ICS 35.020
L70

YD

中华人民共和国通信行业标准

YD/T 3775—2020

大数据 分布式事务数据库 技术要求与测试方法

Big data-distributed transactional database-technical specification and test methods

2020-12-09 发布　　2021-01-01 实施

中华人民共和国工业和信息化部　发布

目　　次

前　言

本标准是大数据系列标准之一，该系列标准名称和结构如下。

——大数据 分布式批处理平台技术要求与测试方法

——大数据 分布式分析型数据库技术要求与测试方法

——大数据 分布式事务型数据库技术要求与测试方法

——大数据 分布式流处理平台技术要求与测试方法

——大数据 时序数据库技术要求与测试方法

——大数据 商务智能分析工具技术要求与测试方法

——大数据 数据管理平台技术要求与测试方法

——大数据 数据集成工具技术要求与测试方法

——大数据 数据挖掘平台技术要求与测试方法

本标准按照 GB/T 1.1—2009 给出的规则起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本标准由中国通信标准化协会提出并归口。

本标准起草单位：中国信息通信研究院、华为技术有限公司、腾讯云计算有限公司、阿里云计算有限公司、中兴通讯股份有限公司、浙江蚂蚁小微金融服务集团股份有限公司、广州巨杉软件开发有限公司、上海爱可生信息技术股份有限公司、中国科学院计算技术研究所。

本标准主要起草人：魏凯、姜春宇、马鹏玮、王妙琼、王卓、李俊逸、叶涛、朱松、潘安群、韩雪、吴月玲、张玲东、付裕、周日明、蒋志勇、余军、郝大为、魏晗清、金官丁、张翔、詹剑锋、王磊、查礼。

大数据分布式事务数据库 技术要求与测试方法

1 范围

本标准规定了大数据分布式事务数据库的技术要求和具体的测试方法。

本标准适用于大数据分布式事务数据库产品的评估、验收等。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 5271.18—2008 信息技术词汇第 18 部分：分布式数据处理

GB/T 32400—2015 信息技术 云计算 概览与词汇

GB/T 35295—2017 信息技术大数据术语

3 术语定义及缩略语

3.1 术语定义

下列定义及术语适用于本标准

3.1.1

分布式系统 distributed system

网络联通的多台计算机组成，不同计算机之间通过网络通信传递信息进行计算任务相关的交流与协同。

3.1.2

节点 node

连接至网络中的一个连接点，具体的定义根据所应用的网络和协议各有不同。在分布式系统中，一个节点指代系统中一台连接至网络的计算或存储设备。

3.1.3

路由 Route

分组从源到目的地时，决定端到端路径的网络范围的进程。

3.1.4

分布式计算 distributed computing

覆盖存储层和处理层的，用于实现多类型程序设计算法模型的计算模式。

[GB/T 35295—2017，定义 2.1.22]

3.1.5

分布式数据处理 distributed data processing

数据操作分散到计算机网络各节点进行计算的过程。

[改写 GB/T 5271.18—2008，定义 18.1.8]

3.1.6

分布式批处理 distributed batch processing

在分布式系统架构上根据分布式数据处理和计算模式进行的批处理计算。

3.1.7

分布式文件系统 distributed file system

能够管理分布在多个节点上的文件的文件管理系统，节点间通过分布式系统中的网络进行通信和数据传输。

3.1.8

非关系型数据库 NoSQL database

键值、列、文件和图数据库管理系统等。在非关系型数据库管理系统中数据不是以关系模型组织的。

3.1.9

水平扩展 scale out

集成的一群个体资源作为一个单系统使用的过程。

[GB/T 35295—2017，定义 2.1.17]

3.1.10

扩展性测试 extensibility test

衡量系统的扩展性能加速比。

3.1.11

分布式事务数据库管理系统 distributed transactional database management system

面向事务交易型业务场景的分布式架构的数据库管理系统，以标准 SQL 语言形式进行操作，具有水平扩展能力，同时保证事务正确性。

3.1.12

租户 tenant

对一组物理和虚拟资源进行共享访问的一个或多个用户。

[改写 GB/T 32400—2015，定义 3.2.37]

3.1.13

多租户 multi-tenancy

对物理或虚拟资源的分配实现多个租户以及他们的计算和数据彼此隔离和不可访问。

[GB/T 32400—2015，定义 2.1.24]

3.1.14

流数据 streaming data

经由接口传递，从连续运行的数据源产生的数据。

[GB/T 35295—2017，定义 2.1.24]

3.1.15

结构化数据 structured data

数据表示形式，按此种形式，由数据元素汇集而成的每个记录的结构都是一致的并且可以使用关系模型予以有效描述。

[GB/T 35295—2017，定义 2.2.13]

3.1.16

非结构化数据 unstructured data

相对于结构化数据（即行数据，存储在数据库里，可以用二维表结构来逻辑表达实现的数据）而言，不方便用数据库二维逻辑表来表现的数据即称为非结构化数据。

[GB/T 35295—2017，定义 2.1.25]

3.1.17

半结构化数据 semi-structured data

介于结构化与非结构化之间的数据，表单是一个包含表单元素的区域，而表单元素是允许用户在表单中（比如：文本域、下拉列表、单选框、复选框等）输入信息的元素。

3.1.18

测试环境 test environment

软件测试的基础，测试项目依托测试环境，运行于测试环境之上，测试环境是测试所需的所有软硬件的总称。

3.1.19

吞吐率 throughout rate

衡量业务系统单位时间内提供服务的能力指标，在分布式批处理平台中可以指单位时间内处理的数据量或作业量。

3.1.20

表连接 table join

一张表中的行按照某个条件（连接条件）和另一张表中的行连接起来形成一个新行的过程。

3.1.21

子查询 subquery

常用计算机语言 SELECT-SQL 语言中嵌套查询下层的程序模块。当一个查询是另一个查询的条件时，称之为子查询。

3.1.22

表空间 table space

用来管理数据存储逻辑概念，表空间只是和数据文件（ORA 或者 DBF 文件）发生关系，数据文件是物理的，一个表空间可以包含多个数据文件，而一个数据文件只能隶属一个表空间。

3.1.23

临时表 temporary table

用于存储业务执行过程中产生的临时数据，从而增加业务执行效率或便于执行查询的数据存储方式。

3.1.24

索引 index

单独的、物理的对数据库表中一列或多列的值进行排序的一种存储结构，它是某个表中一列或若干列值的集合和相应的指向表中物理标识这些值的数据页的逻辑清单。

3.1.25

事务/数据库事务 transaction

访问并可能更新数据库中各种数据项的一个程序执行单元，具备原子性、一致性、隔离性、持久性四个特征。

3.1.26

自定义函数 user-defined function

用户按照业务需求自己编写的，存储在数据库中的代码块，以程序方式进行调用并返回结果值。

3.1.27

存储过程 stored procedure

在大型数据库系统中，一组为了完成特定功能的 SQL 语句集，存储在数据库中，经过第一次编译后再次调用不需要再次编译，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。

3.1.28

系统表 system table

关系型数据库存放结构元数据的地方，比如表和字段以及内部登记信息等。

3.1.29

视图 view

从一个或多个表（或视图）导出的表。视图是一个虚表，即视图所对应的数据不进行实际存储，数据库中只存储视图的定义，在对视图的数据进行操作时，系统根据视图的定义去操作与视图相关联的基本表。

3.1.30

外部表 external table

不存在于数据库中的表。通过提供描述外部表的元数据，把一个操作系统文件当成一个只读的数据库表，就像这些数据存储在一个普通数据库表中一样来进行访问。外部表是对数据库表的延伸。

