrl+s保存，并执行语句* |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表tb\_test1的普通索引*idx\_common*被正常创建成功，结果如下：    3、表tb\_test1的索引信息可被正常查询，包括主键索引和普通索引，结果如下：     1. 表tb\_test1的普通索引idx\_common被正常删除成功，结果如下：      1. 表tb\_test1的索引信息可被正常查询，其中普通索引已被删除，结果如下：      1. ***图形操作***   1、表newtable的普通索引newtable\_idx1被正常创建成功，目录树右侧属性栏显示结果如下：    2、右侧属性栏显示结果如下：      3、删除操作并保存后，右侧目录树索引newtable\_idx1不再出现 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持唯一索引功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持唯一索引功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表*index\_test*，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表*index\_test*中创建唯一索引*unique\_key*，SQL语句见测试脚本中描述  3、通过Waterdrop工具，查询表*index\_test*中索引信息，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表*index\_test*可成功创建。  2、表*index\_test*的唯一索引*unique\_key创建成功*  3、表*index\_test*的索引信息可被正常查询，存在创建的唯一索引 |
| 测试脚本 | ***1、非图形操作***  *--建表SQL语句*  *drop table if exists index\_test;*  *create table if not exists index\_test(id int,name varchar(20),age int,address varchar(20)) partition by hash(name) USING `binary\_md5`;*  *--添加唯一索引SQL语句*  *create unique index unique\_key on index\_test(name);*  *--查询索引SQL语句*  *show index from index\_test;*  ***2、图形操作***  *1.右键点击“新建 表”*    *2.展开所建表，在“列”分栏下面右键点击“新建列”*    3.*根据需要设置列名和字段类型*      *4.使用 ctrl + s 或按钮进行保存，并执行语句*    *5. 目录树右键单击进行新建索引*    *6.根据需要选择索引列并选中唯一索引*    7.ctrl+s 保存 |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表*index\_test*的唯一索引*unique\_key创建成功*，结果如下：    3、表*index\_test*的索引信息可被正常查询，存在创建的唯一索引，结果如下：     1. ***图形操作***   1、表newtable的唯一索引*newtable\_idx1*被正常创建成功，目录树右侧属性栏显示结果如下：  &pfm111&  2、右侧属性栏显示结果如下，可见唯一索引已被勾选中： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持删除唯一索引功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持删除唯一索引功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表*index\_test*，SQL语句见测试脚本中描述  2、通过Waterdrop工具，在表*index\_test*中创建唯一索引*unique\_key*，SQL语句见测试脚本中描述  3、通过Waterdrop工具，查询表*index\_test*中索引信息，SQL语句见测试脚本中描述  4、通过Waterdrop工具，删除表*index\_test*中唯一索引*unique\_key*，SQL语句见测试脚本中描述  5、通过Waterdrop工具，查询表*index\_test*中索引信息，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表*index\_test*可成功创建。  2、表*index\_test*的唯一索引*unique\_key创建成功*  3、表*index\_test*的索引信息可被正常查询，存在创建的唯一索引*unique\_key*  4、表*index\_test*的唯一索引*unique\_key删除成功*  5、表*index\_test*的索引信息可被正常查询，唯一索引unique\_key已被删除 |
| 测试脚本 | ***1、非图形操作***  *--建表SQL语句*  *drop table if exists index\_test;*  *create table if not exists index\_test(id int,name varchar(20),age int,address varchar(20)) partition by hash(name) USING `binary\_md5`;*  *--创建唯一索引SQL语句*  *create unique index unique\_key on index\_test(name);*  *--查询唯一索引SQL语句*  *show index from index\_test;*  *--删除唯一索引SQL语句*  *drop index unique\_key on index\_test;*  ***2、图形操作***  *1.右键点击“新建 表”*    *2.展开所建表，在“列”分栏下面右键点击“新建列”*    3.*根据需要设置列名和字段类型*      *4.使用 ctrl + s 或按钮进行保存，并执行语句*    *5. 目录树右键单击进行新建索引*    *6.根据需要选择索引列并选中唯一索引*    7.ctrl+s 保存    8.右键单击目录树所建索引，选择“删除”    9. contrl+S或按钮保存，并执行语句 |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表*index\_test*的唯一索引*unique\_key创建成功*，结果如下：    3、表*index\_test*的索引信息可被正常查询，存在创建的唯一索引，结果如下：    4、表*index\_test*的唯一索引*unique\_key删除成功，结果如下：*    5、表index\_test的索引信息可被正常查询，唯一索引unique\_key已被删除，结果如下：     1. ***图形操作***   1、表newtable的唯一索引*newtable\_idx1*被正常创建成功，目录树右侧属性栏显示结果如下：    2、右侧属性栏显示结果如下，可见唯一索引已被勾选中：      3、删除操作并保存后，右侧目录树索引newtable\_idx1不再出现 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持多健值索引可提升查询功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持多健值索引可提升查询功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1、通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表*customer*，并导入300万行数据，SQL语句见测试脚本中描述，数据导入操作见备注描述  2、通过Waterdrop工具，在表*customer*中执行查询语句，记录查询时间，SQL语句见测试脚本中描述  3、通过Waterdrop工具，给表*customer*中创建多健值索引，SQL语句见测试脚本中描述  4、通过Waterdrop工具，在表*customer*中执行查询语句，记录查询时间，SQL语句见测试脚本中描述 |
| 预期结果 | 1、表*customer* 可成功创建，300万数据均导入成功  2、查询成功  3、表*customer* 的索引信息可被正常查询，存在创建的多健值唯一索引  4、查询成功，时间比未建索引前有明显提升 |
| 测试脚本 | *--建表SQL语句*  *drop table if exists customer ;*  *create table customer (*  *c\_id integer not null,*  *c\_d\_id integer not null,*  *c\_w\_id integer not null,*  *c\_discount decimal(4,4),*  *c\_credit char(2),*  *c\_last varchar(16),*  *c\_first varchar(16),*  *c\_credit\_lim decimal(12,2),*  *c\_balance decimal(12,2),*  *c\_ytd\_payment float,*  *c\_payment\_cnt integer,*  *c\_delivery\_cnt integer,*  *c\_street\_1 varchar(20),*  *c\_street\_2 varchar(20),*  *c\_city varchar(20),*  *c\_state char(2),*  *c\_zip char(9),*  *c\_phone char(16),*  *c\_since timestamp,*  *c\_middle char(2),*  *c\_data varchar(500)*  *) PARTITION BY HASH(c\_w\_id) USING binary\_md5;*  *--查询语句*  *select \* from customer where c\_w\_id=1 and c\_d\_id=1 and c\_last='XXXXXX' and c\_first ='XXXXX';*  *--创建多健值索引SQL语句*  *create index ndx\_customer\_name on customer (c\_w\_id, c\_d\_id, c\_last, c\_first);* |
| 实际结果 | 1、表*customer* 可成功创建，300万数据均导入成功  2、查询成功，结果如下：  IMG_260  3、表*customer* 的索引信息可被正常查询，存在创建的多健值唯一索引，结果如下：    4、查询成功，时间比未建索引前有明显提升，结果如下：  IMG_256 |
| 备注 | Customer表装载数据方法详见附录1、附录2 |

### 验证KunDB支持图形化管理索引功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持图形化界面修建/编辑/删除索引功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1;  4、测试机上已成功安装Waterdrop客户端工具，且与KunDB通过AccessToken已经建立链接 |
| 测试步骤 | 1. 通过Waterdrop工具，在数据库dbtest1中执行创建表的SQL语句，创建表tb\_test1，表中存在主键，SQL语句见测试脚本中描述 2. 通过Waterdrop工具图形化操作，在数据库dbtest1中执行创建普通索引*idx\_common* 3. 通过Waterdrop工具，查询表tb\_test1中索引信息，SQL语句见测试脚本中描述 4. 通过Waterdrop工具图形化操作，在数据库dbtest1中执行删除索引*idx\_common* |
| 预期结果 | 1、表tb\_test1可成功创建  2、表tb\_test1的普通索引*idx\_common*被正常创建成功  3、表tb\_test1的索引信息可被正常查询，包括主键索引和普通索引  4、表tb\_test1的普通索引*idx\_common*被正常删除成功 |
| 测试脚本 | ***1、非图形化操作***  *--建表SQL语句*  *CREATE TABLE if not exists `tb\_test1` (*  *`id` int,*  *`zjhm` varchar(30),*  *`create\_time` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*  *`name` varchar(100) DEFAULT NULL,*  *`data` varchar(500) DEFAULT NULL,*  *`age` int(11),*  *`price` float(11, 0),*  *`tiny\_avatar` tinyblob NULL,*  *`avatar` mediumblob NULL,*  *`avatar\_small` blob NULL,*  *`extra\_info` clob null,*  *PRIMARY KEY (`id`)*  *) ENGINE=InnoDB DEFAULT CHARSET=utf8 PARTITION BY HASH(id) USING `binary\_md5`;*  *--查询索引SQL语句*  *show index from tb\_test1;*  *--创建普通索引SQL语句*  *CREATE INDEX idx\_common ON tb\_test1 (name);*  ***2、图形化操作***  *1.右键点击“新建 表”*    *2.展开所建表，在“列”分栏下面右键点击“新建列”*    3.*根据需要设置列名和字段类型*      *4.使用 ctrl + s 或按钮进行保存，并执行语句*    *5. 目录树右键单击进行新建索引*    *6.根据需要选择索引列和普通索引的类型、*    *7.ctrl+s进行保存*  、  *8.右键单击目录树索引，选择”查看索引“*    *9.修改名称：*  *右侧属性栏右键想要编辑的索引，进行重命名*    对索引名字进行修改并点击确定    .ctrl+s 或按钮进行保存，并执行语句    *10.修改索引相关列：*  右键单击目录树的索引，选择“查看索引”进行属性查看    在右侧属性栏单击列如column1，点击出现的编辑按钮    修改索引相关列    ctrl+S 或按钮保存，并执行语句 |
| 实际结果 | 1. ***非图形操作***   1、表创建成功，结果如下：    2、表tb\_test1的普通索引*idx\_common*被正常创建成功，结果如下：          3、表tb\_test1的索引信息可被正常查询，包括主键索引和普通索引，结果如下：    4、表tb\_test1的普通索引idx\_common被正常删除成功，结果如下：       1. ***图形操作***   1、右侧目录树与属性栏中索引名称修改成功，修改为newtable\_idx2      2、索引相关列修改成功，由column1修改为column1和column2 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 数据均匀分布

### 验证KunDB支持表的数据均匀分布到每个存储节点能力

|  |  |
| --- | --- |
| 测试目的 | 验证TPC-C 1000仓数据的装载时间和正确性 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有2个shard分区，且分布在不同的节点上。 |
| 测试步骤 | 1、通过benchmarksql生成1000仓数据的csv文件，  ./runLoader.sh props.kun numWarehouses 10 fileLocation /home/tpcc\_data/  2、通过BenchmarkSQL，创建表  ./runSQL.sh props.kun sqlTableCreates  3、通过BenchmarkSQL，创建索引  ./runSQL.sh props.kun sqlIndexCreates  4、通过BenchmarkSQL，将1000仓数据导入到数据库kundb1  5、分别到各个存储节点，查询各个存储节点的TPCC表的数据数量，具体查询SQL见测试脚本描述 |
| 预期结果 | 1、1000仓数据生成成功  2、TPCC表创建成功  3、TPCC表索引创建成功  4、1000仓数据导入成功  5、各个存储节点的TPCC表的数据数量均匀分布，差距不大 |
| 测试脚本 | *--数据量的一致性校验*  *select count(\*) from customer;*  *select count(\*) from district;*  *select count(\*) from history;*  *select count(\*) from item;*  *select count(\*) from new\_order;*  *select count(\*) from oorder;*  *select count(\*) from order\_line;*  *select count(\*) from stock;*  *select count(\*) from warehouse;* |
| 实际结果 | 1、10仓数据生成成功  2、TPCC表创建成功  3、TPCC表索引创建成功  4、1000仓数据导入成功  5、各个存储节点的TPCC表的数据数量均匀分布，差距不大，结果如下（以customer表威为例）：  Customer表    分区1    分区2    分区3    分区4 |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

## 图形化工具

目前KUNDB具备的配套工具有Waterdrop客户端可视化工具，具体能力请见第2章功能测试章节

## 配套工具

目前KUNDB具备TDT数据同步工具、KunLoader数据加载工具、KunDR数据备份还原等工具，具体能力请参见地5章可靠性测试章节。

## MVCC

### 验证KunDB支持MVCC功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持MVCC功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且已存在数据库dbtest1，存在三个用户分别为vt\_app,u1,u2,均具备所有权限 |
| 测试步骤 | 1、通过vt\_app用户，在数据库dbtest1中创建表test\_1，SQL语句见测试脚本中描述  2、通过u1用户，向表test\_1执行插入事务操作，通过u2用户，在u1用户插入事务提交前后，分别执行查询操作，SQL语句见测试脚本中描述 |
| 预期结果 | 1. 表test\_11创建成功 2. u1用户插入事务操作执行成功，u2 用户在u1插入事务提交前，查询不到数据，提交后可查询到提交的插入数据 |
| 测试脚本 | *--建表SQL语句*  *Create table test\_1(a int not null, b int, c char(8));*  *--一致性事务操作*   | *u1用户* | *u2用户* | | --- | --- | | *SQL>Begin work;*  *SQL>Insert into test\_1 values(1,11,'test1'),(2,12,'test2');* |  | |  | *SQL>select \* from test\_1;*    *---返回0行* | | *SQL>commit;*    *--- SQL operation complete.* |  | |  | *SQL>select \* from test\_1;*    *A           B           C*  *----------- ----------- --------*  *2          12 test2*  *1          11 test1*    *--- 查询出两条记录* | |
| 实际结果 | 1. 表test\_11创建成功，结果如下：      1. u1用户插入事务操作执行成功，u2 用户在u1插入事务提交前，查询不到数据，提交后可查询到提交的插入数据，结果如下：   IMG_256 |
| 备注 |  |

## 在线升级

### 验证KunDB支持在线升级功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持在线升级功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位。  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB-1.2.1-final和KunDB-1.3.1-final的产品包已经被成功上传。 |
| 测试步骤 | 1、使用KunDB运行一个业务，如使用BenchmarkSQL跑一个TPCC的测试。  2、在Manager界面选择要升级的KunDB服务。    点击升级，并且选择要升级的版本。    4、点击下一步，进入升级准备，再点击下一步，直到最后的升级一步 |
| 预期结果 | 1、KunDB成功升级并且不影响在线业务。 |
| 测试脚本 |  |
| 实际结果 | 2、KunDB可以在线升级成功，并且对业务没有影响 |
| 备注 |  |