3.1.31

数据库连接 DBlink

单向的数据库连接，通过配置可以访问其他数据库中的数据。

3.1.32

会话 session

通信双方从开始通信到通信结束期间的一个上下文。这个上下文是一段位于服务器端的内存：记录了本次连接的客户端机器、通过哪个应用程序、哪个用户登录等信息。

3.1.33

作业 job

数据库按顺序执行的指定操作。作业可以执行一系列活动，包括运行 Transact-SQL 脚本、命令行应用程序、查询任务等。

3.1.34

锁 lock

在执行多线程时用于强行限制资源访问的同步机制，即用于在并发控制中保证对互斥要求的满足。

3.1.35

缓存 cache

数据交换的缓冲区，当某一硬件要读取数据时，会首先从缓存中查找需要的数据，如果找到了则直接执行，找不到的话则从内存中找。

3.1.36

数据分区 partition

物理数据库设计技术，将数据从逻辑上进行区分，并可以在操作时制定访问特定的分区，主要目的是为了在特定的 SQL 操作中减少数据读写的总量以缩减响应时间。

3.1.37

相关子查询 correlated sub-query

特指内部子查询需要引用外部查询中的结果的子查询语句类型。

3.1.38

非相关子查询 uncorrelated sub-query

特指内部子查询不需要引用外部查询中的结果的子查询语句类型。

3.1.39

负载 Load

在计算领域，负载指计算机在指定时间需要完成的计算量，是实例或一组实例将要执行的实际工作的抽象。

3.1.40

实时交互分析 real-time interactive analysis

实时分析应用，预期在短时间内完成，以查询为主。

3.1.41

数据库实例 Service

独立占用物理内存的数据库服务进程。

3.2 缩略语

下列缩略语适用于本标准。

CPU	中央处理器	Central Processing Unit
CSV	文本分隔值	Comma-Separated Values
DML	数据管理语言	Data Management Language
JDBC	Java 数据库连接	Java Database Connectivity
MPP	大规模并行处理	Massively Parallel Processing
NoSQL	非关系型的数据库	Not Only SQL
ODBC	开放数据库连接	Open Database Connectivity
SQL	结构化查询语言	Structural Query Language

4 总体要求

分布式事务数据库主要功能是存储海量结构化的数据，支持关系型模型，提供结构化数据导入、数据管理等基本功能，支持即席查询、复杂查询、统计等分析能力，实现对结构化数据全生命周期的处理。

分布式事务数据库应包括如下主要功能模块：

a） 分布式架构；

b） SQL 执行引擎；

c） 事务管理；

d） 运维监控模块；

e） 安全认证模块。

分布式事务数据库应满足如下功能和性能要求：

a） 具备结构化数据处理和关系型模型的基本功能，包括对 SQL 语法的支持，基本的关系数据操作等；

b） 具备完善的运维管理系统，包括集群的安装部署、资源监控、集群操作、作业执行、用户管理、升级等；

c） 具备良好的系统安全性，具备认证能力，支持权限分配，包括数据库、表、行列级别权限控制，支持数据的加密和操作的安全审计；

d） 具备高可用性，包括各个模块的高可用性，支持数据的冗余备份，具备多种备份机制；

e） 具备良好的扩展能力；

f） 保障日常任务执行的性能要求。

5 技术要求

5.1 基本功能

分布式事务数据库应该具备结构化数据处理和关系型模型的基本功能，包括：

a） 支持数值、字符、时间日期、布尔等常用数据类型；

b） 支持支持数值运算、比较运算、逻辑运算、字串拼接、类型强转等常用操作符；

c） 支持 utf-8，gbk 等常见数据类型；

d） 支持数值函数、字符函数、时间日期函数、类型转换、条件表达式、正则表达式、安全函数、聚合函数等多种函数；

e） 涵盖常见功能操作语句类型，如增删改查、子查询、连接、排序等；

f） 能够使属于不同分片的数据实现完整数据操作，且实现结果与单库一致；

g） 具备不同隔离级别下的事务能力；

h） 能够显式展现 SQL 语句的执行计划；

i） 支持数据表分区功能；

j） 具备将表和分区按某种规则分片存储的能力；

k） 能够建立适应精准查询和范围查询的索引；

l） 能够实现可编程函数；

m） 实现 prepare 能力，能够完成 Oracle 数据存储过程同功能；

n） 基于 SQL 语句结果集的可视化表；

o） 数据库能够按照一定规则生成数字序列标识数据库动作；

p） 能够以显式及隐式两种方式提交事务；

q） 能够查看节点、表级别的元信息；

r） 能够实现数据副本数量的配置能力。

5.2 兼容性要求

分布式事务数据库应该对上层应用开放开发接口，支持异构硬件平台和不同的操作系统，具体包括：

a） 能够以 JDBC 连接方式连接；

b） 能够以 ODBC 连接方式连接；

c） 能够完成数据类型隐式转换工作；

d） DB2、Oracle，MySQL 等数据源单次全量迁移的能力；

e） 能够使用其他数据库进行远端连接功能；

f） 数据库能够在 X86 等主流硬件上正常运行。

5.3 管理能力要求

分布式事务数据库应该具备良好的运维管理能力，帮助运维人员平滑使用和维护分布式事务数据库，主要包括以下要求：

a） 数据库具备友好的部署能力；

b） 全局范围内的数据库配置在线管理能力；

c） 能够对数据库状态进行实时监控；

d） 具备实时告警能力；

e） 数据库不会产生死锁或具备死锁自动处理能力；

f） 具备节点结构及数据保证强一致性的能力；

g） 数据库具备在线组件升级功能；

h） 能够进行在线的建表及路由分片操作能力；

i） 进行常见的数据库管理操作命令能力；

j） 面向运维的友好的统计分析能力；

k） 各类日志（查询/慢查询等）的查看及处理能力；

l） 以多种方式（定时、增量、全量、加密、压缩）对数据完成备份工作；

m） 能够按时间或版本信息进行数据恢复，且实现恢复一致性；

n） 具备对集群内的数据库进行资源分组配置的能力。

5.4 高可用性要求

分布事务数据库应该具备在出现故障后系统进行恢复能力，主要包括数据备份和恢复、各组件的主备节点切换等。主要的故障类型有插拔硬盘、网线、关机、重启等。主要包括：

a） 电源插拔，硬盘插拔网线，交换机，机架，机房故障下的数据库服务能力（故障监测、自动切换、副本自动补齐）；

b） CPU 资源占用，I/O 资源占用，内存资源占用，磁盘空间占用下数据库服务能力（故障监测、自动切换、副本自动补齐）；

c） 数据库系统文件被损坏情况下的数据库服务能力（故障监测、自动切换、副本自动补齐）；

d） 上层应用与数据交互过程中发生故障下的数据库服务能力（故障监测、自动切换、副本自动补齐）。

5.5 扩展性要求

分布式事务数据库应该具备良好的扩容能力，根据业务需求随时进行集群的扩展和收缩。主要包括：

a） 具备将接入数据库的计算压力进行平衡的能力；

b） 具备将数据库内的数据进行平衡的能力，并能够评测；

c） 数据库集群具备集群扩展能力，且性能能够随之提升；

d） 数据库集群具备集群缩减能力。

5.6 安全性要求

分布式事务数据库应该具有安全保护的技术，以防止恶意的访问和攻击，防止关键数据的泄露，支持完备的权限管理和审计日志功能。具体包括如下安全功能：

a） 能够对数据库内数据操作进行权限验证工作；

b） 能够对接入数据库用户进行身份认证工作；

c） 能够对数据库内操作进行审计工作；

d） 能够配置数据库的流量上限，例如提供连接数限制等。

5.7 性能要求

分布式事务数据库应该能够满足特定应用场景中，对高频词的事务业务性能要求，性能测试主要考虑如下方面：

a） 测试需基于一定资源冗余的硬件环境、包括服务器节点规模、网络链路、内存总量、CPU 总量、磁盘总量等；

b） 测试需定义良好的测试业务场景，包括合理规模的数据量，与实际业务相符的数据模式和操作模式；

c） 测试需定义合理的衡量指标，例如单节点吞吐量、系统总吞吐量、平均时延、最大时延、执行时间等；

d） 常见的测试场景包括多类型查询、混合负载、压力测试、稳定性测试等。

6 测试方法

本章规定了分布式事务数据库各类技术要求的测试方法，包括基本功能、运维能力、兼容能力、安全能力、高可用能力、扩展性能力。

6.1 测试环境要求

a） 要求测试集群节点数大于3。

b） 要求测试环境网络通畅。

6.2 详细测试方法

基本功能测试方法

测试项目	数据类型
测试目的	1） 验证分布式数据库产品数据类型的支持度； 2） 验证是否涵盖数值型、浮点型（精确和非精确）和字符串型等常用数据类型
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 为数值型、浮点型（精确和非精确）、字符串型和时间日期型分别新建四张包含所有声明支持的数据类型的分片表； 2） 在四张表执行对应的插入语句；伪SQL示例：INSERT INTO inttable VALUES（123，...）；INSERT INTO floattable VALUES（123.456，...）；INSERT INTO stringtable VALUES（'ABC'，...）；INSERT INTO datetable（'2018-01-01'，...）； 3） 在四张表执行查询语句；伪SQL示例：SELECT intcolumn，... FROM inttable ；SELECT floatcolumn，... FROM floattable；SELECT stringcolumn，... FROM stringtable；SELECT datecolumn，... FROM datetable； 4） 在四张表各自查询字段与字段对应数据类型的常量数据相加或连接的查询语句；伪SQL示例：SELECT intcolumn+1，... FROM inttable ；SELECT floatcolumn+1.00，... FROM floattable；SELECT CONCAT（stringcolumn，'DEF'），... FROM stringtable；SELECT DATE_ADD（datecolumn，INTERVAL 1 DAY），... FROM datetable； 5） 在四张表各自查询字段与字段对应数据同大类型（数值型、字符串型等）但具体数值类型不同的常量相加或连接（例如整形和双精度浮点相加，短长度和长长度的字符串连接）；伪SQL示例：SELECT intcolumn+unsignedlongcolumn，... FROM inttable ；SELECT floatcolumn+decimalcolumn，... FROM floattable；SELECT CONCAT（stringcolumn，varbinarycolumn），... FROM stringtable；SELECT DATE_DIFF（datecolumn，timestampcolumn），... FROM datetable； 6） 再创建一张含有所有支持的数据类型的表，填充少量数据。然后查询数值型字段全部相加、字符串型字段全部连接的结果、混合大类型数据相加或连接（例如数值和字符串连接，数值和字符串相加）；伪SQL示例：SELECT intcolumn+unsignedlongcolumn+floatcolumn+decimalcolumn，CONCAT（stringcolumn，varbinarycolumn），DATE_DIFF（datecolumn，timestampcolumn），CONCAT（CONCAT（intcolumn+unsignedlongcolumn+floatcolumn+decimalcolumn，CONCAT（stringcolumn，varbinarycolumn）），DATE_DIFF（datecolumn，timestampcolumn）），... FROM alltable
预期结果	1） 第一步四张表都可以创建成功；

	2） 第二步插入语句没有报错，数据正常插入； 3） 第三步查询语句能够正常查询； 4） 第四步应当返回同样数据类型的合理且正确的查询结果； 5） 第五步应当返回合理且正确的查询结果； 6） 第六步表可以创建成功，数据能正常插入，查询语句应当返回合理查询结果（结果正确和报错均合理，结果错误或出现故障为不合理）

测试项目	操作符
测试目的	验证分布式数据库是否支持操作符号，例如|，&，!等常见操作符
测试环境	分布式数据库环境
前置条件	1） 分布式数据库产品部署正常且正常运行； 2） 测试表预先填充 100 行或更多数据，数据分布到两个或更多节点； 3） 测试表应当包含至少一行有效数据（intcolumn 等于 1，stringcolumn 等于'abc'），使得测试中使用的 SQL 符合条件的数据不为空集
测试步骤	1） 测试算术运算符：加减乘除等，不限语法；伪 SQL 示例：SELECT intcolumn+1，intcolumn-1，intcolumn*1，intcolumn/1，intcolumn+intcolumn，intcolumn-intcolumn，intcolumn*intcolumn，intcolumn/intcolumn FROM inttable WHERE intcolumn=（2-1×2）/2+1； 2） 测试字符串连接：||或 CONCAT 等方式，不限语法；伪 SQL 示例：SELECT CONCAT（stringcolumn，'def'），CONCAT（stringcolumn，stringcolumn）FROM stringtable WHERE stringcolumn=CONCAT（'ab'，'c'）； 3） 测试逻辑运算与比较符，与或非，不限语法；伪 SQL 示例；SELECT intcolumn AND 0，intcolumn AND 1，intcolumn AND intcolumn ，intcolumn OR 1，intcolumn OR 0，intcolumn OR intcolumn ，NOT intcolumn， intcolumn>0， intcolumn<0， intcolumn=1， intcolumn=intcolumn FROM inttable WHERE intcolumn>0 AND intcolumn<=1； 4） 测试集合运算，IN EXISTS 等集合操作，不限语法：伪 SQL 示例 SELECT intcolumn FROM inttalbe where intcolumn IN（1，2，3，....）and NOT IN（2）；SELECT intcolumn a FROM inttalbe where EXISTS（SELECT intcolumn FROM inttable b WHERE a.intcolumn=b.intcolumn AND b.intcolumn=1）
预期结果	1） 所有查询正常运行； 2） 所有正常运行的查询结果值和普通单机数据库一致

测试项目	字符集
测试目的	验证分布式数据库是否支持 utf-8，gbk 等常见数据类型
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 针对每一种声明支持的字符集（支持的字符集应当包含 utf8），创建一张表，并测试： 1.1）测试增删改查操作； 1.2）对于支持 utf8、gbk 等含中文的情况，测试汉字“中国”增删改查操作；

	1.3）对于支持查询二进制值的情况，测试查询汉字“中国”的编码值，确定以正确的字符集存储。 2） 针对声明支持两种或更多字符集的情况，测试： 2.1）如果声明支持存储字符集转换，在支持的范围内测试存储字符集转换； 2.2）如果声明支持连接字符集和存储字符集不一致，测试连接字符集与存储字符集不相同时，数据增删改查操作是否合理处理，连接字符集与存储字符集都支持的汉字在增删改查时是否正确转换。 3） 如果声明支持二进制字符串，测试插入和查询出的二进制字符串的二进制数值是否完全一致
预期结果	1） 支持的字符集包含 utf8； 2） 第 1.1 步操作正常，结果正确； 3） 第 1.2 步查询到的汉字能正常显示； 4） 第 1.3 步查询到的汉字编码值正确，和普通数据库相同； 5） 第 2.1 步数据正确转换； 6） 第 2.2 步数据正确转换； 7） 第 3 步编码值一致

测试项目	常用函数
测试目的	验证分布式数据库的日期函数、算数函数等常见函数
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 测试算术函数的增改查操作，例如四舍五入、截断、绝对值等； 2） 测试字符串函数的增改查操作，例如字符串连接，字符串替换等； 3） 测试时间日期函数的增改查操作，例如时间加减，时间显示格式转换等； 4） 如果声明支持聚合函数，测试支持的聚合函数的查询操作，例如最大最小值，计数等
预期结果	操作结果正确，和普通单机数据库相同

测试项目	SQL 语法
测试目的	涵盖常见功能操作语句类型，如增删改查、子查询、连接、排序等
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	测试样例伪 SQL 如下： 1） INSERT INTO inttable（intcolumn，...）VALUES（1，...）； 2） INSERT INTO inttable（intcolumn，...）VALUES（2，...），（11，...），（12，...），（101，...）； 3） UPDATE inttable SET intcolumn=3 WHERE intcolumn=1； 4） UPDATE inttable SET intcolumn=intcolumn-1 WHERE intcolumn=2； 5） DELETE FROM inttable WHERE intcolumn=3； 6） SELECT intcolumn FROM inttable WHERE intcolumn=1； 7） SELECT intcolumn FROM inttable WHERE intcolumn>10 AND intcolumn <1000；