# 性能效率

## 分布式读写

### 验证TPC-C 1000仓数据的装载时间和正确性

|  |  |
| --- | --- |
| 测试目的 | 验证TPC-C 1000仓数据的装载时间和正确性 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、存在已创建的数据库kundb1，有4个shard分区，且分布在不同的节点上  6、导数过程中，将Orchestrator组件停掉，防止在导数过程中Mysql发生主备切换，导致最终使用Kunloader导数工具导入的数据不一致。导数完毕后再打开。  7、Manager界面添加如下参数:  innodb\_buffer\_pool\_size=89G(根据物理机的最大内存\*70%)  innodb\_buffer\_pool\_instances=1  添加完参数后，在Manager界面配置kundb服务，使添加的参数被写到配置文件中，配置完之后重启kundb服务，使KunDB重新读取新的配置文件。 |
| 测试步骤 | 1、通过benchmarksql生成1000仓数据的csv文件，  ./runLoader.sh props.kun numWarehouses 1000 fileLocation /home/tpcc\_data/  2、通过BenchmarkSQL，创建表  ./runSQL.sh props.kun sqlTableCreates  3、通过BenchmarkSQL，创建索引  ./runSQL.sh props.kun sqlIndexCreates  4、通过KunLoader工具，将1000仓数据导入到数据库kundb1  5、执行TPCC基准测试一致性验证语句，验证导入结果的正确性，具体执行语句见测试脚本。 |
| 预期结果 | 1、1000仓数据生成成功  2、TPCC表创建成功  3、C表索引创建成功  4、1000仓数据导入成功  5、TPC-C一致性校验语句返回的结果全部为0行 |
| 测试脚本 | *--数据量的一致性校验*  *select count(\*) from customer; ---3千万行*  *select count(\*) from district; ---1万行*  *select count(\*) from history; ---3千万行*  *select count(\*) from item; ---10万行*  *select count(\*) from new\_order; ---9百万行*  *select count(\*) from oorder; ---3千万行*  *select count(\*) from order\_line; ---300002113行*  *select count(\*) from stock; ---1亿行*  *select count(\*) from warehouse; ---1千行*  *--TPC-C一致性校验语句*  *select a.\* from (Select w\_id, w\_ytd from warehouse) a left join (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) b on a.w\_id=b.d\_w\_id and a.w\_ytd=b.s where b.d\_w\_id is null;*  *select b.\* from (Select w\_id, w\_ytd from warehouse) a right join (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) b on a.w\_id=b.d\_w\_id and a.w\_ytd=b.s where a.w\_id is null;*  *select a.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a left join (select o\_w\_id, o\_d\_id, max(o\_id) m from oorder group by o\_w\_id, o\_d\_id) b on a.d\_w\_id=b.o\_w\_id and a.d\_id=b.o\_d\_id and a.d=b.m where b.o\_w\_id is null;*  *select b.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a right join (select o\_w\_id, o\_d\_id, max(o\_id) m from oorder group by o\_w\_id, o\_d\_id) b on a.d\_w\_id=b.o\_w\_id and a.d\_id=b.o\_d\_id and a.d=b.m where a.d\_w\_id is null;*  *select a.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a left join (select no\_w\_id, no\_d\_id, max(no\_o\_id) m from new\_order group by no\_w\_id, no\_d\_id) b on a.d\_w\_id=b.no\_w\_id and a.d\_id=b.no\_d\_id and a.d=b.m where b.no\_w\_id is null;*  *select b.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a right join (select no\_w\_id, no\_d\_id, max(no\_o\_id) m from new\_order group by no\_w\_id, no\_d\_id) b on a.d\_w\_id=b.no\_w\_id and a.d\_id=b.no\_d\_id and a.d=b.m where a.d\_w\_id is null;*  *select \* from (select (count(no\_o\_id)-(max(no\_o\_id)-min(no\_o\_id)+1)) as diff from new\_order group by no\_w\_id, no\_d\_id) where diff != 0;*    *select a.\* from (select o\_w\_id, o\_d\_id, sum(o\_ol\_cnt) s from oorder group by o\_w\_id, o\_d\_id) a left join (select ol\_w\_id, ol\_d\_id, count(ol\_o\_id) c from order\_line group by ol\_w\_id, ol\_d\_id) b on a.o\_w\_id=b.ol\_w\_id and a.o\_d\_id=b.ol\_d\_id and a.s=b.c where b.ol\_w\_id is null;*  *select b.\* from (select o\_w\_id, o\_d\_id, sum(o\_ol\_cnt) s from oorder group by o\_w\_id, o\_d\_id) a right join (select ol\_w\_id, ol\_d\_id, count(ol\_o\_id) c from order\_line group by ol\_w\_id, ol\_d\_id) b on a.o\_w\_id=b.ol\_w\_id and a.o\_d\_id=b.ol\_d\_id and a.s=b.c where a.o\_w\_id is null;*  *select a.\* from (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) a left join (Select w\_id, w\_ytd from warehouse) b on a.d\_w\_id=b.w\_id and a.s=b.w\_ytd where b.w\_id is null;*  *select b.\* from (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) a right join (Select w\_id, w\_ytd from warehouse) b on a.d\_w\_id=b.w\_id and a.s=b.w\_ytd where a.d\_w\_id is null;* |
| 实际结果 | 1、1000仓数据生成成功    2、TPCC表创建成功    3、TPCC表索引创建成功    4、1000仓数据导入成功，共耗时11m左右  b26ae0c295645ea36ffb1a5e9458e3a      5、TPC-C一致性校验语句返回的结果全部为0行 |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

## TPC-C模拟测试

### 验证1000并发用户的TPC-C模拟测试的tpmC和数据一致性

|  |  |
| --- | --- |
| 测试目的 | 验证1000并发用户的TPC-C模拟测试的tpmC |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有4个shard分区，且分布在不同的节点上。  6、1000仓的数据已经通过BenchmarkSQL工具生成并已导入到数据库kundb1中  7、Benchmarksql中的props.kun参数设置：  warehouses=1000  terminals=1000  runMins=120  newOrderWeight=45  paymentWeight=43  orderStatusWeight=4  deliveryWeight=4  stockLevelWeight=4  8、测试性能时，manager界面关闭Aquila中日志收集相关角色（Filebeat、Kafka Server、Logstash、Search Server），因性能测试需要使用几乎整个机器的内存。  9、测试前，在manager所在的节点修改以下参数  /var/lib/transwarp-manager/master/content/meta/services/KUNDB/kundb-1.2.1-final/templates/mfed-up.sh.raw 文件中的参数  -queryserver-config-transaction-timeout 90（修改事务超时时间为90s）  /var/lib/transwarp-manager/master/content/meta/services/KUNDB/kundb-1.2.1-final  /templates/kuntablet-up.sh.raw 文件中的参数  -queryserver-config-transaction-timeout 90（修改事务超时时间为90s）  10、在manager界面添加参数如下（mysql.cfg）  innodb\_buffer\_pool\_instances=1  innodb\_lock\_wait\_timeout=30  innodb\_log\_buffer\_size=3G  innodb\_log\_file\_size=1G  innodb\_read\_io\_threads=16  innodb\_write\_io\_threads=16  net\_read\_timeout=300  net\_write\_timeout=300  transaction-isolation=READ-COMMITTED  innodb\_doublewrite = 0  11、在manager界面修改参数如下  kundb.always\_set\_effective\_user=false  kundb.enable\_twopc=false  12、manager界面将KunDB审计日志关闭  computenode.audit\_filter、kungate.audit\_filter、kuntablet.audit\_filter均置为空  13、在修改完以上所有参数后，配置KunDB服务，配置完之后再重启KunDB |
| 测试步骤 | 1、通过benchmarksql工具，执行TPC-C事务操作，执行2个小时，具体操作步骤见附件  2、观察执行过程中各个节点的资源消耗情况，并记录执行结果执行语句见测试脚本  3、执行完成后，进行数据一致性校验，具体执行语句见测试脚本 |
| 预期结果 | 1、2个小时TPC-C事务操作执行成功  2、各个计算节点和存储节点的资源占用情况分布均匀  3、TPC-C一致性校验语句返回的结果全部为0行 |
| 测试脚本 | --TPC-C一致性校验语句  select a.\* from (Select w\_id, w\_ytd from warehouse) a left join (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) b on a.w\_id=b.d\_w\_id and a.w\_ytd=b.s where b.d\_w\_id is null;  select b.\* from (Select w\_id, w\_ytd from warehouse) a right join (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) b on a.w\_id=b.d\_w\_id and a.w\_ytd=b.s where a.w\_id is null;  select a.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a left join (select o\_w\_id, o\_d\_id, max(o\_id) m from oorder group by o\_w\_id, o\_d\_id) b on a.d\_w\_id=b.o\_w\_id and a.d\_id=b.o\_d\_id and a.d=b.m where b.o\_w\_id is null;  select b.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a right join (select o\_w\_id, o\_d\_id, max(o\_id) m from oorder group by o\_w\_id, o\_d\_id) b on a.d\_w\_id=b.o\_w\_id and a.d\_id=b.o\_d\_id and a.d=b.m where a.d\_w\_id is null;  select a.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a left join (select no\_w\_id, no\_d\_id, max(no\_o\_id) m from new\_order group by no\_w\_id, no\_d\_id) b on a.d\_w\_id=b.no\_w\_id and a.d\_id=b.no\_d\_id and a.d=b.m where b.no\_w\_id is null;  select b.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a right join (select no\_w\_id, no\_d\_id, max(no\_o\_id) m from new\_order group by no\_w\_id, no\_d\_id) b on a.d\_w\_id=b.no\_w\_id and a.d\_id=b.no\_d\_id and a.d=b.m where a.d\_w\_id is null;  select \* from (select (count(no\_o\_id)-(max(no\_o\_id)-min(no\_o\_id)+1)) as diff from new\_order group by no\_w\_id, no\_d\_id) where diff != 0;    select a.\* from (select o\_w\_id, o\_d\_id, sum(o\_ol\_cnt) s from oorder group by o\_w\_id, o\_d\_id) a left join (select ol\_w\_id, ol\_d\_id, count(ol\_o\_id) c from order\_line group by ol\_w\_id, ol\_d\_id) b on a.o\_w\_id=b.ol\_w\_id and a.o\_d\_id=b.ol\_d\_id and a.s=b.c where b.ol\_w\_id is null;  select b.\* from (select o\_w\_id, o\_d\_id, sum(o\_ol\_cnt) s from oorder group by o\_w\_id, o\_d\_id) a right join (select ol\_w\_id, ol\_d\_id, count(ol\_o\_id) c from order\_line group by ol\_w\_id, ol\_d\_id) b on a.o\_w\_id=b.ol\_w\_id and a.o\_d\_id=b.ol\_d\_id and a.s=b.c where a.o\_w\_id is null;  select a.\* from (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) a left join (Select w\_id, w\_ytd from warehouse) b on a.d\_w\_id=b.w\_id and a.s=b.w\_ytd where b.w\_id is null;  select b.\* from (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) a right join (Select w\_id, w\_ytd from warehouse) b on a.d\_w\_id=b.w\_id and a.s=b.w\_ytd where a.d\_w\_id is null; |
| 实际结果 | 1、2个小时TPC-C事务操作执行成功，结果如下：  e21d2bc31150a19fe08fad30baf58d7  tpmC（NewOrder）：1292312.9  2、各个计算节点和存储节点的资源占用情况分布均匀  3、TPC-C执行一致性校验语句返回的结果全部为0行，结果如下： |
| 备注 | 装载数据和TPCC建表、建索引、执行事务操作等方法详见附录1，附录2 |

### 验证3000并发用户的TPC-C模拟测试的tpmC和数据一致性

|  |  |
| --- | --- |
| 测试目的 | 验证3000并发用户的TPC-C模拟测试的tpmC和和数据一致性 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有4个shard分区，且分布在不同的节点上。  6、3000仓的数据已经通过BenchmarkSQL工具生成并已导入到数据库kundb1中  7、Benchmarksql中的props.kun参数设置：  warehouses=1000  terminals=3000  runMins=120  newOrderWeight=45  paymentWeight=43  orderStatusWeight=4  deliveryWeight=4  stockLevelWeight=4  8、测试性能时，manager界面关闭Aquila中日志收集相关角色（Filebeat、Kafka Server、Logstash、Search Server），因性能测试需要使用几乎整个机器的内存。  9、测试前，在manager所在的节点修改以下参数:  /var/lib/transwarp-manager/master/content/meta/services/KUNDB/kundb-1.2.1-final/templates/mfed-up.sh.raw 文件中的参数  -queryserver-config-transaction-timeout 90（修改事务超时时间为90s）  /var/lib/transwarp-manager/master/content/meta/services/KUNDB/kundb-1.2.1-final/templates/kuntablet-up.sh.raw 文件中的参数  -queryserver-config-transaction-timeout 90（修改事务超时时间为90s）  10、在manager界面添加参数如下（mysql.cfg）  innodb\_buffer\_pool\_instances=1  innodb\_lock\_wait\_timeout=30  innodb\_log\_buffer\_size=3G  innodb\_log\_file\_size=1G  innodb\_read\_io\_threads=16  innodb\_write\_io\_threads=16  net\_read\_timeout=300  net\_write\_timeout=300  transaction-isolation=READ-COMMITTED  innodb\_doublewrite = 0  11、在manager界面修改参数如下  kundb.always\_set\_effective\_user=false  kundb.enable\_twopc=false  12、manager界面将KunDB审计日志关闭  computenode.audit\_filter、kungate.audit\_filter、kuntablet.audit\_filter均置为空  13、在修改完以上所有参数后，配置KunDB服务，配置完之后再重启KunDB |
| 测试步骤 | 1、通过benchmarksql工具，执行TPC-C事务操作，执行2个小时，具体操作步骤见附件  2、观察执行过程中各个节点的资源消耗情况，并记录执行结果执行语句见测试脚本  3、执行完成后，进行数据一致性校验，具体执行语句见测试脚本 |
| 预期结果 | 1、2个小时TPC-C事务操作执行成功  2、各个计算节点和存储节点的资源占用情况分布均匀  3、TPC-C执行一致性校验语句返回的结果全部为0行 |
| 测试脚本 | --TPC-C一致性校验语句  select a.\* from (Select w\_id, w\_ytd from warehouse) a left join (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) b on a.w\_id=b.d\_w\_id and a.w\_ytd=b.s where b.d\_w\_id is null;  select b.\* from (Select w\_id, w\_ytd from warehouse) a right join (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) b on a.w\_id=b.d\_w\_id and a.w\_ytd=b.s where a.w\_id is null;  select a.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a left join (select o\_w\_id, o\_d\_id, max(o\_id) m from oorder group by o\_w\_id, o\_d\_id) b on a.d\_w\_id=b.o\_w\_id and a.d\_id=b.o\_d\_id and a.d=b.m where b.o\_w\_id is null;  select b.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a right join (select o\_w\_id, o\_d\_id, max(o\_id) m from oorder group by o\_w\_id, o\_d\_id) b on a.d\_w\_id=b.o\_w\_id and a.d\_id=b.o\_d\_id and a.d=b.m where a.d\_w\_id is null;  select a.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a left join (select no\_w\_id, no\_d\_id, max(no\_o\_id) m from new\_order group by no\_w\_id, no\_d\_id) b on a.d\_w\_id=b.no\_w\_id and a.d\_id=b.no\_d\_id and a.d=b.m where b.no\_w\_id is null;  select b.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a right join (select no\_w\_id, no\_d\_id, max(no\_o\_id) m from new\_order group by no\_w\_id, no\_d\_id) b on a.d\_w\_id=b.no\_w\_id and a.d\_id=b.no\_d\_id and a.d=b.m where a.d\_w\_id is null;  select \* from (select (count(no\_o\_id)-(max(no\_o\_id)-min(no\_o\_id)+1)) as diff from new\_order group by no\_w\_id, no\_d\_id) where diff != 0;    select a.\* from (select o\_w\_id, o\_d\_id, sum(o\_ol\_cnt) s from oorder group by o\_w\_id, o\_d\_id) a left join (select ol\_w\_id, ol\_d\_id, count(ol\_o\_id) c from order\_line group by ol\_w\_id, ol\_d\_id) b on a.o\_w\_id=b.ol\_w\_id and a.o\_d\_id=b.ol\_d\_id and a.s=b.c where b.ol\_w\_id is null;  select b.\* from (select o\_w\_id, o\_d\_id, sum(o\_ol\_cnt) s from oorder group by o\_w\_id, o\_d\_id) a right join (select ol\_w\_id, ol\_d\_id, count(ol\_o\_id) c from order\_line group by ol\_w\_id, ol\_d\_id) b on a.o\_w\_id=b.ol\_w\_id and a.o\_d\_id=b.ol\_d\_id and a.s=b.c where a.o\_w\_id is null;  select a.\* from (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) a left join (Select w\_id, w\_ytd from warehouse) b on a.d\_w\_id=b.w\_id and a.s=b.w\_ytd where b.w\_id is null;  select b.\* from (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) a right join (Select w\_id, w\_ytd from warehouse) b on a.d\_w\_id=b.w\_id and a.s=b.w\_ytd where a.d\_w\_id is null; |
| 实际结果 | 1、2个小时TPC-C事务操作执行成功，结果如下：  938eae0731283b40e67eccba99b2e04  tpmC（NewOrder）：206618.7  2、各个计算节点和存储节点的资源占用情况分布均匀  3、TPC-C执行一致性校验语句返回的结果全部为0行，结果如下： |
| 备注 | 装载数据和TPCC建表、建索引、执行事务操作等方法详见附录1，附录2 |