	8） SELECT FLOOR（intcolumn/10），COUNT（intcolumn）FROM inttable WHERE intcolumn <1000 GROUP BY FLOOR（intcolumn/10）； 9） SELECT FLOOR（intcolumn/10），COUNT（intcolumn）FROM inttable WHERE intcolumn <1000 GROUP BY FLOOR（intcolumn/10）ORDER BY FLOOR（intcolumn/10）DESC； 10）SELECT FLOOR（intcolumn/10），COUNT（intcolumn）FROM inttable WHERE intcolumn <1000 GROUP BY FLOOR（intcolumn/10）ORDER BY FLOOR（intcolumn/10）DESC LIMIT 2； 11）SELECT FLOOR（intcolumn/10），COUNT（intcolumn）FROM inttable WHERE intcolumn <1000 GROUP BY FLOOR（intcolumn/10） HAVING COUNT（intcolumn）<=1 ORDER BY FLOOR（intcolumn/10）DESC LIMIT 2； 12）插入足量数据后测试：SELECT intcolumn FROM inttable ORDER BY intcolumn LIMIT 1000； 13）SELECT intcolumn FROM inttable ORDER BY intcolumn LIMIT 10，10； 14）SELECT a.intcolumn，b.intcolumn FROM inttable a，inttable b WHERE a.intcolumn=b.intcolumn AND a.intcolumn <100； 15）SELECT a.intcolumn，COUNT（b.intcolumn）FROM inttable a，inttable b WHERE a.intcolumn=b.intcolumn AND a.intcolumn <100 GROUP BY a.intcolumn HAVING COUNT（b.intcolumn）<=1 ORDER BY a.intcolumn LIMIT 1，3； 16）SELECT（SELECT intcolumn FROM inttable a WHERE a.intcolumn=b.intcolumn）FROM inttable b WHERE b.intcolumn<100； 17）SELECT intcolumn FROM（SELECT intcolumn FROM inttable WHERE intcolumn<100）b WHERE intcolumn >10； 18）SELECT intcolumn FROM inttable WHERE intcolumn IN（SELECT intcolumn FROM inttable WHERE intcolumn<100）AND intcolumn >10； 19）SELECT intcolumn FROM inttable WHERE intcolumn <10 UNIONSELECT intcolumn FROM inttable WHERE intcolumn <100； 20）SELECT intcolumn FROM inttable WHERE intcolumn <10 UNIONALL SELECT intcolumn FROM inttable WHERE intcolumn <100； 测试样例在分片表上测试； 只测试被声明支持的样例
预期结果	1） 各个查询结果正确； 2） 1 到 20 测试除第 8 个测试允许排序不同，和普通数据库相同

测试项目	分片操作能力
测试目的	能够使属于不同分片的数据实现完整数据操作，且实现结果与单库一致
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 新建一张分片表，并插入一定量数据，设法展示或证明数据都分布到各个分片上； 2） 采用 where 条件对分片字段过滤，查询单条数据； 3） 对分片字段范围查询，且排序；

	4） 对分片字段进行分组查询，并统计各个分组相同值的个数
预期结果	1） 第二步查询的 SQL 发送到正确的分片，并查询到正确的数据； 2） 第三步查询到范围内数据，并正确排序； 3） 第四步正常分组，分组内行数计算正确

测试项目	分布式事务
测试目的	具备不同隔离级别下的事务能力
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 新建一张分片表，除分片字段外，需要有至少一个 int 型字段，并插入部分数据； 2） 分别在声明支持的隔离级别下简单测试隔离级别正确性：使用测试普通单机数据库的隔离级别正确性的方法测试； 3） 分别在声明支持的隔离级别下充分测试隔离级别正确性；充分测试方法： 3.1）构造或选择两行不在同一节点的数据行，数据行的 int 型非分片字段设置数值为 0，后称该字段为 C1，两行数据后称 R1 与 R2； 3.2）T1 线程，产生一个新的整型的范围在 1 到 100 的随机数，在分布式数据库开启事务，扣减 R1 的 C1 字段该随机值数值，增加 R2 的 C1 字段该随机值数值，提交事务；然后 T1 线程循环执行此过程； 3.3）T2 线程，休眠 0.1 秒，开启事务，查询 R1 与 R2 行的 C1 字段及其和并记录结果，休眠 0.1 秒，查询 R1 与 R2 行的 C1 字段及其和并记录结果，提交事务；然后 T2 线程循环执行此过程 1000 次； 3.4）停止 T1 线程； 3.5）汇总分析 T2 线程记录的结果。 补充：可按照上述测试步骤或使用 TPCC 测试程序进行验证
预期结果	1） 简单测试方法下，隔离级别正确； 2） 充分测试方法下： 2.1）在 read uncommitted 的情况下，T2 线程在事务中的读取，可能并且应该出现字段和不为 0 的现象，若满足则 read uncommitted 隔离级别大概率正确。 2.2）在 read commited 的情况下，T2 线程在事务中的读取，一次都不出现字段和不为 0 的现象，可能并且应该出现一个事务中两次查询 R1 的 C1 字段值发生变化，若满足则 read commited 隔离级别大概率正确。 2.3）在 repeatable read 的情况下，T2 线程在事务中的读取，一次都不出现字段和不为 0 的现象，并且一次都不出现一个事务中两次查询 R1 的 C1 字段值发生变化的现象，可能并且应该出现，两个事务之间查询到的 R1 的 C1 字段值发生变化，若满足则 repeatable read 隔离级别大概率正确。 2.4）在 serializable 的情况下，此测试现象的要求和 repeatable read 相同，T1 线程可能出现因锁等待变慢（根据实现方式不同，也可能无此现象），但该测试方法并不足以证明 serializable 的正确性

测试项目	执行计划解析
测试目的	能够显式展现 SQL 语句的执行计划

测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 准备一张分片表，并插入部分数据； 2） 查询一条跨库语句的执行计划； 3） 执行这条跨库语句
预期结果	1） 第二步显示跨库语句将要发送的分片； 2） 第三步观察到跨库语句发送到执行计划中列出的分片，并查询正确

测试项目	表分区（partition）
测试目的	支持数据表分区功能
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 如果支持分片字段和分区字段是同一字段，创建一张分片表，分区用的字段是分片字段，插入数据，确保每个分区每个分片都有数据； 2） 执行一个范围查询，确保这个查询的数据分布在不同的分片，但是都在同一个分区； 3） 提供证明方式，证明查询只访问了限定的分区，方式不限，例如：在这个上一步的 SQL 上指定错误分区查询，再指定正确分区查询； 4） 如果支持分片字段和分区字段不是同一字段，再创建另一张分片表，分区用的字段不是分片字段，插入数据，确保每个分区每个分片都有数据； 5） 执行一个范围查询，确保这个查询的数据分布在不同的分片，但是都在同一个分区； 6） 提供证明方式，证明查询只访问了限定的分区，方式不限，例如：在这个上一步的 SQL 上指定错误分区查询，再指定正确分区查询
预期结果	1） 第二步能够正确查询数据； 2） 第三步可以证明只访问了范围内的分区，例如：指定错误分区后，查不到数据，指定正确分区了才查出数据； 3） 第五步能够正确查询数据； 4） 第六步可以证明只访问了范围内的分区，例如：指定错误分区后，查不到数据，指定正确分区了才查出数据

测试项目	数据 sharding 能力
测试目的	具备将表和分区按某种规则分片存储的能力
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 分别以声明支持的分片算法新建分片表。向这些张表插入一定量数据； 2） 分别对这些表执行范围查询，精确查询以及聚合函数查询； 3） 提供方式证明，数据被按照声明的分片算法正确分布，方式不限