## 海量数据复杂查询、Join查询性能

### 验证KunDB系统海量数据复杂查询、Join查询的性能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统海量数据复杂查询、Join查询的性能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有4个shard分区，且分布在不同的节点上。 |
| 测试步骤 | 1. 通过Waterdrop工具，在数据库dbtest1中执行创建表*test\_tab1、test\_tab2*，SQL语句见测试脚本中描述 2. 用kundb导数工具KunLoader向两张表导入对应数量级的数据量 3. 对表创建索引,SQL语句见测试脚本中描述 4. 查询前执行analyze table,SQL语句见测试脚本中描述 5. 查询SQL1,SQL2,SQL3,并记录返回结果数量与执行时间 |
| 预期结果 | 1. 表test\_tab1、test\_tab2创建成功 2. 表test\_tab1、test\_tab2数据导入成功 3. 表test\_tab1、test\_tab2索引创建成功 4. Analyze table后查询SQL1,SQL2,SQL3返回结果数量与预期一致,执行时间较短 |
| 测试脚本 | *--建表语句*  *--模拟账户信息表(基础数据:百万级)*  create table test\_tab1( codeid numeric(10,0), prov varchar(50), city varchar(50), country varchar(50), name varchar(20), birth datetime, comm varchar(100) ) ENGINE=InnoDB partition by hash(codeid) using binary\_md5;  *--模拟账户交易信息表（基础数据:千万级）*  create table test\_tab2( codeid numeric(10,0), account\_id numeric(13,0), deal\_dt datetime, deal\_type numeric(10,0), deal\_money numeric(18,3), charge\_money numeric(18,3), lock\_type numeric(10,0), expire\_time datetime, comms varchar(100) ) ENGINE=InnoDB partition by hash(codeid) using binary\_md5;  ---创建索引语句  CREATE INDEX i\_t1\_name ON test\_tab1 (`name`);  CREATE INDEX i\_t1\_cc ON test\_tab1(`city`,`country`);  CREATE INDEX i\_t2\_cdd ON test\_tab2(`codeid`,`deal\_dt`,`deal\_type`);  ---analyze table语句  analyze table test\_tab1;  analyze table test\_tab2; |
| 实际结果 | 1. 表test\_tab1、test\_tab2创建成功 2. 表test\_tab1、test\_tab2数据导入成功 3. 表test\_tab1、test\_tab2索引创建成功   4、Analyze table后查询SQL1,SQL2,SQL3返回结果数量与预期一致,执行时间较短 |
| 备注 |  |

# 可靠性

## 168小时稳定性测试

### 验证500并发用户、500仓数据的168小时TPC-C模拟测试

|  |  |
| --- | --- |
| 测试目的 | 验证500并发用户、500仓数据的168小时TPC-C模拟测试 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有4个shard分区，且分布在不同的节点上。  6、500仓的数据已经通过BenchmarkSQL工具生成并已导入到数据库kundb1中  7、Benchmarksql中的props.kun参数设置：  warehouses=500  terminals=500  runMins=241920 |
| 测试步骤 | 1、通过benchmarksql工具，执行TPC-C事务操作，执行168个小时，观察执行过程中各个节点的资源消耗情况，并记录执行结果，具体操作步骤见备注，执行语句见测试脚本  2、执行完成后，进行数据一致性校验，具体执行语句见测试脚本 |
| 预期结果 | 1、168个小时TPC-C事务操作执行成功，各个计算节点和存储节点的资源占用情况分布均匀  2、TPC-C一致性校验通过 |
| 测试脚本 | --TPC-C一致性校验语句  select a.\* from (Select w\_id, w\_ytd from warehouse) a left join (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) b on a.w\_id=b.d\_w\_id and a.w\_ytd=b.s where b.d\_w\_id is null;  select b.\* from (Select w\_id, w\_ytd from warehouse) a right join (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) b on a.w\_id=b.d\_w\_id and a.w\_ytd=b.s where a.w\_id is null;  select a.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a left join (select o\_w\_id, o\_d\_id, max(o\_id) m from oorder group by o\_w\_id, o\_d\_id) b on a.d\_w\_id=b.o\_w\_id and a.d\_id=b.o\_d\_id and a.d=b.m where b.o\_w\_id is null;  select b.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a right join (select o\_w\_id, o\_d\_id, max(o\_id) m from oorder group by o\_w\_id, o\_d\_id) b on a.d\_w\_id=b.o\_w\_id and a.d\_id=b.o\_d\_id and a.d=b.m where a.d\_w\_id is null;  select a.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a left join (select no\_w\_id, no\_d\_id, max(no\_o\_id) m from new\_order group by no\_w\_id, no\_d\_id) b on a.d\_w\_id=b.no\_w\_id and a.d\_id=b.no\_d\_id and a.d=b.m where b.no\_w\_id is null;  select b.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a right join (select no\_w\_id, no\_d\_id, max(no\_o\_id) m from new\_order group by no\_w\_id, no\_d\_id) b on a.d\_w\_id=b.no\_w\_id and a.d\_id=b.no\_d\_id and a.d=b.m where a.d\_w\_id is null;  select \* from (select (count(no\_o\_id)-(max(no\_o\_id)-min(no\_o\_id)+1)) as diff from new\_order group by no\_w\_id, no\_d\_id) where diff != 0;    select a.\* from (select o\_w\_id, o\_d\_id, sum(o\_ol\_cnt) s from oorder group by o\_w\_id, o\_d\_id) a left join (select ol\_w\_id, ol\_d\_id, count(ol\_o\_id) c from order\_line group by ol\_w\_id, ol\_d\_id) b on a.o\_w\_id=b.ol\_w\_id and a.o\_d\_id=b.ol\_d\_id and a.s=b.c where b.ol\_w\_id is null;  select b.\* from (select o\_w\_id, o\_d\_id, sum(o\_ol\_cnt) s from oorder group by o\_w\_id, o\_d\_id) a right join (select ol\_w\_id, ol\_d\_id, count(ol\_o\_id) c from order\_line group by ol\_w\_id, ol\_d\_id) b on a.o\_w\_id=b.ol\_w\_id and a.o\_d\_id=b.ol\_d\_id and a.s=b.c where a.o\_w\_id is null;  select a.\* from (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) a left join (Select w\_id, w\_ytd from warehouse) b on a.d\_w\_id=b.w\_id and a.s=b.w\_ytd where b.w\_id is null;  select b.\* from (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) a right join (Select w\_id, w\_ytd from warehouse) b on a.d\_w\_id=b.w\_id and a.s=b.w\_ytd where a.d\_w\_id is null; |
| 实际结果 | 1、168个小时TPC-C事务操作执行成功  2、TPC-C执行一致性校验语句返回的结果全部为0行，结果如下： |
| 备注 | 装载数据和TPCC建表、建索引、执行事务操作等方法详见附录1，附录2 |

## 模拟节点断电测试

### 验证KunDB系统节点断电情况下的运行情况

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统节点断电情况下的运行情况 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、存在已创建的数据库kundb1，有4个shard分区，且分布在不同的节点上  6、Loadrunner与webService程序均已调试好  7、在Manager界面添加/修改以下参数  innodb\_buffer\_pool\_size=30G  innodb\_flush\_log\_at\_trx\_commit=1  sync\_binlog=1  kundb.enable\_twopc=true  修改/添加完参数后，在Manager界面配置KunDB服务，配置完之后重启KunDB。 |
| 测试步骤 | 1、通过loadrunner对web应用进行加压，并发数为100用户，在压力测试过程中，随意关掉任1个计算节点电源，观察Loadrunner运行情况，具体操作方法见备注  2、在loadrunner运行平稳后，再次启动关闭的节点，观察Loadrunner运行情况 |
| 预期结果 | 1、关掉任1个计算节点电源，只是性能有所下降，程序不退出  2、再次启动关闭的节点后，系统性能可以回升到原来的水平 |
| 测试脚本 |  |
| 实际结果 | 1、关掉任1个计算节点电源，只是性能有所下降，程序不退出，结果如下：  C:\Users\tlk\AppData\Local\Temp\WeChat Files\e58edebcca72f7837d42b8e545e05a5.png  2、再次启动关闭的节点后，系统性能可以回升到原来的水平，结果如下：  C:\Users\tlk\AppData\Local\Temp\WeChat Files\e58edebcca72f7837d42b8e545e05a5.png |
| 备注 | Loadrunner与webService程序及详细使用方法见附件5 |

## 模拟节点断网测试

### 验证KunDB系统节点断网情况下的运行情况

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统节点断网情况下的运行情况 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有4个shard分区，且分布在不同的节点上。  6、Loadrunner与webService程序均已调试好。  7、在Manager界面添加/修改以下参数  innodb\_buffer\_pool\_size=30G  innodb\_flush\_log\_at\_trx\_commit=1  sync\_binlog=1  kundb.enable\_twopc=true  修改/添加完参数后，在Manager界面配置KunDB服务，配置完之后重启KunDB。 |
| 测试步骤 | 1、通过loadrunner对web应用进行加压，并发数为100用户，在压力测试过程中，随意拔掉一个计算节点的网线，观察Loadrunner运行情况  2、在loadrunner运行平稳后，再次插上断网额计算节点的网线，观察Loadrunner运行情况 |
| 预期结果 | 1、拔掉一个计算节点的网线，只是性能有所下降，程序不退出  2、再次插上断网额计算节点的网线，系统性能可以回升到原来的水平 |
| 测试脚本 |  |
| 实际结果 | 1、拔掉一个计算节点的网线，只是性能有所下降，程序不退出，结果如下：  C:\Users\tlk\AppData\Local\Temp\WeChat Files\e58edebcca72f7837d42b8e545e05a5.png  2、再次插上断网额计算节点的网线，系统性能可以回升到原来的水平，结果如下：  C:\Users\tlk\AppData\Local\Temp\WeChat Files\e58edebcca72f7837d42b8e545e05a5.png |
| 备注 | Loadrunner与webService程序及详细使用方法见附件5 |

## 全断网测试

### 验证KunDB系统节点全断电情况下的运行情况

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统节点全断电情况下的运行情况 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有4个shard分区，且分布在不同的节点上。  6、1000仓的数据已经通过BenchmarkSQL工具生成并已导入到数据库kundb1中。 |
| 测试步骤 | 1、通过benchmarksql工具，执行TPC-C事务操作，在运行10分钟后，断电全部计算节点，具体操作步骤见备注  2、重启所有计算节点，执行TPC-C一致性校验，具体执行语句见测试脚本 |
| 预期结果 | 1、全部节点断电后，TPC-C执行失败  2、重启所有计算节点，TPC-C一致性校验成功，无数据异常 |
| 测试脚本 | --TPC-C一致性校验语句  select a.\* from (Select w\_id, w\_ytd from warehouse) a left join (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) b on a.w\_id=b.d\_w\_id and a.w\_ytd=b.s where b.d\_w\_id is null;  select b.\* from (Select w\_id, w\_ytd from warehouse) a right join (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) b on a.w\_id=b.d\_w\_id and a.w\_ytd=b.s where a.w\_id is null;  select a.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a left join (select o\_w\_id, o\_d\_id, max(o\_id) m from oorder group by o\_w\_id, o\_d\_id) b on a.d\_w\_id=b.o\_w\_id and a.d\_id=b.o\_d\_id and a.d=b.m where b.o\_w\_id is null;  select b.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a right join (select o\_w\_id, o\_d\_id, max(o\_id) m from oorder group by o\_w\_id, o\_d\_id) b on a.d\_w\_id=b.o\_w\_id and a.d\_id=b.o\_d\_id and a.d=b.m where a.d\_w\_id is null;  select a.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a left join (select no\_w\_id, no\_d\_id, max(no\_o\_id) m from new\_order group by no\_w\_id, no\_d\_id) b on a.d\_w\_id=b.no\_w\_id and a.d\_id=b.no\_d\_id and a.d=b.m where b.no\_w\_id is null;  select b.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a right join (select no\_w\_id, no\_d\_id, max(no\_o\_id) m from new\_order group by no\_w\_id, no\_d\_id) b on a.d\_w\_id=b.no\_w\_id and a.d\_id=b.no\_d\_id and a.d=b.m where a.d\_w\_id is null;  select \* from (select (count(no\_o\_id)-(max(no\_o\_id)-min(no\_o\_id)+1)) as diff from new\_order group by no\_w\_id, no\_d\_id) where diff != 0;    select a.\* from (select o\_w\_id, o\_d\_id, sum(o\_ol\_cnt) s from oorder group by o\_w\_id, o\_d\_id) a left join (select ol\_w\_id, ol\_d\_id, count(ol\_o\_id) c from order\_line group by ol\_w\_id, ol\_d\_id) b on a.o\_w\_id=b.ol\_w\_id and a.o\_d\_id=b.ol\_d\_id and a.s=b.c where b.ol\_w\_id is null;  select b.\* from (select o\_w\_id, o\_d\_id, sum(o\_ol\_cnt) s from oorder group by o\_w\_id, o\_d\_id) a right join (select ol\_w\_id, ol\_d\_id, count(ol\_o\_id) c from order\_line group by ol\_w\_id, ol\_d\_id) b on a.o\_w\_id=b.ol\_w\_id and a.o\_d\_id=b.ol\_d\_id and a.s=b.c where a.o\_w\_id is null;  select a.\* from (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) a left join (Select w\_id, w\_ytd from warehouse) b on a.d\_w\_id=b.w\_id and a.s=b.w\_ytd where b.w\_id is null;  select b.\* from (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) a right join (Select w\_id, w\_ytd from warehouse) b on a.d\_w\_id=b.w\_id and a.s=b.w\_ytd where a.d\_w\_id is null; |
| 实际结果 | 1、全部节点断电后，TPC-C执行失败  2、重启所有计算节点，TPC-C一致性校验成功，无数据异常，结果如下： |
| 备注 | 装载数据和TPCC建表、建索引、执行事务操作等方法详见附录1，附录2 |