预期结果	1） 第一步建表成功，数据插入成功； 2） 第二步查询数据全部正确，和普通单机数据库相同； 3） 第三步提供确实证据证明正确分布

测试项目	索引
测试目的	能够建立适应精准查询和范围查询的索引
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 声明是否支持索引，索引是本地索引还是全局索引； 2） 新建一张分片表，分别在分片字段上（如果支持的话）和其他普通 int 字段上建索引；向分片表中灌入千万级数据； 3） 通过分片字段进行精准查询和范围查询； 4） 通过 int 字段进行精准查询和范围查询； 5） 通过执行计划（如果支持的话）查询上一步使用的索引
预期结果	1） 第三步查询时间比不用索引的查询时间花费较少；使用分片字段的查询能够发送到正确分片（不支持分片字段建索引或使用分片字段查询的系统无此要求）； 2） 第四步查询时间比不用索引的查询时间花费较少； 3） 通过执行计划可以了解到查询语句使用了索引

测试项目	自定义函数
测试目的	能够实现可编程函数
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 声明是否支持； 2） 若支持，提供测试用例，证明支持
预期结果	最后查询到自定义函数返回的结果都是正确的结果

测试项目	存储过程或 JDBC PREPARE 或 SQL Prepare 语法
测试目的	能够完成 Oracle 数据存储过程同功能或支持使用 JDBC Prepare 或实现 SQL Prepare 能力
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 声明是否支持； 2） 若支持，提供测试用例，证明支持
预期结果	预处理或存储过程创建成功，最后执行后的返回结果正确

测试项目	静态视图
测试目的	基于 SQL 语句结果集的可视化表
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 新建一张分片表，并导入部分数据； 2） 创建一个静态视图，例如：视图能够查询出分片表以分片字段从大到小排列的数据； 3） 查询视图
预期结果	1） 第三步能够正确查询出数据

测试项目	序列
测试目的	数据库能够按照一定规则生成数字序列标识数据库动作
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1.1）声明是否支持类 SEQUENCE 功能，是否支持类 AUTO_INCREMENT 功能，应当至少支持一种； 1.2）声明序列是全局还是本地； 1.3）声明序列是否保证单调递增； 1.4）声明序列是否保证连续； 1.5）声明序列是否保证唯一； 2） 针对声明支持 AUTO_INCREMENT 的情况： 2.1）创建一张包含一个 int 字段的分片表，字段属性为 AUTO_INCREMENT； 2.2）向分片表插入几条数据，再查询全表，确认 int 字段值情况； 2.3）创建另一张分片表； 2.4）向分片表插入几条数据； 2.5）通过 alter 语句添加一个属性为 AUTO_INCREMENT 的 int 字段； 2.6）确认增加的 int 字段数据情况； 3） 针对声明支持 SEQUENCE 的情况： 3.1）创建一个 SEQUENCE； 3.2）使用 SEQUENCE 生成序列值，确认序列值请况
预期结果	第 2.2、2.6、3.2 步确认的数值满足声明的情况

测试项目	显式/隐式事务
测试目的	能够以显式及隐式两种方式提交事务
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 新建分片表之后，打开自动提交；

	2） 插入几条数据，执行回滚事务的语句，最后查询该表，确认数据是否存在； 3） 执行开启事务的语句，插入几条数据，执行回滚事务的语句，最后确认数据是否存在； 4） 执行开启事务的语句，插入几条数据，执行提交事务的语句，最后确认数据是否存在； 5） 关闭自动提交； 6） 插入几条数据，并执行回滚事务的语句，最后确认数据是否存在； 7） 插入几条数据，并执行提交事务的语句，最后确认数据是否存在
预期结果	1） 第二步的数据已经成功插入； 2） 第三步数据没有插入，被回滚了； 3） 第四步的数据成功插入； 4） 第六步数据被成功回滚； 5） 第七步数据成功插入

测试项目	元信息查看
测试目的	能够查看节点、表级别的元信息
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	利用分布式数据库提供的方法查询节点个数，节点名称，表结构，表的分片规则、集群的元信息等信息
预期结果	1） 分布式数据库能够提供上诉信息的查询方式且能查询到更多的元信息； 2） 查询不影响业务

测试项目	数据副本控制能力
测试目的	能够实现数据副本数量的配置能力
测试环境	分布式数据库环境
前置条件	分布式数据库系统正常，且存在一定的压力测试
测试步骤	1） 通过分布式数据库提供的方法，动态添加一个数据副本，如果支持读写分离，该数据备份可提供读写分离。且该数据副本可以为该节点提供高可用服务。添加过程不中断压力测试； 2） 再增加一个数据副本，如果支持读写分离，这个数据副本可以进一步分摊整个节点的读请求； 3） 通过分布式数据库提供的方法，删除第三步添加的数据副本，这个过程不会中断压力测试
预期结果	1） 第二步成功增加一个数据副本，如果支持读写分离，可成功实现读写分离； 2） 第三步增加的数据副本，如果支持读写分离，可进一步分摊读请求； 3） 第四步删除数据副本之后，除性能因素外，分布式系统运行没有受到影响

兼容能力测试方法

测试项目	JDBC
测试目的	验证分布式事务数据库与 JDBC 接口的兼容性

测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 配置 JDBC 数据源和驱动； 2） 进行查询、插入、更新等操作 3） 查看执行结果
预期结果	1） JDBC 连接配置成功； 2） 查询、插入、更新操作得到正确结果

测试项目	ODBC
测试目的	验证分布式事务数据库与 ODBC 接口的兼容性
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 配置 ODBC 数据源和驱动； 2） 进行查询、插入、更新等操作； 3） 查看执行结果
预期结果	1） ODBC 连接配置成功； 2） 查询、插入、更新操作得到正确结果

测试项目	隐式转换
测试目的	验证分布式事务数据库支持数据类型隐式转换工作
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	执行附件 SQL 语句
预期结果	附件 SQL 语句执行结果正确
测试结果	
备注	— 准备 CREATE TABLE TYPE01（COL0 INT，COL1 INT）； CREATE TABLE TYPE02（COL0 FLOAT（3））； CREATE TABLE TYPE03（COL0 DECIMAL，COL1 DECIMAL）； CREATE TABLE TYPE04（COL0 DECIMAL（3，1））； CREATE TABLE TYPE05（COL0 INT，COL1 VARCHAR（3））； INSERT INTO TYPE05 VALUES（123，'123'）； — 执行隐式转换-转换成整数类型时小数部分截断情况，预期成功 INSERT INTO TYPE01 VALUES（123.456，23）；

	— 执行隐式转换-转换成近似数值类型时小数部分四舍五入情况，预期成功 INSERT INTO TYPE02 VALUES（1.23456789012345678）； — 执行隐式转换-超过精确数值类型极值范围，预期失败 INSERT INTO TYPE03 VALUES（-10000000000000000000000000000000000000000，123.2）； INSERT INTO TYPE03 VALUES（10000000000000000000000000000000000000000，34.45）； — 执行隐式转换-转换成精确数值类型时小数部分四舍五入情况，预期成功 INSERT INTO TYPE04 VALUES（12.89）； — 验证隐式转换-转换成整数类型时小数部分截断情况，预期值为（123，23） SELECT * FROM TYPE01； — 验证隐式转换-转换成近似数值类型时小数部分四舍五入情况，预期值为 1.23457 SELECT * FROM TYPE02； — 验证隐式转换-转换成精确数值类型时小数部分四舍五入情况，预期值为 12.9 SELECT * FROM TYPE04； — 验证隐式转换-VARCHAR 与 INT 之间的隐士转换，预期至少有一条查询能成功执行并返回一条记录 SELECT * FROM TYPE05 WHERE COL0='123'； SELECT * FROM TYPE05 WHERE COL1=123； — 删除表 DROP TABLE TYPE01； DROP TABLE TYPE02； DROP TABLE TYPE03； DROP TABLE TYPE04； DROP TABLE TYPE05；

测试项目	异构数据全量迁移
测试目的	验证分布式事务数据库对于 DB2、Oracle、Mysql 等数据源单次全量迁移的能力
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 在要迁移的异构数据库中建立一个 1G 大小的数据表；

	2） 使用迁移工具或程序将 1 步骤中建立数据表迁移到分布式数据库中； 3） 在分布式数据库中对迁移数据表执行查询操作，验证迁移表的完整性
预期结果	1） 1G 数据表成功迁移到分布式数据库中； 2） 测试步骤 3 中查询操作得到正确结果