## 全断电测试

### 验证KunDB系统节点全断网情况下的运行情况

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统节点全断网情况下的运行情况 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有4个shard分区，且分布在不同的节点上。  6、1000仓的数据已经通过BenchmarkSQL工具生成并已导入到数据库kundb1中。 |
| 测试步骤 | 1、通过benchmarksql工具，执行TPC-C事务操作，在运行10分钟后，断网全部计算节点，具体操作步骤见备注  2、重新联网所有计算节点，执行TPC-C一致性校验，具体执行语句见测试脚本 |
| 预期结果 | 1、全部节点断网后，TPC-C执行失败  2、重新联网所有计算节点，TPC-C一致性校验成功，无数据异常 |
| 测试脚本 | --TPC-C一致性校验语句  select a.\* from (Select w\_id, w\_ytd from warehouse) a left join (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) b on a.w\_id=b.d\_w\_id and a.w\_ytd=b.s where b.d\_w\_id is null;  select b.\* from (Select w\_id, w\_ytd from warehouse) a right join (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) b on a.w\_id=b.d\_w\_id and a.w\_ytd=b.s where a.w\_id is null;  select a.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a left join (select o\_w\_id, o\_d\_id, max(o\_id) m from oorder group by o\_w\_id, o\_d\_id) b on a.d\_w\_id=b.o\_w\_id and a.d\_id=b.o\_d\_id and a.d=b.m where b.o\_w\_id is null;  select b.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a right join (select o\_w\_id, o\_d\_id, max(o\_id) m from oorder group by o\_w\_id, o\_d\_id) b on a.d\_w\_id=b.o\_w\_id and a.d\_id=b.o\_d\_id and a.d=b.m where a.d\_w\_id is null;  select a.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a left join (select no\_w\_id, no\_d\_id, max(no\_o\_id) m from new\_order group by no\_w\_id, no\_d\_id) b on a.d\_w\_id=b.no\_w\_id and a.d\_id=b.no\_d\_id and a.d=b.m where b.no\_w\_id is null;  select b.\* from (Select d\_w\_id, d\_id, (D\_NEXT\_O\_ID - 1) d from district) a right join (select no\_w\_id, no\_d\_id, max(no\_o\_id) m from new\_order group by no\_w\_id, no\_d\_id) b on a.d\_w\_id=b.no\_w\_id and a.d\_id=b.no\_d\_id and a.d=b.m where a.d\_w\_id is null;  select \* from (select (count(no\_o\_id)-(max(no\_o\_id)-min(no\_o\_id)+1)) as diff from new\_order group by no\_w\_id, no\_d\_id) where diff != 0;    select a.\* from (select o\_w\_id, o\_d\_id, sum(o\_ol\_cnt) s from oorder group by o\_w\_id, o\_d\_id) a left join (select ol\_w\_id, ol\_d\_id, count(ol\_o\_id) c from order\_line group by ol\_w\_id, ol\_d\_id) b on a.o\_w\_id=b.ol\_w\_id and a.o\_d\_id=b.ol\_d\_id and a.s=b.c where b.ol\_w\_id is null;  select b.\* from (select o\_w\_id, o\_d\_id, sum(o\_ol\_cnt) s from oorder group by o\_w\_id, o\_d\_id) a right join (select ol\_w\_id, ol\_d\_id, count(ol\_o\_id) c from order\_line group by ol\_w\_id, ol\_d\_id) b on a.o\_w\_id=b.ol\_w\_id and a.o\_d\_id=b.ol\_d\_id and a.s=b.c where a.o\_w\_id is null;  select a.\* from (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) a left join (Select w\_id, w\_ytd from warehouse) b on a.d\_w\_id=b.w\_id and a.s=b.w\_ytd where b.w\_id is null;  select b.\* from (select d\_w\_id, sum(d\_ytd) s from district group by d\_w\_id) a right join (Select w\_id, w\_ytd from warehouse) b on a.d\_w\_id=b.w\_id and a.s=b.w\_ytd where a.d\_w\_id is null; |
| 实际结果 | 1、全部节点断网后，TPC-C执行失败，结果如下：  2、重新联网所有计算节点，TPC-C校验成功，无数据异常，结果如下： |
| 备注 | 装载数据和TPCC建表、建索引、执行事务操作等方法详见附录1，附录2 |

## 逻辑备份还原

### 验证KunDB系统逻辑备份还原功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统逻辑备份还原功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有5个shard分区，且分布在不同的节点上。 |
| 测试步骤 | 1. 准备数据,创建两张表,并导入10w行数据,创建视图,存储过程,触发器 2. 配置KunDB的备份还原工具KunDR，选择需要备份的对象(指定的数据库下的所有对象)，执行逻辑备份操作，具体操作方法见备注   3、备份好之后，进入kundb中删除之前备份的对象,制造数据丢失的灾难  4.执行逻辑备份还原操作，具体操作方法见备注  5.检查还原后数据的正确性  6.配置KunDB的备份还原工具KunDR，选择需要备份的对象(指定的表,视图,存储过程,触发器,函数等)，执行逻辑备份,还原,操作同上 |
| 预期结果 | 1、选择的对象备份成功  2、对象还原成功，数据检查正确 |
| 测试脚本 | --创建两张表,并导入数据  create table iback\_tbl1(  codeid bigint,  pad bigint,  name varchar(50),  birth datetime,  primary key(codeid)  ) ENGINE=InnoDB partition by hash(codeid) using binary\_md5;  create table iback\_tbl2(  codeid bigint,  pad bigint,  name varchar(50),  birth datetime,  primary key(codeid)  ) ENGINE=InnoDB partition by hash(codeid) using binary\_md5;  数据生成脚本:  cat load.sh  #!/bin/bash  i=1  if [ -e /tmp/load.sql ];then  rm -rf /tmp/load.sql  fi  while [ $i -le 100000 ]  do  echo "insert into iback\_tbl1 values($i,1,'NA$i','2020-01-01');" >> /tmp/load.sql  echo "insert into iback\_tbl2 values($i,1,'NA$i','2020-01-01');" >> /tmp/load.sql  let i++  done  执行bash load.sh生成导数sql文件load,sql  进入kundb对应的数据库,执行source命令进行导数:  Source /tmp/load.sql;  --创建视图  见3.6.1创建视图对象功能测试脚本  --创建存储过程  见3.12.1创建存储过程功能测试脚本  --创建触发器  见3.13.1创建触发器功能测试脚本  检查视图,存储过程,触发器:  Select \* from view\_test;  Show create procedure NewProc;  Show create trigger newtrigger;  检查两张表的数据:  --iback\_tbl1:  select count(\*) from iback\_tbl1;  select min(codeid) from iback\_tbl1;  select max(codeid) from iback\_tbl1;  --iback\_tbl2:  select count(distinct pad) from iback\_tbl2;  select count(pad) from iback\_tbl2 where pad>1;  select count(pad) from iback\_tbl2 where pad=1; |
| 实际结果 | 1. 选择的对象备份成功，结果如下：   备份指定的数据库下的所有对象:  选区_737  备份指定的表,视图,存储过程,触发器,函数:  选区_738   1. 对象还原成功，数据检查正确，结果如下：   2020-07-09 11-13-51屏幕截图  2020-07-09 11-14-23屏幕截图 |
| 备注 | 1、KunDR逻辑备份还原的详细操作见附录3 |

## 完全备份还原

### 验证KunDB系统完全备份还原功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统完全备份还原功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有5个shard分区，且分布在不同的节点上。  6.运行备份还原工具所在的机器和kundb shard instance所在的节点需要装有rsync工具  7.运行备份还原工具所在的机器和kundb shard instance所在的节点需要ssh免秘钥登录  8.运行备份还原工具所在的机器和kundb shard instance所在的节点需要安装xtrabackup |
| 测试步骤 | 1、准备数据,,创建两张表,并导入10w行数据,创建视图,存储过程,触发器  2.配置KunDB的备份还原工具KunDR，执行完全备份操作.具体操作方法见备注  3、备份好之后，随机删除数据库中的数据,制造数据丢失的灾难  4.用步骤1中生成的备份文件进行完全备份还原.具体操作方法见备注  5.检查还原后库中的数据正确性 |
| 预期结果 | 1、备份成功  2、还原成功，数据正确 |
| 测试脚本 | --创建两张表,并导入数据  create table iback\_tbl1(  codeid bigint,  pad bigint,  name varchar(50),  birth datetime,  primary key(codeid)  ) ENGINE=InnoDB partition by hash(codeid) using binary\_md5;  create table iback\_tbl2(  codeid bigint,  pad bigint,  name varchar(50),  birth datetime,  primary key(codeid)  ) ENGINE=InnoDB partition by hash(codeid) using binary\_md5;  数据生成脚本:  cat load.sh  #!/bin/bash  i=1  if [ -e /tmp/load.sql ];then  rm -rf /tmp/load.sql  fi  while [ $i -le 100000 ]  do  echo "insert into iback\_tbl1 values($i,1,'NA$i','2020-01-01');" >> /tmp/load.sql  echo "insert into iback\_tbl2 values($i,1,'NA$i','2020-01-01');" >> /tmp/load.sql  let i++  done  执行bash load.sh生成导数sql文件load,sql  进入kundb对应的数据库,执行source命令进行导数:  Source /tmp/load.sql;  --创建视图  见3.6.1创建视图对象功能测试脚本  --创建存储过程  见3.12.1创建存储过程功能测试脚本  --创建触发器  见3.13.1创建触发器功能测试脚本  --检查还原后的数据正确性  检查视图,存储过程,触发器:  Select \* from view\_test;  Show create procedure NewProc;  Show create trigger newtrigger;  检查两张表的数据:  --iback\_tbl1:  select count(\*) from iback\_tbl1;  select min(codeid) from iback\_tbl1;  select max(codeid) from iback\_tbl1;  --iback\_tbl2:  select count(distinct pad) from iback\_tbl2;  select count(pad) from iback\_tbl2 where pad>1;  select count(pad) from iback\_tbl2 where pad=1; |
| 实际结果 | 1. 备份成功，结果如下：   选区_739   1. 对象还原成功，数据检查正确，结果如下：   2020-07-09 11-13-51屏幕截图  2020-07-09 11-14-23屏幕截图  webwxgetmsgimg (2) |
| 备注 | 1、KunDR完全备份还原的详细操作见附录3  2.完全备份还原注意事项:  2.1 只能冷备,备份过程中,kundb不能进行在线业务,并确保kundb主从同步一致  2.2 物理备份还原时,kundb必须关闭 |

## 增量备份还原

### 验证KunDB系统增量备份还原功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统支持两个数据库之间的数据同步功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有5个shard分区，且分布在不同的节点上。  6.运行备份还原工具所在的机器和kundb shard instance所在的节点需要装有rsync工具  7.运行备份还原工具所在的机器和kundb shard instance所在的节点需要ssh免秘钥登录  8.运行备份还原工具所在的机器和kundb shard instance所在的节点需要安装xtrabackup |
| 测试步骤 | 1、准备数据,配置KunDB的备份还原工具KunDR，执行完全备份操作.具体操作方法见5.7.1完全备份部分  2、完全备份好之后，进入数据库,进行数据增量积累:创建新表,对其中三张表分别进行crud操作,修改已有视图等  3.执行物理增量备份操作.具体操作方法见备注  4.物理增量备份好之后,随机删除数据库中的数据,制造数据丢失的灾难  5.用步骤1中产生的完全备份和步骤3中产生的物理增量备份进行备份还原.具体操作方法见备注  7.检查还原后库中的数据正确性 |
| 预期结果 | 1. 完全备份成功 2. 增量备份成功   3、用完全备份和增量备份还原成功,数据正确 |
| 测试脚本 | --原始数据准备  见完全备份部分  --新建表:  create table iback\_tbl3(  codeid bigint,  pad bigint,  name varchar(50),  birth datetime,  primary key(codeid)  ) ENGINE=InnoDB partition by hash(codeid) using binary\_md5;  --增量数据积累  对三张表(iback\_tbl1,iback\_tbl2,iback\_tbl3)crud操作sql的生成脚本:  cat crud.sh  #!/bin/bash  if [ -e /tmp/crud.sql ];then  rm -rf /tmp/crud.sql  fi  n=1  while [ $n -le 50000 ]  do  echo "insert into iback\_tbl3 values($n,2,'NA$n','2020-03-03');" >> /tmp/crud.sql  echo "update iback\_tbl2 set pad=$n where codeid=$n;" >> /tmp/crud.sql  echo "delete from iback\_tbl1 where codeid=$n;" >> /tmp/crud.sql  let n++  done  执行bash crud.sql生成对三张表crud操作的sql文件crud.sql  进入kundb对应的数据库,执行source命令对表进行crud:  Source crud.sql;  --修改视图:  alter view view\_test as select id, zjhm, name from tb\_test1;  检查还原后数据的正确性:  --检查视图,存储过程,触发器  Select \* from view\_test;  Show create procedure NewProc;  Show create trigger newtrigger;  --检查三张表的数据:  --iback\_tbl1:  select count(\*) from iback\_tbl1;  select min(codeid) from iback\_tbl1;  select max(codeid) from iback\_tbl1;  select count(\*) from iback\_tbl1 where codeid<=50000;  select count(\*) from iback\_tbl1 where codeid>50000;  --iback\_tbl2:  select count(distinct pad) from iback\_tbl2;  select count(pad) from iback\_tbl2 where pad>1;  select count(pad) from iback\_tbl2 where pad=1;  --iback\_tbl3:  select count(codeid) from iback\_tbl3;  select min(codeid) from iback\_tbl3;  select max(codeid) from iback\_tbl3; |
| 实际结果 | 1.完全备份和增量备份成功,结果如下:  选区_742  选区_744  2.对象还原成功，数据检查正确，结果如下：  2020-07-09 10-56-38屏幕截图  2020-07-09 10-56-54屏幕截图 |
| 备注 | 1、KunDR完全备份还原的详细操作见附录3  2.完全备份还原注意事项:  2.1 只能冷备,备份过程中,kundb不能进行在线业务,并确保kundb主从同步一致  2.2 物理备份还原时,kundb必须关闭  3.批量对表进行导数与crud操作的脚本,详见附件6 |

## 数据同步

### 验证KunDB系统支持两个数据库之间的数据同步功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统支持两个数据库之间的数据同步功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、两套KunDB产品在服务器上已经被成功安装，且运行正常。每套KunDB各有5个shard分区，且分布在不同的节点上。  6、TDT产品在服务器上已经被成功安装，且运行正常 |
| 测试步骤 | 1、通过TDT工具，创建源端数据库与目标端数据库的数据同步任务，同步task的个数与源端数据库的shard个数一样，并上线任务，具体操作见备注描述  2、在源端数据库执行建表，增删改查等操作．观察目标端数据库的同步结果，具体脚本见测试脚本 |
| 预期结果 | 1、数据同步任务创建和上线成功  2、源端数据库执行的操作都能正确同步到目标端数据库 |
| 测试脚本 | *--源端执行的DDL、DML等SQL脚本*  *drop table if exists messages3;*  *CREATE TABLE messages3 (*  *page BIGINT(20) UNSIGNED,*  *time\_created\_ns BIGINT(20) UNSIGNED,*  *message VARCHAR(100),*  *PRIMARY KEY (page, time\_created\_ns)*  *)ENGINE=InnoDB partition by hash(page) using binary\_md5;*  *insert into messages3 values(1,2,'1');*  *insert into messages3 values(2,3,'1');*  *select \* from messages3;*  *update messages3 set message='2' where page=2;*  *select \* from messages3;*  *delete from messages3 where page=1;*  *select \* from messages3;*  *alter table messages3 add column page1 varchar(20) after page;*  *select \* from messages3;*  *insert into messages3 values(3,'3',3,'1');*  *select \* from messages3;*  *alter table messages3 drop column page1;*  *alter table messages3 modify page char(10);*  *alter table messages3 change message information varchar(100);*  *desc messages3;*  *insert into messages3 values('4',5,'10');*  *alter table messages3 rename to message1;*  *select \* from message1;*  *drop table if exists message1;* |
| 实际结果 | 1、数据同步任务创建和上线成功  2、源端数据库kundb1执行的操作都能正确同步到目标端数据库syncdb，结果如下：  选区_547  选区_548 |
| 备注 | 数据库数据同步任务创建操作方法，详见附件7 |

## 恢复点目标RPO

### 验证KunDB故障恢复后业务可正常进行功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB故障恢复后业务可正常进行功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有4个shard分区，且分布在不同的节点上。  6、Loadrunner与web程序均已调试成功。 |
| 测试步骤 | 1、通过loadrunner实现100用户并发对创建表单的插入操作  2、测试中平稳运行5分钟后，进行一个工作节点的断网操作，观察LoadRunner执行情况  3、记录通过的transaction数、数据库该表的所有记录数，并记录并发用户数、 thinktime数、测试运行设置时长，并查询数据库记录时间戳不连续的时间间隔 |
| 预期结果 | 1、LoadRunner高并发插入操作执行成功，断网后，KunDB发生主从切换  2、LoadRunner会报插入操错误，但是程序不会退出  3、数据库记录时间戳不连续的时间间隔和组从切换时间能够匹配 |
| 测试脚本 |  |
| 实际结果 | 1、LoadRunner高并发插入操作执行成功，结果如下：  2、应用程序会报错误，但是程序不会退出，结果如下：    3、Loadrunner的运行结果情况如下：    transaction数:530728  数据库该表的所有记录数: 530751  并记录并发用户数:100  thinktime数:0  测试运行设置时长:10mins  数据库记录时间戳不连续的时间间隔:100s |
| 备注 | Loadrunner与webService程序及详细使用方法见附件5 |

# 信息安全性

## 身份鉴别

### 验证KunDB系统用户身份进行鉴别功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统用户身份进行鉴别功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB，登录命令见测试脚本描述 |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功  3、通过用户user\_test1登录KunDB成功 |
| 测试脚本 | *--建表SQL语句*  *create user if not exists user\_test1;*  *--用户登录命令*  *mysql -h172.16.130.123 -P15307 -uuser\_test1 -p123 --ssl-ca=/etc/kundb1/conf/ca-cert.pem --ssl-cert=/etc/kundb1/conf/kungate-client-cert.pem --ssl-key=/etc/kundb1/conf/kungate-client-key.pem* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，结果如下：    3、通过用户user\_test1登录KunDB成功，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB系统用户身份鉴别信息加密存储功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统用户身份鉴别信息加密存储功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，SQL语句见测试脚本描述 3. 通过root用户登录人一个master节点，查询用户user\_test1信息，登录命令和查询脚本见测试脚本描述 |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功  3、通过用户user\_test1登录KunDB成功，用户user\_test1信息查询成功，且为加密形式 |
| 测试脚本 | *--建表SQL语句*  *create user if not exists user\_test1;*  *--用户登录KunDB命令*  *mysql -h172.16.130.123 -P15307 -uuser\_test1 -p123 --ssl-ca=/etc/kundb1/conf/ca-cert.pem --ssl-cert=/etc/kundb1/conf/kungate-client-cert.pem --ssl-key=/etc/kundb1/conf/kungate-client-key.pem*  *--用户登录master命令*  *mysql -h 172.16.130.125 -P 14100 -uroot -pTranswarp! -A*  *--查询用户信息SQL*  *select authentication\_string from mysql.user where User='user\_test1';* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，结果如下：    3、通过用户user\_test1登录KunDB成功，用户user\_test1信息查询成功，且为加密形式，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB系统用户身份鉴别失败次数告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统用户身份进行鉴别功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB，输入错误密码，登录11次，进入Aquila界面，命令见测试脚本描述 4. 进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功  3、11次认证错误命令执行成功  4、存在认证失败告警信息 |
| 测试脚本 | *--建表SQL语句*  *create user if not exists user\_test1;*  *--用户登录命令*  *mysql -h172.16.130.123 -P15307 -uuser\_test1 -p1234 --ssl-ca=/etc/kundb1/conf/ca-cert.pem --ssl-cert=/etc/kundb1/conf/kungate-client-cert.pem --ssl-key=/etc/kundb1/conf/kungate-client-key.pem* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，结果如下：    3、11次认证错误命令执行成功，结果如下：    4、存在认证失败告警信息，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB系统定义用户登录失败最大次数

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统定义用户登录失败最大次数 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 在Manager中修改KunDB默认参数kundb.ldap\_cache\_max\_size值为0，关闭认证kungate认证缓存，配置服务并且重启kungate角色 2. 在Guardian设置页面的密码策略子页面中设置“发生以下情况锁定账户”的失败登录次数为5，重启Guardian的ApacheDS角色 3. 在Guardian中新建用户 user\_test2 4. 在KunDB中新建用户 user\_test2 5. 使用user\_test2的正确密码验证能够成功登录KunDB 6. 使用user\_test2的错误密码连续5次登录失败KunDB 7. 使用user\_test2的正确密码验证能否成功登录KunDB 8. 在Guardian中解锁用户，并且使用user\_test2的正确密码验证能够成功登录KunDB |
| 预期结果 | 1、Gudian Server 用户user\_test2创建成功  2、数据库用户user\_test2创建成功  3、使用user\_test2的错误密码登录失败5次之后，用户被锁定，不能登录KunDB，解锁后可成功登录 |
| 测试脚本 | *--创建用户SQL语句*  *create user if not exists user\_test2;*  *--用户登录KunDB命令*  *mysql -h172.16.130.123 -P15307 -uuser\_test2 -p123 --ssl-ca=/etc/kundb1/conf/ca-cert.pem --ssl-cert=/etc/kundb1/conf/kungate-client-cert.pem --ssl-key=/etc/kundb1/conf/kungate-client-key.pem* |
| 实际结果 | 1、Gudian Server 用户user\_test2创建成功  2、数据库用户user\_test2创建成功  3、通过用户user\_test2登录KunDB成功  4、使用user\_test2的错误密码登录5次之后，用户被锁定，登录系统会返回错误       1. 解锁后，user\_test2可以成功登录kundb |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 访问控制

### 验证KunDB支持强制访问控制功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持强制访问控制功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕  4、KunDB中已存在数据库dbtest1，且强制访问控制开关已经打开，具体打开方式见备注  5、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test2，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test2，SQL语句见测试脚本描述 3. 通过用户vt\_app登录KunDB，创建表t1,SQL语句见测试脚本描述 4. 通过用户vt\_app登录KunDB，给用户user\_test2赋予全局的所有DAC权限(防止DAC干扰)，SQL语句见测试脚本描述 5. 分别通过user\_test2登录KUNDB，查询表t1的数据 6. 通过超级用户vt\_app登录KunDB,给用户user\_test2赋予表t1的c1列的权限 7. 通过用户user\_test2登录KunDB,查询表t1的c1和c2字段 |
| 预期结果 | 1、Gudian Server 用户user\_test2创建成功  2、数据库用户user\_test2创建成功  3、表t1创建成功  4、用户user\_test2的DAC权限成功赋予  5、user\_test2用户查询t1表的数据被拒绝  6、user\_test2MAC权限赋值成功  7、用户user\_test2查询表t1的c1字段查询成功，c2字段查询被拒绝 |
| *测试脚本* | *--创建用户SQL语句*  *create user user\_test2;*  *--创建表t1 SQL语句*  *create table t1(c1 int, c2 int);*  *--用户user\_test2赋DAC权限SQL语句*  *grant all privileges on \*.\* to user\_test2;*  *--用户user\_test2赋MAC权限SQL语句*  *grant level 1 to user user\_test2;*  *grant label l1 on table dbtest1.t1;*  *grant label l0 on column dbtest1.t1.c1;*  *grant level 1 to label l1;*  *grant level 0 to label l0;*  *--删除用户SQL语句*  *drop user if exists user\_test1;* |
| 实际结果 | 1、Gudian Server 用户*user\_test2*创建成功  2、数据库用户*user\_test2*创建成功  3、表t1创建成功  4、用户*user\_test2*的DAC权限成功赋予  5、*user\_test2*用户查询t1表的数据被拒绝，结果如下：    6、u1的MAC权限赋值成功，结果如下：    7、用户user\_test2查询表t1的c1字段查询成功，c2字段查询被拒绝，结果如下： |
| 备注 | 1. 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 2. 强制访问控制开关，通过Manager界面修改，参数为enable\_mandatory\_access\_ctrl，修改值为true,修改后重新配置服务，并启动 |

### 验证KunDB支持用户数据库对象级访问控制功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持用户数据库对象级权限功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、KunDB中已存在数据库dbtest1  5、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB，并在数据库dbtest1中创建表*tb\_test1，SQL语句见测试脚本描述* 4. 通过超级用户vt\_app登录KunDB,给用户user\_test1赋予数据库dbtest1所有的操作权限，SQL语句见测试脚本描述 5. 再次通过用户user\_test1登录KunDB，并在数据库dbtest1中创建表*tb\_test1，SQL语句见测试脚本描述* |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功  3、通过用户user\_test1登录KunDB成功，建表失败，报没有权限错误  4、给用户user\_test1赋予数据库dbtest1所有的权限操作成功  5、通过用户user\_test1登录KunDB成功，建表成功 |
| 测试脚本 | *--创建用户SQL语句*  *create user user\_test1;*  *--删除用户SQL语句*  *drop user if exists user\_test1;*  *--授权用户SQL语句*  *grant create,drop,alter,select,insert,update,delete on dbtest1.\* to user\_test1;*  *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，结果如下：    3、通过用户user\_test1登录KunDB成功，建表失败，报没有权限错误，结果如下：    4、给用户user\_test1赋予数据库dbtest1所有的权限操作成功，结果如下：  5、通过用户user\_test1登录KunDB成功，建表成功，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持用户视图对象级访问控制功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持用户视图对象级访问控制功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、KunDB中已存在数据库dbtest1，且数据库存在表tb\_test1和视图view\_test  5、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，并创建视图view\_test,SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB，并在数据库dbtest1中查询视图view\_test*，SQL语句见测试脚本描述* 4. 通过超级用户vt\_app登录KunDB,给用户user\_test1赋予视图view\_test所有的操作权限，SQL语句见测试脚本描述 5. 再次通过用户user\_test1登录KunDB，并在数据库dbtest1中查询视图view\_test*，SQL语句见测试脚本描述* |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功,视图view\_test创建成功  3、通过用户user\_test1登录KunDB成功，查询视图失败，报没有权限错误  4、给用户user\_test1赋予视图view\_test所有的权限操作成功  5、通过用户user\_test1登录KunDB成功，查询视图成功 |
| 测试脚本 | *--创建用户SQL语句*  *create user user\_test1;*  *--删除用户SQL语句*  *drop user if exists user\_test1;*  *--授权用户视图权限SQL语句*  *grant all privileges on dbtest1.\* to user\_test1;*  *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *--创建视图SQL语句*  *create or replace view dbtest1.view\_test as select \* from dbtest1.tb\_test1;*  *--查询视图SQL语句*  *select \* from view\_test;* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，视图view\_test创建成功，结果如下：      3、通过用户user\_test1登录KunDB成功，查询视图失败，报没有权限错误，结果如下：    4、给用户user\_test1赋予查询视图所有的权限操作成功，结果如下：    5、通过用户user\_test1登录KunDB成功，查询视图成功，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB支持用户表对象级访问控制功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB支持用户表对象级访问控制功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、KunDB中已存在数据库dbtest1，且数据库存在表*tb\_test1*  5、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB，并在数据库dbtest1中表*tb\_test1插入记录，SQL语句见测试脚本描述* 4. 通过超级用户vt\_app登录KunDB,给用户user\_test1赋予表*tb\_test1*所有的操作权限，SQL语句见测试脚本描述 5. 再次通过用户user\_test1登录KunDB，并在数据库dbtest1中表*tb\_test1插入记录，SQL语句见测试脚本描述* |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功  3、通过用户user\_test1登录KunDB成功，数据插入失败，报没有权限错误  4、给用户user\_test1赋予表*tb\_test1*所有的权限操作成功  5、通过用户user\_test1登录KunDB成功，数据插入成功 |
| 测试脚本 | *--创建用户SQL语句*  *create user user\_test1;*  *--删除用户SQL语句*  *drop user if exists user\_test1;*  *--授权用户SQL语句*  *grant create,drop,alter,select,insert,update,delete on dbtest1.tb\_test1 to user\_test1;*  *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *--插入记录SQL语句*  *insert into tb\_test1(a,b,c) values(1,2,'test');* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，结果如下：    3、通过用户user\_test1登录KunDB成功，数据插入失败，报没有权限错误，结果如下：    4、给用户user\_test1赋予表*tb\_test1*所有的权限操作成功，结果如下：    5、通过用户user\_test1登录KunDB成功，数据插入成功，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 安全审计