测试项目	外部表
测试目的	验证分布式事务数据库能够使用其他数据库进行远端连接功能
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 对分布式事务数据库数据表建立外部表； 2） 使用分布式事务数据库查询数据表内容； 3） 在远端连接执行数据表内容； 4） 对比步骤 2 与步骤 3 结果
预期结果	1） 外部表建立成功； 2） 测试步骤 4 对比结果正确

测试项目	X86 等主流硬件兼容性
测试目的	数据库能够在 X86 等主流硬件上正常运行
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 验证分布式事务数据库能够正常运行； 2） 查看当前分布式事务数据库所基于的硬件环境； 3） 同时可以通过硬件厂商出具的报告进行验证
预期结果	分布式事务数据库能够支持 X86 等主流硬件兼容性

管理能力测试方法

测试项目	安装部署能力
测试目的	• 支持所有组件的自动化部署； • 安装部署操作以向导方式进行； • 是否显示安装进度； • 考察安装效率
测试环境	分布式数据库系统测试环境
前置条件	获取分布式数据库系统版本
测试步骤	• 选择节点准备进行数据库系统的安装； • 以向导的方式安装数据库系统；

	• 安装完成后由测试人员记录安装时间； • 查看各组件的状态
预期结果	• 组件均可以自动部署，部署过程简单易用； • 安装过程以向导方式进行； • 有进度提示，通过界面查看各组件状态正常

测试项目	配置管理能力
测试目的	• 支持数据库配置在线统一管理能力 • 是否可以采集到各节点配置信息并进行展示 • 配置信息是否可修改及修改后是否能实时动态生效 • 能否对多个节点同时下发配置信息
测试环境	分布式数据库系统测试环境
前置条件	分布式数据库系统运行正常
测试步骤	— 登录分布式数据库系统配置管理页面； — 查看各节点配置信息正确性； — 修改选定节点的配置信息并使其生效； — 确认配置修改是否生效
预期结果	— 支持各节点配置信息的同一采集及展示； — 支持在线动态生效配置项的修改； — 配置修改后可动态生效

测试项目	实时监控
测试目的	• 是否支持分布式数据库系统各节点状态信息的统一展示 • 当某节点出现状态异常时，是否能够实时反映
测试环境	分布式数据库系统测试环境
前置条件	分布式数据库系统运行正常
测试步骤	1. 登录分布式数据库监控系统页面； 2. 查看分布式数据库系统各节点状态信息； 3. 停止其中一个或多个节点，再次观察节点状态信息
预期结果	1. 支持各节点状态信息的统一展示能力； 2. 当节点状态发生变化时，能实时或准实时的反应状态变化

测试项目	告警功能
测试目的	● 是否支持实时告警 ● 可自动恢复类告警是否支持自动恢复

	● 是否支持告警的手动恢复 ● 是否支持历史告警查询 ● 告警管理能力支持情况，比如：告警级别的修改、告警阈值的设置等 ● 告警通知方式多样性能力
测试环境	分布式数据库系统测试环境
前置条件	分布式数据库系统运行正常
测试步骤	1. 构造实时告警（比如：停止分布式数据库系统中任一节点的进程），检查是否产生正确的告警信息； 2. 构造可自动恢复类告警，检查实时告警正确性，恢复该告警，查看实时告警情况； 3. 查询历史告警信息； 4. 进行告警级别修改或告警阈值设置等告警管理能力测试，并校验操作结果正确性； 5. 上述告警测试过程中，同步进行告警通知多样性能力检查
预期结果	1. 故障发生后，能够产生实时告警，告警源、告警原因和告警级别等信息正确，告警描述合理，告警产生时间正确。 2. 可自动恢复类告警产生正确，告警恢复后，告警信息可自动恢复，历史告警可查询； 3. 历史告警信息管理正确，可构造不同查询条件进行检索； 4. 具备常用的告警管理能力，如：告警级别修改、告警阈值设置等功能，且功能正确易用。 5. 告警通知具备至少 2 种及以上方式，方便 DBA 及时接收到告警信息，比如：告警平台、告警箱、短信、Email 等

测试项目	死锁处理能力
测试目的	● 测试分布式数据库不同分片间事务之间存在死锁情况下的处理能力 ● 测试分布式数据库相同分片内事务间存在死锁情况下的处理能力
测试环境	分布式数据库系统测试环境
前置条件	不同分片间死锁处理能力 1. 分布式数据库系统运行正常 2. 创建两分片表 tb1，并向 tb1 表中插入数据 create table if not exists testdb.tb1（id int key，name varchar（10））； start transaction； insert into testdb.tb1（id，name）values（11，'abc'）； insert into testdb.tb1（id，name）values（12，'abc'）； insert into testdb.tb1（id，name）values（13，'abc'）； insert into testdb.tb1（id，name）values（14，'abc'）； insert into testdb.tb1（id，name）values（15，'abc'）； commit； 相同分片内死锁处理能力： 1. 分布式数据库系统运行正常 2. 存在表 tb1，并向 tb1 表中插入数据

	create table if not exists testdb.tb1（id int key，name varchar（10））； start transaction； insert into testdb.tb1（id，name）values（11，'abc'）； insert into testdb.tb1（id，name）values（12，'abc'）； insert into testdb.tb1（id，name）values（13，'abc'）； insert into testdb.tb1（id，name）values（14，'abc'）； insert into testdb.tb1（id，name）values（15，'abc'）； commit；
测试步骤	1. 开启一客户端执行 A 事务，如下： select * from tb1 where id=11 for update； update tb1 set name='bcd' where id=14； 2. 开启另一客户端执行 B 事务，如下： select * from tb1 where id=14 for update； update tb1 set name='bcd' where id=11； 3. 观察客户端上事务 A 和事务 B 的执行结果
预期结果	分布式数据库能够检测到发生了死锁，并在有限的时间内返回用户提示信息，不会导致永远死锁

测试项目	数据副本间数据一致性
测试目的	• 测试分布式数据库具备副本间数据保证一致性的能力
测试环境	分布式数据库系统测试环境
前置条件	1. 分布式数据库系统正常运行 2. 集群中各分片单元是一主多备的部署架构
测试步骤	1. 通过跑批工具进行业务跑批； 2. 跑批过程中构造分布式数据库数据备节点异常； 3. 恢复上述备节点异常； 4. 跑批过程中构造分布式数据库数据主节点异常； 5. 恢复上述主节点异常； 6. 停止业务跑批； 7. 查看跑批工具测试结果； 8. 校验各分片主备数据一致性
预期结果	1. 步骤 7，测试结果显示正常，无不一致； 2. 步骤 8，校验结果显示所有分片主备间数据均一致

测试项目	在线升级
测试目的	• 支持所有组件的在线升级，升级对业务无明显影响 • 是否显示升级进度

	• 考察升级效率
测试环境	分布式数据库系统测试环境
前置条件	1. 分布式数据库系统正常运行； 2. 获取升级版本包
测试步骤	1. 业务端工具保持业务正常运行； 2. 按照升级方案升级分布式数据库系统各个节点； 3. 记录升级过程对业务的影响
预期结果	1. 升级中可以查看升级进度，升级成功后，对应节点自动启动； 2. 升级失败，能够查看到升级失败的原因，能够回退到升级之前版本，保证业务能够正常恢复； 3. 能够查询之前所有的升级操作记录； 4. 升级过程对在线业务的影响甚小或无