### 验证KunDB系统对用户身份认证操作的安全审计功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统对用户身份认证操作的安全审计功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB，登录命令见测试脚本描述 4. 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB->KunDB 相关审计->用户认证审计仪表盘，查看用户user\_test1认证操作的审计日志 |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功  3、通过用户user\_test1登录KunDB成功  4、仪表盘可正确显示user\_test1认证操作的审计日志信息 |
| 测试脚本 | *--建表SQL语句*  *create user if not exists user\_test1;*  *--用户登录命令*  *mysql -h172.16.130.123 -P15307 -uuser\_test1 -p123 --ssl-ca=/etc/kundb1/conf/ca-cert.pem --ssl-cert=/etc/kundb1/conf/kungate-client-cert.pem --ssl-key=/etc/kundb1/conf/kungate-client-key.pem* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，结果如下：    3、通过用户user\_test1登录KunDB成功，结果如下：    4、仪表盘可正确显示user\_test1认证操作的审计日志信息，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB系统对SQL操作的安全审计功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统对SQL操作的安全审计功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Waterdrop工具，在数据库dbtest1中执行一些列SQL语句，SQL语句见测试脚本中描述 2. 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB->KunDB 相关审计->SQL审计报表，查看SQL执行的审计日志 |
| 预期结果 | 1、SQL语句执行成功  2、仪表盘可正确显示SQL操作的审计日志信息 |
| 测试脚本 | *--SQL语句*  *drop table if exists tb\_test1;*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *insert into tb\_test1(a,b,c) values(1,2,'test');*  *update tb\_test1 set c='test1' where a=1;*  *select \* from tb\_test1;*  *delete from tb\_test1;*  *drop user if exists user\_test1;*  *create user if not exists user\_test1;*  *grant create,drop,alter,select,insert,update,delete on dbtest1.\* to user\_test1;* |
| 实际结果 | 1、SQL语句执行成功，结果如下：    2、仪表盘可正确显示SQL操作的审计日志信息，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB系统对配置操作的安全审计功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统对配置操作的安全审计功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1、通过Transwarp Manager修改KunGate的audit\_filter的参数  2、修改完成后，通过Transwarp Manager重新配置KunDB的服务  3、进入Aquila界面，打仪表盘->仪表盘管理->KUNDB->KunDB 相关审计->SQL审计报表，查看配置操作的审计日志 |
| 预期结果 | 1、KunGate的audit\_filter可正确修改成功  2、KunDB重新配置服务成功  3、仪表盘可正确显示配置操作的审计日志信息 |
| 测试脚本 |  |
| 实际结果 | 1、KunGate的audit\_filter参数配置修改成功，结果如下：    2、KunDB重新配置服务成功，结果如下：    3、仪表盘可正确显示配置操作的审计日志信息，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证KunDB安全审计实时报警功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统对配置操作的安全审计功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 进入Aquila告警页面，查看是否有“KunDB越权操作”告警，如果有，先解除告警。 2. 使用用户user\_test1对没有update权限的表执行update操作 3. 观测Aquila的告警页面中是否有“KunDB越权操作”告警产生 |
| 预期结果 | 1. 解除“KunDB越权操作”告警成功 2. user\_test1执行update操作失败，报权限问题 3. 观测到Aquila的告警页面中，产生新的“KunDB越权操作” 告警，并且触发时间为越权执行update操作的时间 |
| 测试脚本 |  |
| 实际结果 | 1. 解除“KunDB越权操作”告警成功 2. user\_test1执行update操作失败，报权限问题      1. 观测到Aquila的告警页面中，产生新的“KunDB越权操作” 告警，并且触发时间为越权执行update操作的时间 |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

# 易用性

# 可维护性

## 错误检测与提示

### 验证对用户错误操作/命令的检查和提示功能

|  |  |
| --- | --- |
| 测试目的 | 验证对用户错误操作/命令的检查和提升功能 。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema dbtest; |
| 测试步骤 | 通过MySQL命令链接KunDBServer，登录方法见备注中描述  2、在Schema dbtest中执行一条错误的建表SQL命令，SQL语句见测试脚本中描述 |
| 预期结果 | 1、链接成功  2、KunDB返回错误信息，且信息中包含具体原因 |
| 测试脚本 | *--SQL命令*  *create tabel wrong(a int, c int, b varchar(10)) PARTITION by HASH(a);* |
| 实际结果 | 1、链接成功  2、返回的报错信息如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证对非法数据输入的检查和提示功能

|  |  |
| --- | --- |
| 测试目的 | 验证对非法数据输入的检查和提示功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema dbtest，且该Schema中存在表test\_1，test\_1表中存在整形列int\_col |
| 测试步骤 | 1、通过MySQL命令链接KunDBServer，登录方法见备注中描述  2、向test\_1中插入一条数据，其中int\_col列的值为char型 |
| 预期结果 | 1、链接成功  2、KunDB返回错误信息，且信息中包含具体原因 |
| 测试脚本 | *--创建Schema*  *create tabel wrong(a int, c int, b varchar(10)) PARTITION by HASH(a);*  *--建表语句*  *create table test\_1(int\_col int, char\_col int, varchar\_col varchar(10)) partition by hash(int\_col) using binary\_md5;*  *--INSERT语句*  *insert into test\_1 values('a','b','c');* |
| 实际结果 | 1、链接成功  2、返回的报错信息如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

## 日志信息

### 验证对KunDB系统记录实时日志功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统记录实时日志功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行  4、KunDB产品在服务器上已经被成功安装，且运行正常 |
| 测试步骤 | 1、 进入Aquila界面，打开日志搜索菜单，查看实时日志信息 |
| 预期结果 | 1、日志搜索菜单可正常实时显示日志信息 |
| 测试脚本 |  |
| 实际结果 | 1、日志搜索菜单可正常实时显示日志信息,结果如下： |
| 备注 |  |

## 性能监控

### 验证对KunDB系统CPU使用率监控功能

|  |  |
| --- | --- |
| 测试目的 | 验证对非法数据输入的检查和提示功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->CLUSTER->CPUs,在Node中任选一个节点。  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、监控该节点的CPU使用率仪表盘被正常打开，正常显示该被选择的节点的CPU各种使用信息。  2、该仪表盘会动态显示TPCC事务操作过程中的CPU使用变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点CPU使用率仪表盘数据如下：    2、执行TPCC过程中CPU使用率仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统内存的监控功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统内存的监控统计功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->CLUSTER->Memory,在Node中任选一个节点。  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、监控该节点的Memory使用率仪表盘被正常打开，正常显示该被选择的节点的Memory各种使用信息。  2、该仪表盘会动态显示TPCC事务操作过程中的Memory使用变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点Memory使用率仪表盘数据如下：    2、执行TPCC过程中Memory使用率仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统磁盘空间使用率的监控功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统磁盘空间使用率的监控统计功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->CLUSTER-Disk,在Node中任选一个节点。  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、监控该节点的Disk使用率仪表盘被正常打开，正常显示该被选择的节点的Disk各种使用信息。  2、该仪表盘会动态显示TPCC事务操作过程中的Disk使用变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点Disk使用率仪表盘数据如下：    2、执行TPCC过程中Disk使用率仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统Computer Node运行内存状态监控功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统Computer Node运行内存状态监控功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB-Computer node性能监控,在节点中任选一个节点,查看运行内存状态仪表盘  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、运行内存状态仪表盘被正常打开，正常显示Golang runtime 累计分配的堆内存变化速率、Golang runtime当前实际分配的堆内存、Golang runtime 使用的总内存 、Golang runtime 垃圾回收时间占比等运行状态。  2、该仪表盘会动态显示Golang runtime 累计分配的堆内存变化速率、Golang runtime当前实际分配的堆内存、Golang runtime 使用的总内存 、Golang runtime 垃圾回收时间占比使用变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点运行内存状态仪表盘数据如下：    2、执行TPCC过程中节点运行内存状态仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统Computer Node Mysql Server运行状态监控功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统Computer Node Mysql Server运行状态监控功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB->Computer node性能监控,在节点中任选一个节点,查看Mysql Server运行状态仪表盘  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、Mysql Server运行状态仪表盘被正常打开，正常显示Mysql QPS  、Mysql表信息、MySQL慢查询和全表扫描、Mysql连接数等运行状态。  2、该仪表盘会动态显示Mysql QPS、Mysql表信息、MySQL慢查询和全表扫描、Mysql连接数使用变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点运行内存状态仪表盘数据如下：    2、执行TPCC过程中节点运行内存状态仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统各服务运行状态监控功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统各服务运行状态监控功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB->KunDB性能概览,在节点中任选一个节点,查看状态概览仪表盘  2、通过后台操作，故障一个KunGate实例，使其状态变为CrashLoopBackOff，具体操作方法见备注 |
| 预期结果 | 1、状态概览仪表盘被正常打开，正常显示该节点上各服务的运行状态，结果为全部正常。  2、状态概览仪表盘KunGate运行状态中显示为部分正常 |
| 测试脚本 |  |
| 实际结果 | 1、初始该节点状态概览仪表盘数据如下：    2、KunGate故障一个实例后，该节点状态概览仪表盘数据如下： |
| 备注 | 故障KunGate一个实例的操作如下：  1、登录集群中任一个Master节点，执行如下命令，查看KunGat服务的deployment资源信息，命令如下：  kubectl get deployment | grep kungate  2、修改该deployment资源的配置信息，修改参数spec->template->spec->containers->args中的值KUNGATE为KUNGATE1，修改完成后保存，具体修改命令为：  kubectl edit deployment kundb-kungate-kundb1 |

### 验证对KunDB系统KunGate运行内存状态监控功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统KunGate运行内存状态监控功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB-Kungate性能监控,在节点中任选一个节点,查看运行内存状态仪表盘  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、运行内存状态仪表盘被正常打开，正常显示Golang runtime 累计分配的堆内存变化速率、Golang runtime当前实际分配的堆内存、Golang runtime 使用的总内存 、Golang runtime 垃圾回收时间占比等运行状态。  2、该仪表盘会动态显示Golang runtime 累计分配的堆内存变化速率、Golang runtime当前实际分配的堆内存、Golang runtime 使用的总内存 、Golang runtime 垃圾回收时间占比使用变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点运行内存状态仪表盘数据如下：    2、执行TPCC过程中节点运行内存状态仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统KunGate服务执行缓存统计功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统KunGate服务执行缓存统计功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB-Kungate性能监控,在节点中任选一个节点,查看服务执行缓存统计仪表盘  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、服务执行缓存统计仪表盘被正常打开，正常显示服务计划缓存命中率统计数据。  2、该仪表盘会动态显示服务执行缓存统计变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点服务执行缓存统计数据仪表盘如下：    2、执行TPCC过程中节点服务执行缓存统计仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统KunGate查询统计功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统KunGate查询统计功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB-Kungate性能监控,在节点中任选一个节点,查看KunGate查询统计仪表盘  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、KunGate查询统计仪表盘被正常打开，正常显示KunGateQPS、KunGate各类查询数量、KunGate各类查询近30分钟增速统计数据。  2、该仪表盘会动态显示KunGateQPS、KunGate各类查询数量、KunGate各类查询近30分钟增速统计数据变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点查询统计数据仪表盘如下：    2、执行TPCC过程中节点查询统计仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统KunGate事务统计功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统KunGate事务统计功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB-Kungate性能监控,在节点中任选一个节点,查看KunGate事务统计仪表盘  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、KunGate事务统计仪表盘被正常打开，正常显示KunGate TPS、KunGate事务总量等数据。  2、该仪表盘会动态显示KunGate TPS、KunGate事务总量、等数据变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点事务统计数据仪表盘如下：    2、执行TPCC过程中节点事务统计仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统KunGate会话统计功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统KunGate会话统计功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB-Kungate性能监控,在节点中任选一个节点,查看KunGate会话统计仪表盘  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、KunGate会话统计仪表盘被正常打开，正常显示KunGate当前活跃的链接数、KunGate累计链接数的变化率等数据。  2、该仪表盘会动态显示KunGate当前活跃的链接数、KunGate累计链接数的变化率等数据变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点会话统计数据仪表盘如下：    2、执行TPCC过程中节点会话统计仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统Kuntablet运行内存状态监控功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统Kuntablet运行内存状态监控功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB-Kuntablet性能监控,在节点中任选一个节点,查看运行内存状态仪表盘  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、运行内存状态仪表盘被正常打开，正常显示Golang runtime 累计分配的堆内存变化速率、Golang runtime当前实际分配的堆内存、Golang runtime 使用的总内存 、Golang runtime 垃圾回收时间占比等运行状态。  2、该仪表盘会动态显示Golang runtime 累计分配的堆内存变化速率、Golang runtime当前实际分配的堆内存、Golang runtime 使用的总内存 、Golang runtime 垃圾回收时间占比使用变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点运行内存状态仪表盘数据如下：    2、执行TPCC过程中节点运行内存状态仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统Kuntablet服务执行缓存统计功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统Kuntablet服务执行缓存统计功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB-Kuntablet性能监控,在节点中任选一个节点,查看服务执行缓存统计仪表盘  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、服务执行缓存统计仪表盘被正常打开，正常显示服务计划缓存命中率统计数据。  2、该仪表盘会动态显示服务执行缓存统计变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点服务执行缓存统计数据仪表盘如下：    2、执行TPCC过程中节点服务执行缓存统计仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统Kuntablet查询统计功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统Kuntablet查询统计功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见<>中备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB-Kuntablet性能监控,在节点中任选一个节点,查看KunGate查询统计仪表盘  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、KunGate查询统计仪表盘被正常打开，正常显示KunTablet QPS、KunTablet查询数量统计等数据。  2、该仪表盘会动态显示KunTablet QPS、KunTablet查询数量统计等数据变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点查询统计数据仪表盘如下：    2、执行TPCC过程中节点查询统计仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统KunTablet MySQL Server运行状态监控功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统KunTablet MySQL Server运行状态监控功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、 进入Aquila界面，打仪表盘->仪表盘管理->KUNDB-Kuntablet性能监控,在节点中任选一个节点,查看MySql Server运行状态仪表盘  2、使用BenchmarkSQL工具，进行TPCC事务操作 |
| 预期结果 | 1、MySql Server运行状态仪表盘被正常打开，正常显示Mysql QPS、Mysql表信息、MySQL慢查询和全表扫描、Mysql连接数等数据。  2、该仪表盘会动态显示Mysql QPS、Mysql表信息、MySQL慢查询和全表扫描、Mysql连接数等数据变化情况 |
| 测试脚本 |  |
| 实际结果 | 1、初始节点MySql Server运行状态数据仪表盘如下：    2、执行TPCC过程中节点MySql Server运行状态仪表盘数据如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

## 运维告警功能

### 验证对KunDB系统节点CPU使用过大告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统节点CPU使用过大告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 登录系统任一服务器，运行CPU占用工具，使该节点的CPU占用率超过70%，详细操作见备注描述 2. 进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、CPU占用工具运行成功，节点内从使用率超过70%  2、存在节点CPU使用过大告警 |
| 测试脚本 |  |
| 实际结果 | 1、CPU占用工具运行成功，节点内从使用率超过70%，结果如下：    2、存在CPU内存使用过大告警，结果如下： |
| 备注 | CPU、内存、磁盘占用工具使用方法见附件4 |

### 验证对KunDB系统节点内存使用过大告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统节点内存使用过大告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 登录系统任一服务器，运行内存占用工具，使该节点的内存占用率超过70%，详细操作见备注描述 2. 进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、内存占用工具运行成功，节点内从使用率超过70%  2、存在节点内存使用过大告警 |
| 测试脚本 |  |
| 实际结果 | 1、内存占用工具运行成功，节点内从使用率超过70%，结果如下：    2、存在节点内存使用过大告警，结果如下： |
| 备注 | CPU、内存、磁盘占用工具使用方法见附件4 |

### 验证对KunDB系统节点磁盘剩余空间过低告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统节点磁盘剩余空间过低告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 登录系统任一服务器，运行磁盘占用工具，使该节点任一个磁盘分区的剩余容量低于30%，详细操作见备注描述 2. 进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、磁盘占用工具运行成功，选中磁盘分区的剩余容量低于30%  2、存在节点磁盘分区剩余空间过低告警 |
| 测试脚本 |  |
| 实际结果 | 1、磁盘占用工具运行成功，节点选中磁盘分区的剩余容量低于30%，结果如下：    2、存在节点磁盘分区剩余空间过低告警，结果如下： |
| 备注 | CPU、内存、磁盘占用工具使用方法见附件4 |

### 验证对KunDB系统节点失联告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统节点失联告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 关闭节点任一个服务器 2. 进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、选中服务器关闭成功  2、存在节点失联告警 |
| 测试脚本 |  |
| 实际结果 | 1、选中服务器关闭成功  2、存在节点失联告警，结果如下： |
| 备注 | CPU、内存、磁盘占用工具使用方法见附件4 |

### 验证对KunDB系统30分钟内慢查询次数过多告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统30分钟内慢查询次数过多告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、使用BenchmarkSQL工具，进行TPCC事务操作，构造KunGate 30分钟内慢查询次数>10的场景，具体操作见附件  2、进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、TPCC事务操作成功  2、存在KunGate 慢查询过多告警 |
| 测试脚本 |  |
| 实际结果 | 1、TPCC事务操作成功  2、存在KunGate 连接数过多告警，结果如下： |
| 备注 | 1. 装载数据和TPCC建表、建索引等方法详见附录1，附录2 2. 慢查询是指Query语句执行时间超过10s的查询 |

### 验证对KunDB系统ComputeNode CPU 占用过多告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统ComputeNode CPU占用过多告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1、登录系统任一服务器，进入一个ComputeNode实例的POD， 运行CPU占用工具，模拟该实例占用该节点的CPU超过10%，详细操作见备注描述  2、进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、CPU占用工具运行成功，ComputeNode 实例占用该节点的CPU超过10%  2、存在 ComputeNode CPU占用过多告警 |
| 测试脚本 |  |
| 实际结果 | 1、CPU占用工具运行成功，ComputeNode 实例占用该节点的CPU超过10%  2、存在 ComputeNode CPU占用过多告警，结果如下： |
| 备注 | CPU、内存、磁盘占用工具使用方法见附件4 |

### 验证对KunDB系统ComputeNode 内存过大告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统ComputeNode 内存过大告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1、登录系统任一服务器，进入一个ComputeNode实例的POD， 运行内存占用工具，模拟该实例占用该节点的内存超过10%，详细操作见备注描述  2、进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、内存占用工具运行成功，ComputeNode 实例占用该节点的内存超过10%  2、存在 ComputeNode 内存占用过多告警 |
| 测试脚本 |  |
| 实际结果 | 1、CPU占用工具运行成功，ComputeNode 实例占用该节点的内存超过10%  2、存在 ComputeNode 内存过大告警，结果如下： |
| 备注 | CPU、内存、磁盘占用工具使用方法见附件4 |

### 验证对KunDB系统Kundb 认证失败告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统用户身份进行鉴别功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB，输入错误密码，登录11次，进入Aquila界面，命令见测试脚本描述 4. 进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功  3、11次认证错误命令执行成功  4、存在认证失败告警信息 |
| 测试脚本 | *--建表SQL语句*  *create user if not exists user\_test1;*  *--用户登录命令*  *mysql -h172.16.130.123 -P15307 -uuser\_test1 -p1234 --ssl-ca=/etc/kundb1/conf/ca-cert.pem --ssl-cert=/etc/kundb1/conf/kungate-client-cert.pem --ssl-key=/etc/kundb1/conf/kungate-client-key.pem* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，结果如下：    3、11次认证错误命令执行成功，结果如下：    4、存在认证失败告警信息，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证对KunDB系统越权操作告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统支持越权操作告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、KunDB中已存在数据库dbtest1  5、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过Gudian Server服务页面，创建用户user\_test1，详细操作见备注描述 2. 通过超级用户vt\_app登录KunDB,创建用户user\_test1，SQL语句见测试脚本描述 3. 通过用户user\_test1登录KunDB，并在数据库dbtest1中创建表*tb\_test1，SQL语句见测试脚本描述* 4. 进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功  3、通过用户user\_test1登录KunDB成功，建表失败，报没有权限错误  4、存在越权操作的告警信息 |
| 测试脚本 | *--创建用户SQL语句*  *create user user\_test1;*  *--删除用户SQL语句*  *drop user if exists user\_test1;*  *--授权用户SQL语句*  *grant create,drop,alter,select,insert,update,delete on dbtest1.\* to user\_test1;*  *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;* |
| 实际结果 | 1、Gudian Server 用户user\_test1创建成功  2、数据库用户user\_test1创建成功，结果如下：    3、通过用户user\_test1登录KunDB成功，建表失败，报没有权限错误，结果如下：    4、存在越权操作的告警信息，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证对KunDB系统锁超时警功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统锁超时功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、KunDB中已存在数据库dbtest1，且其中个存在表tb\_test1，建表语句见测试脚本  5、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1. 通过通过Waterdrop工具，向表tb\_test1插入一条记录，SQL见测试脚本描述 2. 通过一个会话，执行事务，更新表tb\_test1插入的记录，不提交，SQL见测试脚本描述 3. 通过其他多个会话，执行更新表tb\_test1该条记录的操作，超过10次 4. 进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、表tb\_test1记录插入成功  2、操作执行成功  3、所有会话的操作均超时  4、存在KunDB 锁超时的告警信息 |
| 测试脚本 | *--建表SQL语句*  *CREATE TABLE tb\_test1(*  *a int not null, b int, c char(8)) ENGINE=InnoDB partition by HASH(a) using binary\_md5;*  *--插入记录SQL语句*  *insert into tb\_test1(a,b,c) values(1,2,'test');*  *--更新记录SQL语句*  *update tb\_test1 set c='test1' where a=1;* |
| 实际结果 | 1、表tb\_test1记录插入成功  2、操作执行成功，结果如下：  3、所有会话的操作均超时，结果如下：      4、存在KunDB 锁超时的告警信息，结果如下： |
| 备注 | 在使用Waterdrop客户端连接到KunDB Server之前，需要通过Guadian添加和配置KunDB的访问用户，具体操作见安装文档《KunDBManual.docx》中描述 |

### 验证对KunDB系统相关服务不可用告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统各服务运行状态监控功能。 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常 |
| 测试步骤 | 1、通过后台操作，故障一个KunGate实例，使其状态变为CrashLoopBackOff，具体操作方法见备注  2、进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、KunGate实例故障成功  2、存在相关服务不可用告警 |
| 测试脚本 |  |
| 实际结果 | 1、KunGate实例故障成功，结果如下：    2、存在相关服务不可用告警，结果如下： |
| 备注 | 故障KunGate一个实例的操作如下：  1、登录集群中任一个Master节点，执行如下命令，查看KunGat服务的deployment资源信息，命令如下：  kubectl get deployment | grep kungate  2、修改该deployment资源的配置信息，修改参数spec->template->spec->containers->args中的值KUNGATE为KUNGATE1，修改完成后保存，具体修改命令为：  kubectl edit deployment kundb-kungate-kundb1 |

### 验证对KunDB系统KunGate CPU占用过多告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统KunGate CPU占用过多告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1、登录系统任一服务器，进入一个KunGate 实例的POD， 运行CPU占用工具，模拟该实例占用该节点的CPU超过10%，详细操作见备注描述  2、进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、CPU占用工具运行成功，KunGate 实例占用该节点的CPU超过10%  2、存在 KunGate CPU占用过多告警 |
| 测试脚本 |  |
| 实际结果 | 1、CPU占用工具运行成功，KunGate 实例占用该节点的CPU超过10%  2、存在 KunGate CPU占用过多告警，结果如下： |
| 备注 | CPU、内存、磁盘占用工具使用方法见附件4 |

### 验证对KunDB系统KunGate 内存过大告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统KunGate 内存过大告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1、登录系统任一服务器，进入一个KunGate 实例的POD， 运行内存占用工具，模拟该实例占用该节点的内存超过10%，详细操作见备注描述  2、进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、CPU占用工具运行成功，KunGate 实例占用该节点的内存超过10%  2、存在 KunGate 内存过大告警 |
| 测试脚本 |  |
| 实际结果 | 1、内存占用工具运行成功，KunGate 实例占用该节点的内存超过10%  2、存在 KunGate 内存过大告警，结果如下： |
| 备注 | CPU、内存、磁盘占用工具使用方法见附件4 |

### 验证对KunDB系统KunGate连接数过多告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统KunGate连接数过多告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、使用BenchmarkSQL工具，进行TPCC事务操作，构造KunGate 30秒内连接数数>50的场景，具体操作见附件  2、进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、TPCC事务操作成功  2、存在KunGate 连接数过多告警 |
| 测试脚本 |  |
| 实际结果 | 1、TPCC事务操作成功  2、存在KunGate 连接数过多告警，结果如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

### 验证对KunDB系统KunTablet CPU占用过多告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统KunTablet CPU占用过多告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1、登录系统任一服务器，进入一个KunTablet实例的POD， 运行CPU占用工具，模拟该实例占用该节点的CPU超过10%，详细操作见备注描述  2、进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、CPU占用工具运行成功，KunTablet实例占用该节点的CPU超过10%  2、存在 KunTablet CPU占用过多告警 |
| 测试脚本 |  |
| 实际结果 | 1、CPU占用工具运行成功，KunTablet实例占用该节点的CPU超过10%  2、存在 KunTablet CPU占用过多告警，结果如下： |
| 备注 | CPU、内存、磁盘占用工具使用方法见附件4 |

### 验证对KunDB系统KunTablet 内存过大告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统KunTablet 内存过大告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，且超级用户vt\_app已经创建完毕。  4、测试机上已成功安装Chrome、Firefox、Safari等浏览器，推荐使用Chrome |
| 测试步骤 | 1、登录系统任一服务器，进入一个KunTablet实例的POD， 运行内存占用工具，模拟该实例占用该节点的内存超过10%，详细操作见备注描述  2、进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、CPU占用工具运行成功，KunTablet实例占用该节点的内存超过10%  2、存在 KunTablet 内存过大告警 |
| 测试脚本 |  |
| 实际结果 | 1、内存占用工具运行成功，KunTablet实例占用该节点的内存超过10%  2、存在 KunTablet 内存过大告警，结果如下： |
| 备注 | CPU、内存、磁盘占用工具使用方法见附件4 |

### 验证对KunDB系统Mysql 主从延时过高告警功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统Mysql 主从延时过高告警功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常  5、KunDB中存在已经创建的Schema tpcc，并且已经通过BenchmarkSQL工具和KunLoader工具，创建完TPCC相关表和100仓的数据，具体操作请见备注 |
| 测试步骤 | 1、使用BenchmarkSQL工具，进行TPCC事务操作，构造replica或readonly节点的流水比master节点落后超过30s场景，具体操作见附件  2、进入Aquila界面，打开告警->告警查询，查看告警信息 |
| 预期结果 | 1、TPCC事务操作成功  2、存在Mysql 主从延时过高告警 |
| 测试脚本 |  |
| 实际结果 | 1、TPCC事务操作成功  2、存在Mysql 主从延时过高告警，结果如下： |
| 备注 | 装载数据和TPCC建表、建索引等方法详见附录1，附录2 |

## 数据库多副本同步

### 验证对KunDB系统数据库多副本同步功能

|  |  |
| --- | --- |
| 测试目的 | 验证对KunDB系统数据库多副本同步功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行。  4、KunDB产品在服务器上已经被成功安装，且运行正常 |
| 测试步骤 | 1. 通过Manager，安装单shard数据库dbtest2，有1个master，2个replica，且分布在不同的节点上。 2. 通过BenchmarkSQL工具生成10仓TPCC数据，并导入到数据库dbtest2中，具体操作步骤见备注 3. 关掉master节点的电源，查询两个replica节点的数据 4. 继续关掉一个replica节点的电源，查询另外一个replica节点的数据 |
| 预期结果 | 1、数据库dbtest2安装成功  2、10仓TPCC数据导入成功   1. 两个replica节点的数据与导入的数据一致 2. replica节点的数据与导入的数据一致 |
| 测试脚本 |  |
| 实际结果 | 1、数据库dbtest2安装成功  2、10仓TPCC数据导入成功  3、两个replica节点的数据与导入的数据一致  4、replica节点的数据与导入的数据一致 |
| 备注 | 装载数据和TPCC建表、建索引、执行事务操作等方法详见附录1，附录2 |

# 可移植性

# 兼容性

该文档中，图形化工具Waterdrop采用的是JDBC驱动连接数据库，性能测试中的webservice是通过ibatis中间件连接数据库,并且支持东方通应用服务器软件TongWeb V6.1。

# 可扩展性

## 弹性热扩展

### 验证KunDB系统物理扩缩容功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统物理扩缩容功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，集群中有5个节点。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有4个shard分区，且分布在不同的节点上。  6、Loadrunner与web程序均已调试成功，具体方法备注 |
| 测试步骤 | 1、执行LoadRunner脚本，运行几分钟至平稳状态  2、对一个节点进行断网操作，时间为10分钟  3、10分钟后，恢复网络 |
| 预期结果 | 1. LoadRunner脚本运行正常至平稳状态 2. 断网过程中，LoadRunner脚本运行TPS会下降 3. 网络恢复后，LoadRunner脚本运行TPS恢复到原稳定值 |
| 测试脚本 |  |
| 实际结果 | 1. 整个过程中，LoadRunner脚本运行TPS变化如下图所示，满足节点扩容容的要求   9421345a18f2d9fc80a703bc6172456 |
| 备注 | LoadRunner脚本和WebService程序，由检查方提供并运行 |

### 验证KunDB系统弹性热扩展功能

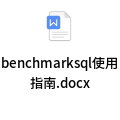
|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统弹性热扩展功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，集群中有5个节点。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有2个shard分区，且分布在不同的节点上。  6、Loadrunner与web程序均已调试成功 |
| 测试步骤 | 1、通过Transwarp Manager，扩展集群节点到6台机器，具体操作方法见文档《KunDBManual.docx》中集群节点扩缩容操作部分  2、100并发tpcc程序运行2mins，记录运行结果  3、通过loadrunner实现100用户并发对创建表单的插入操作  4.在高并发插入执行期间，进行KunDB扩容操作，具体操作方法见文档《KunDBManual.docx》中***KunDB运维和开发人员手册->水平扩缩容->扩容操作***章节  5.100并发tpcc程序运行2mins，记录运行结果 |
| 预期结果 | 1. 集群节点扩容成功   2、loadrunner高并发插入操作执行成功，且可平稳运行  3.在线业务存在情况下，kundb扩容成功  4.相同参数配置下，扩容前后tpcc运行结果对比，扩容后的性能结果高于扩容前的性能结果 |
| 测试脚本 |  |
| 实际结果 | 1. 集群节点扩容成功   扩容前:  2020-05-13 15-06-18屏幕截图  扩容后:  选区_686  2、预定义扩容后的shard扩展成功  2020-03-26 15-24-00屏幕截图  2020-03-26 15-25-45屏幕截图  3.新旧shard的数据迁移和角色切换成功，可通过ctld服务的控制面板查看  扩容前:  2020-03-27 12-09-49屏幕截图  扩容后:  2020-03-26 16-38-17屏幕截图  4.相同参数配置下，扩容前后tpcc运行结果对比，扩容后的性能结果高于扩容前的性能结果  扩容前:  2020-04-30 09-58-51屏幕截图  扩容后:  2020-05-06 17-23-41屏幕截图 |
| 备注 |  |