测试项目	在线 DDL 操作
测试目的	是否支持在线建表及路由分片操作能力
测试环境	分布式数据库系统测试环境
前置条件	分布式数据库系统运行正常
测试步骤	1. 分布式数据库正常服务期间，开启某个业务，业务正常运行中； 2. 通过分布式数据库客户端执行如下 DDL 操作： create table if not exists testdb.tb2（id int key，name varchar（10））distributed by hash（id）（g1，g2）； 3. 对新建表执行如下 DML 操作： start transaction； insert into testdb.tb2（id，name）values（11，'abc'）； insert into testdb.tb2（id，name）values（12，'abc'）； insert into testdb.tb2（id，name）values（13，'abc'）； insert into testdb.tb2（id，name）values（14，'abc'）； insert into testdb.tb2（id，name）values（15，'abc'）； select * from testdb.tb2； update testdb.tb2 set name='bcd' where id=11； update testdb.tb2 set name='bcd' where id=14； select * from testdb.tb2； delete from testdb.tb2 where id>13； select * from testdb.tb2； commit； 4. 执行如下 alter 命令，增加一个字段： ALTER TABLE testdb.t2 ADD newcolum varchar（20）； 5. 执行如下 DML 命令：

	update testdb.tb2 set newcolum ='test123' where id=11； update testdb.tb2 set newcolum ='test123' where id=14； select * from testdb.tb2
预期结果	1. 测试步骤 2～5 语句均执行成功，结果正确； 2. 在线业务运行正常，不受影响

测试项目	管理操作
测试目的	是否支持常见的数据库管理操作
测试环境	分布式数据库系统测试环境
前置条件	分布式数据库系统运行正常
测试步骤	1. 通过分布式数据库客户端执行如下 DDL 操作： Create database if not exists testdb； create table if not exists testdb.tb3（id int key，name varchar（10））distributed by hash（id）（g1，g2）； 2. 通过客户端执行如下管理命令： show databases like '%test%'； show databases； show tables； desc testdb.tb3； show create table testdb.tb3； show distribution from testdb.tb3； show columns from testdb.tb3； show keys from testdb.tb3； 3. 执行数据库服务的启停操作
预期结果	1. 支持上述常见的数据库管理操作

测试项目	运维统计分析
测试目的	面向运维的友好的统计分析能力（当前系统的监控情况以及已经分配的资源情况）
测试环境	分布式事务数据库基准平台测试环境
前置条件	分布式事务数据库正常运行
测试步骤	1. 连接分布式数据库系统，执行批量数据库操作，各类 DDL、DML 等。 2. 查看分布式数据库系统的运维统计信息。 3. 检查平台能够提供接口，从而对接其他监控系统
预期结果	1. 支持面向运维的友好的统计分析能力。 支持以表格、趋势图、柱状图、饼图、Top SQL、TOP 事务等形式展示各对象统计信息

测试项目	查询日志及慢查询日志管理功能
测试目的	• 是否支持 general 日志功能 • 是否支持慢查询日志功能
测试环境	分布式数据库系统测试环境
前置条件	正常查询日志功能测试 1. 分布式数据库系统运行正常 2. 分布式数据库中存在表 a1 和 c1（a1 表分布到分片 1 和分片 2 上，c1 表分布到分片 2 和分片 3 上） DROP TABLE IF EXISTS testdb.a1； CREATE TABLE IF NOT EXISTS testdb.a1（a INT，sname CHAR（10）NOT NULL PRIMARY KEY，b INT，c INT NULL，d INT）； DROP TABLE IF EXISTS testdb.c1； CREATE TABLE IF NOT EXISTS testdb.c1（a INT，sname CHAR（10）NOT NULL PRIMARY KEY，b INT，score INT NULL，cname CHAR（20），c CHAR（10），d CHAR（10））； 3. a1 和 c1 表中插入数据： INSERT INTO testdb.a1（a，sname，b，c，d）VALUES （1，'a'，102，104，1）， （2，'b'，100，100，2）， （2，'c'，106，2，2）， （3，'d'，1，21，150）， （5，'e'，2，107，3）， （6，'f'，1，90，1）， （7，'ddd'，2，100，106）， （8，'dcd'，2，22，106）， （9，'dadd'，2，302，106）， （10，'dddf'，2，25，106）， （9000，'d vadd'，2，27，106）， （1000000，'dddedf'，2，2，106）； INSERT INTO testdb.c1（a，sname，b，score，cname，c，d） VALUES （1，'a'，1，100，'Class A'，'a'，'b'）， （2，'b'，102，100，'Class B'，'a'，'b'）， （3，'c'，103，100，'Class C'，'a'，'b'）， （4，'d'，1，100，'Class E'，'a'，'b'）， （5，'e'，2，80，'Class A'，'a'，'b'）， （6，'f'，1，90，'Class A'，'a'，'b'）， （7，'ddd'，4，NULL，'Class A'，'a'，'b'）， （9，'ff'，1，90，'Class A'，'a'，'b'），

	（10，'bb'，4，NULL，'Class A'，'a'，'b'）， （22，'bafw'，102，102，'Class B'，'a'，'b'）， （25，'bsdfwe'，102，106，'Class B'，'a'，'b'）； 4. 开启查询日志开关 慢查询日志功能测试 1. 分布式数据库系统运行正常 2. 开启慢查询日志开关 3. 配置慢 SQL 执行时间阈值为 0.01 秒； 错误日志功能测试： 执行数据库非法操作，查询日志中是否存在错误操作记录
测试步骤	正常查询日志功能测试： 1. 执行如下 DML 操作： Select * from testdb.a1 where sname=分片 1； Select * from testdb.c1 where sname=分片 3； SELECT testdb.a1.sname，testdb.c1.score FROM testdb.a1 JOIN testdb.c1 ON testdb.a1.a = testdb.c1.a； 2. 查看查询日志是否生成，日志内容是否正确； 慢查询日志功能测试： 1. testdb.a1 表中插入 1000W 数据 2. 构造慢查询事务，如下： Select * from testdb.a1 limit 10000； 3. 查看慢查询日志文件是否生成，日志内容是否正确
预期结果	1. 可查询到生成的通用查询日志，日志内容正确，分布式数据库具备全局 general 日志功能； 2. 可查询到生成的慢查询日志，日志内容正确，分布式数据库具备全局慢查询日志功能

测试项目	数据备份
测试目的	• 是否支持定时备份能力 • 是否支持实时备份能力 • 是否支持在线全量备份能力 • 是否支持在线增量备份能力 • 是否支持加密备份能力 • 是否支持压缩备份能力
测试环境	分布式数据库系统测试环境
前置条件	分布式数据库系统运行正常
测试步骤	定时备份能力测试： 1. 按照表结构预置 1000w 数据； 2. 设置定时备份任务（例如周日全量备份，周一到周六增量备份）； 3. 修改系统时间触发定时备份任务；

	4. 查看备份文件是否符合预期。 实时备份能力测试： 1. 登录分布式数据库备份管理页面； 2. 发起实时备份； 3. 查看功能是否正确； 在线全量备份能力测试： 1. 在线业务持续运行过程中，发起实时全量备份； 2. 检查备份结果是否成功； 在线增量备份能力测试： 1. 在线业务持续运行过程中，发起实时全量备份； 2. 检查备份结果是否成功； 加密备份能力测试： 1. 按照表结构预置 1000w 数据； 2. 执行加密全量备份操作完成加密全量备份； 3. 向表中导入 1000 条数据； 4. 执行加密增量备份操作完成加密增量备份； 5. 检查备份文件加密情况。 压缩备份能力测试： 1. 按照表结构预置 1000w 数据； 2. 执行压缩全量备份操作完成压缩全量备份； 3. 向表中导入 1000 条数据； 4. 执行压缩增量备份操作完成压缩增量备份； 5. 检查备份文件压缩情况
预期结果	支持备份功能

测试项目	数据恢复
测试目的	• 测试分布式数据库数据恢复的一致性 • 测试使用全量备份和增量备份文件恢复到任意一个时间点
测试环境	分布式数据库系统测试环境
前置条件	1. 分布式数据库系统运行正常
测试步骤	1. 按照表结构预置 1000w 数据； 2. 执行全量备份操作完成全量备份； 3. 向表中导入 1000 条数据； 4. 执行增量备份操作完成增量备份； 5. 增量备份完成后 2 分钟，开启事务 A 向表中插入 1000 条数据，并 commit； 6. 增量备份完成后 5 分钟，开启事务 B 向表中插入 1000 条数据，不 commit； 7. 增量备份完成后 15 分钟，开启事务 c 向表中插入 1000 条数据，并 commit；