## 在线缩容

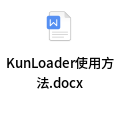
### 验证KunDB系统支持在线缩容功能

|  |  |
| --- | --- |
| 测试目的 | 验证KunDB系统支持在线缩容功能 |
| 前提条件 | 1、6台基于国产化CPU和UOS操作系统的服务器正常运行，各个节点网络已被成功配置且可正常互通。  2、各服务器上JAVA环境已被正常安装，版本为Oracle JDK 1.8 64位  3、服务器上Transwarp Manager产品已被成功安装并正常运行，集群中有6个节点。  4、KunDB产品在服务器上已经被成功安装，且运行正常。  5、存在已创建的数据库kundb1，有4个shard分区，且分布在不同的节点上。 |
| 测试步骤 | 1、通过Transwarp Manager，预定义数据库kundb1缩容后的2个shard  2、通过后台命令，进行新旧shard的数据迁移和角色切换，具体操作见文档《KunDBManual.docx》中***KunDB运维和开发人员手册->水平扩缩容->缩容操作***章节  3.通过Transwarp Manager，将集群节点缩容到5台机器,具体操作方法见文档《KunDBManual.docx》中集群节点扩缩容操作部分 |
| 预期结果 | 1、预定义缩容后的shard操作成功，可通过Manager、ctld服务的控制面板查看结果  2、新旧shard的数据迁移和角色切换成功，可通过ctld服务的控制面板查看结果  3、集群节点缩容成功 |
| 测试脚本 |  |
| 实际结果 | 1、预定义缩容后的shard操作成功  2020-03-26 17-23-03屏幕截图  2020-03-26 17-24-01屏幕截图  2、新旧shard的数据迁移和角色切换成功，可通过ctld服务的控制面板查看  缩容前:  2020-03-26 16-38-17屏幕截图  缩容后:  2020-03-27 12-09-49屏幕截图  3、集群节点缩容成功  缩容前:  选区_686  缩容后:  2020-05-13 15-06-18屏幕截图 |
| 备注 |  |

# 附录1 BenchmarkSQL使用方法



# 附录2 KunLoader使用方法



# 附录3 KunDR使用方法



# 附录4 CPU、内存、磁盘占用工具使用方法

三个程序脚本见刻盘程序occupy\_resources.tar.gz，具体使用方法如下：

1. **occupy\_cpu.sh**

./occupy\_cup.sh [num] 参数[num]为需要暂用的CPU核数，需要根据具体资源确定

如：./occupy\_cup.sh 3

需要在结束之后，手动kill掉进程

1. **occupy\_mem.sh**

./occupy\_mem.sh [size] [time] 参数size为需要占用的Mem大小，time为持续时间，单位为s，建议以后台模式执行，如：

./occupy\_mem.sh 1G 10 &

执行结束后，会自动释放暂用的内存

1. occupy\_mem.sh

./occupy\_disk.sh [num] [blocksize] num为block的数量，blocksize为每个block的大小，如生成一个1M的文件，命令如下：

./occupy\_disk.sh 1024 1M

执行完之后，会生成期望大小的文件file.txt，需要手动清理

# 附件5 LoadeRunner运行脚本WebService执行程序

见刻盘程序kundb-lr-scritps.tar.gz、SpringBootDemo.jar，其中kundb-lr-scritps.tar.gz为LoadeRunner的测试脚本，SpringBootDemo.jar为WebService程序

# 附件6 增量备份相关工具和操作

## 增量备份与数据库同步测试案例中测试表的DDL:

create table iback\_tbl1(

codeid bigint,

pad bigint,

name varchar(50),

birth datetime,

primary key(codeid)

) ENGINE=InnoDB partition by hash(codeid) using binary\_md5;

create table iback\_tbl2(

codeid bigint,

pad bigint,

name varchar(50),

birth datetime,

primary key(codeid)

) ENGINE=InnoDB partition by hash(codeid) using binary\_md5;

create table iback\_tbl3(

codeid bigint,

pad bigint,

name varchar(50),

birth datetime,

primary key(codeid)

) ENGINE=InnoDB partition by hash(codeid) using binary\_md5;

## 增量备份与数据库同步测试案例中测试表数据生成脚本

#!/bin/bash

i=1

if [ -e /tmp/load.sql ];then

rm -rf /tmp/load.sql

fi

while [ $i -le 100000 ]

do

echo "insert into iback\_tbl1 values($i,1,'NA$i','2020-01-01');" >> /tmp/load.sql

echo "insert into iback\_tbl2 values($i,1,'NA$i','2020-01-01');" >> /tmp/load.sql

let i++

done

## 增量备份与数据库同步测试案例中测试表数据操作脚本

#!/bin/bash

if [ -e /tmp/crud.sql ];then

rm -rf /tmp/crud.sql

fi

n=1

while [ $n -le 50000 ]

do

echo "insert into iback\_tbl3 values($n,2,'NA$n','2020-03-03');" >> /tmp/crud.sql

echo "update iback\_tbl2 set pad=$n where codeid=$n;" >> /tmp/crud.sql

echo "delete from iback\_tbl1 where codeid=$n;" >> /tmp/crud.sql

let n++

done

## 增量备份与数据库同步测试案例中测试表数据检查sql

--iback\_tbl1:

select count(\*) from iback\_tbl1;

select min(codeid) from iback\_tbl1;

select max(codeid) from iback\_tbl1;

select count(\*) from iback\_tbl1 where codeid<=50000;

select count(\*) from iback\_tbl1 where codeid>50000;

--iback\_tbl2:

select count(distinct pad) from iback\_tbl2;

select count(pad) from iback\_tbl2 where pad>1;

select count(pad) from iback\_tbl2 where pad=1;

--iback\_tbl3:

select count(codeid) from iback\_tbl3;

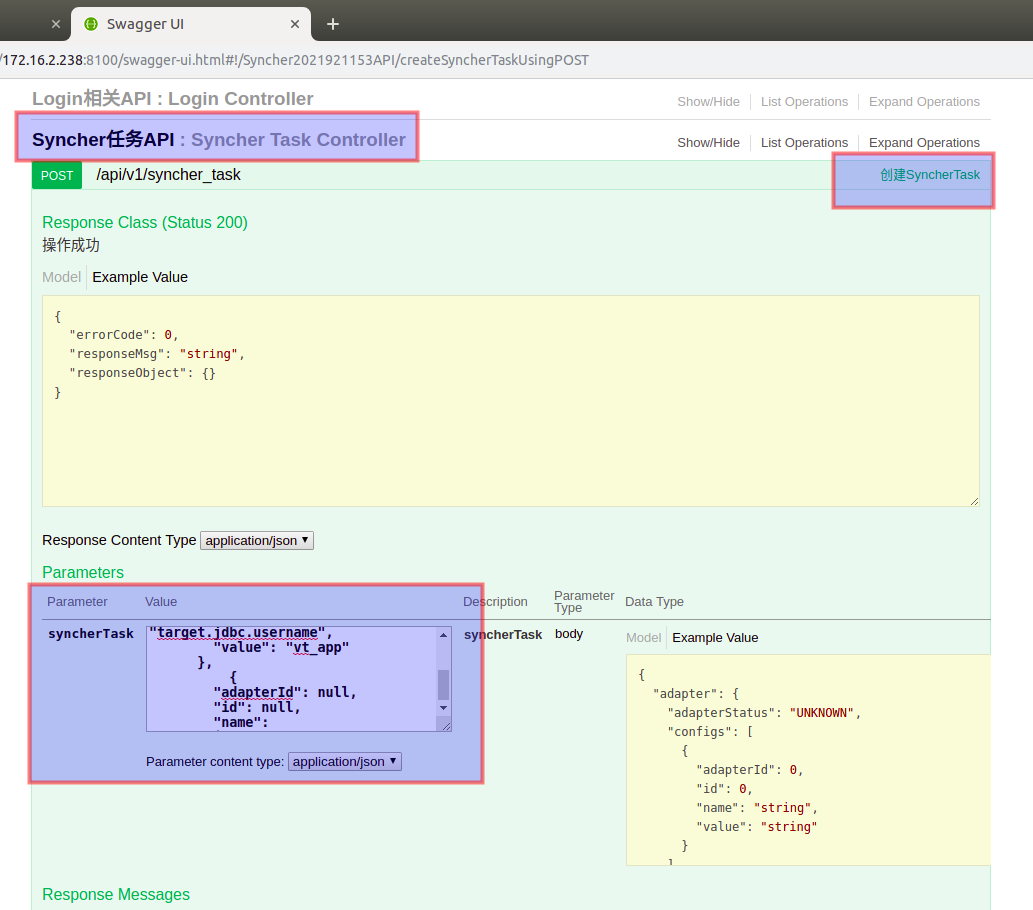
select min(codeid) from iback\_tbl3;

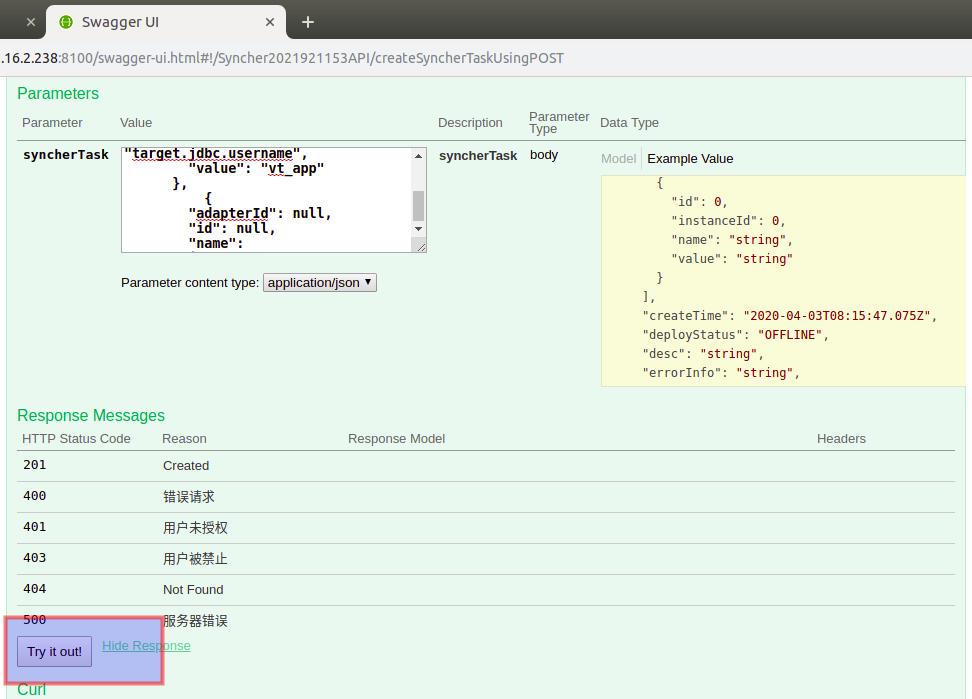
select max(codeid) from iback\_tbl3;

# 附件7 数据同步相关工具和操作

## 创建增量备份任务的操作步骤

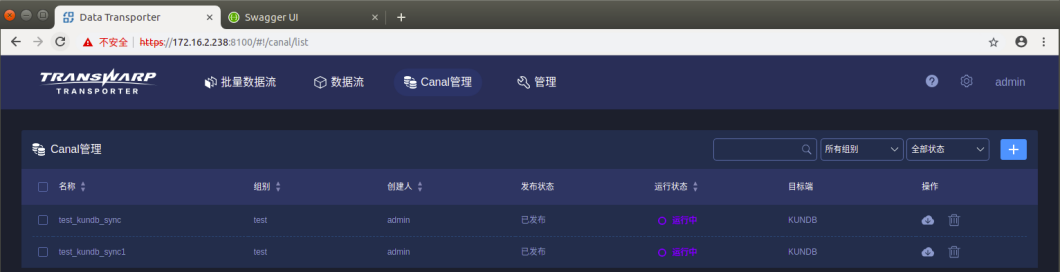
1. 进入tdt的swagger界面创建sync task,task的个数与源端kundb的shard个数一样



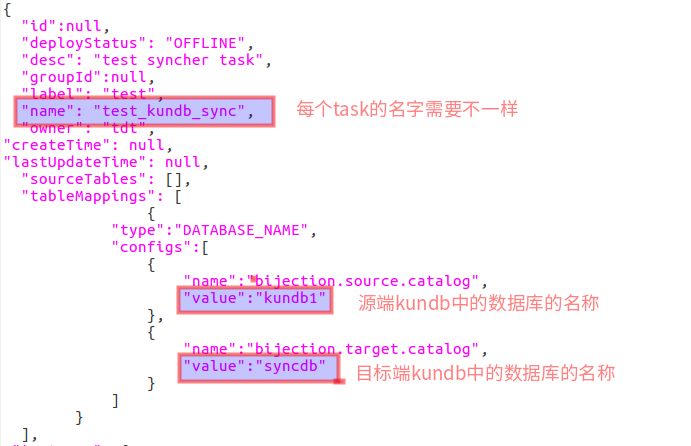


其中，TDT的swagger界面创建sync task的配置文件模板，见kundb\_incremental\_backup目录下的kundb\_increback\_task.json文件，kundb\_increback\_task.json中的注意事项见配置文件中的注意事项文件

1. 在tdt的transporter server界面的canal管理页面，将之前创建的同步task上线



1. Syc task的配置模板里的内容需要注意的地方如下图所示：







# 附件8 manager集群部署两套kundb产品

## manager集群部署两套kundb产品的操作步骤

1、需要部署两个zookeeper,安装第二个zookeeper的时候,在步骤[配置服务]的[基础参数]部分需要把所有的端口号都跟默认的(即第一个zookeeper的端口号)区分开.

注意,有一个进程的端口号(zookeeper.jmxremote.port)在[基础参数]中未暴露,在安装过程中会遇到该端口号冲突的问题,此时去zookeeper2的参数配置界面找到这个端口号进行修改,再继续安装就可以了

2、部署两个kundb产品.需要将两个kundb的依赖与端口号区分开

安装第一个kundb的时候,依赖和端口可按默认的设置.在安装第二个kundb的时候,需要将依赖修改为zookeeper2(即与第一个kundb依赖的zookeeper1区别开);在安装步骤[配置服务]的[基础参数]部分需要把所有的端口号都跟默认的(即第一个kundb的端口号)区分开.建议:可将所有以1开头的端口号改为以2开头;以2开头的端口号改为已4开头