	8. 利用全量备份文件和增量备份文件将 DB 恢复到增量备份结束后 8 分钟时间内； 9. 查看恢复后 DB 数据，数据量应为 10002000，事务 A 数据存在，事务 B 和事务 C 数据均不存在
预期结果	支持按时间点进行恢复，并保证数据的全局一致性

测试项目	集群内资源隔离能力
测试目的	• 是否具备对集群内的数据库进行资源分组配置的能力
测试环境	分布式数据库系统测试环境
前置条件	1. 分布式数据库系统运
测试步骤	1. 构造两个业务，分别拥有自己的库表； 2. 配置集群，使不同集群能够从物理或操作系统层级进行资源隔离
预期结果	分布式数据库具备资源分组配置的能力

高可用能力测试方法

测试项目	硬件故障
测试目的	测试系统在发生硬件故障时的数据服务能力，包括电源插拔，硬盘故障，插拔网线，交换机、机架、机房故障等
测试环境	分布式事务数据库测试环境
前置条件	1） 测试环境下完成产品部署，按照同城双活模式配置； 2） 准备好测试客户端，创建好库表； 3） 执行背景业务，即创建一个写线程，每隔 5 秒插入 1000 条数据，持续插入
测试步骤	1） 通过测试客户端，持续写入数据； 2） 关闭一台参与业务的节点服务器，验证系统是否能够自动检测故障和恢复服务； 3） 在一台参与业务的节点上模拟磁盘损坏，验证系统是否能够自动检测故障和恢复服务； 4） 通过 iptables，断开一台参与业务的节点的网络，验证系统是否能够自动检测故障和恢复服务； 5） 检查客户端数据写入日志，检测写入数据的一致性
预期结果	1） 产品能够自动检测电源、磁盘和网络等硬件故障，并能够自动恢复服务； 2） 产品能够自动检测机房故障，在主机房发生故障时，备用机房能够完全接管业务提供数据服务； 3） 业务不中断，已经写入的数据完全一致，且最后插入的数据应该为 1000 的倍数

测试项目	操作系统故障
测试目的	测试系统在发生操作系统故障时的数据服务能力，包括操作系统损坏、操作系统资源占用过高等
测试环境	分布式事务数据库测试环境
前置条件	1. 测试环境下完成产品部署，按照同城双活模式配置； 2. 准备好测试客户端，创建好库表； 3. 执行背景业务，即创建一个写线程，每隔 5 秒插入 1000 条数据，持续插入

测试步骤	1. 通过测试客户端，持续写入数据； 2. 重启一台参与业务的节点服务器，验证系统是否能够自动检测故障和恢复服务； 3. 在一台参与业务的节点上通过拷贝大文件，模拟磁盘 IO 过载，验证系统是否能够自动检测故障和恢复服务； 4. 在一台参与业务的节点上通过启动大程序模拟内存过载，验证系统是否能够自动检测故障和恢复服务； 5. 通过 dd 命令，将一台参与业务的节点的磁盘写满，验证系统是否能够自动检测故障和恢复服务，允许产生告警和服务失败，但不允许产生错误； 6. 检查客户端数据写入日志，检测写入数据的一致性
预期结果	1. 产品能够自动检测操作系统故障，并能够自动恢复服务； 2. 已经写入的数据完全一致； 3. 大部分业务不中断，已经写入的数据完全一致，且最后插入的数据应该为 1000 的倍数

测试项目	数据库服务故障
测试目的	测试系统在发生数据库服务故障时的数据服务能力，包括进程崩溃、数据文件损坏等。
测试环境	分布式事务数据库测试环境
前置条件	1） 测试环境下完成产品部署； 2） 准备好测试客户端，创建好库表； 3） 执行背景业务，即创建一个写线程，每隔 5 秒插入 1000 条数据，持续插入
测试步骤	1） 通过测试客户端，持续写入数据； 2） 杀掉一台存储节点的数据库服务进程，验证系统是否能够自动检测故障和恢复服务； 3） 通过 kill -9，杀掉一台计算节点的数据库服务进程，验证系统是否能够自动检测故障和恢复服务； 4） 检查客户端数据写入日志，检测写入数据的一致性
预期结果	1） 产品能够自动检测数据库服务故障，并能够自动恢复服务； 2） 已经写入的数据完全一致； 3） 大部分业务不中断，已经写入的数据完全一致，且最后插入的数据应该为 1000 的倍数

测试项目	上层应用故障
测试目的	测试数据库系统与客户端传输过程中发生故障时的数据服务能力
测试环境	分布式事务数据库测试环境
前置条件	1） 测试环境下完成产品部署； 2） 准备好测试客户端，创建好库表
测试步骤	1） 通过测试客户端，持续写入数据； 2） 启动另一个测试客户端，持续写入数据； 3） 利用 iptables，断开一台测试客户端和数据服务之间的网络，检测写入数据的一致性； 4） 利用 tc，将一台测试客户端和数据服务之间的网络延迟设置为 1s 和 100s，检测写入数据的一致性；

	5） 杀掉一台测试客户端的进程，检测写入数据的一致性； 6） 重新启动测试客户端，持续写入数据，检测写入数据的一致性
预期结果	1） 上层应用故障时，系统能够持续提供服务； 2） 已经写入的数据完全一致

扩展性能力测试方法

测试项目	计算均衡能力
测试目的	验证分布式事务数据库具备将接入数据库的计算压力进行平衡的能力
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 运行高并发测试程序，例如 TPCC，并发程度大约为 100 并发/台规模； 2） 展示各节点连接数和 CPU 使用率
预期结果	各节点计算压力平衡

测试项目	数据按需均衡能力
测试目的	验证分布式事务数据库具备将数据库内的数据进行平衡的能力，并能够评测
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 插入大量数据； 2） 查询系统试图展示各节点存储的数据量分布情况
预期结果	数据平衡分布

测试项目	集群在线扩容能力
测试目的	测试系统具备在线扩容的能力，并且扩容后性能可随之提升
测试环境	分布式事务数据库测试环境
前置条件	1. 测试环境下完成产品部署，按照同城双活模式配置； 2. 准备好测试客户端，创建好库表； 3. 准备好新的计算节点和存储节点
测试步骤	1. 通过测试客户端，开启一个线程，每 10s 持续写入 1000 条数据； 2. 增加一台存储节点，检查数据是否自动重新分布，检查扩容前后数据是否一致，数据写入性能是否提升，记录扩容过程中服务中断的时间； 3. 增加一台计算节点，检查服务是否中断，检查数据写入的性能是否提升
预期结果	1. 系统可在线扩充存储节点，扩容过程中服务中断时间为秒级，数据自动重新分布，扩容之后性能随之提升； 2. 系统可在线扩充计算节点，扩容过程中服务无中断，扩容后性能随之提升；

	3. 数据表行数在查询验证时应一直为 1000 行的倍数

测试项目	集群在线缩容能力
测试目的	验证分布式事务数据库集群具备集群缩减能力
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行 TPC-H 数据加载完毕
测试步骤	1） 准备分布表 lineitem，记录 lineitem 表的行数； 2） 准备数据文件 lineitem.csv，记录 lineitem.csv 文件行数； 3） 准备 insert.sql 脚本，内容为 10000 条 insert values 操作，向 lineitem 表插入数据； 4） 对 lineitem 表执行缩容操作； 5） 缩容过程中加载 lineitem.csv 文件到 lineitem 表； 6） 缩容过程中，每间隔 20 秒执行一次 SQL-1； 7） 缩容过程中，执行 insert.sql 脚本； 8） 缩容完成后，检查结果是否正确
预期结果	数据平衡分布

安全能力测试方法

测试项目	权限验证
测试目的	1） 针对特定用户赋予或收回相应的集群操作权限； 2） 权限可以分级赋予：库、表； 3） 权限可以分类型赋予：create、drop、alter、update、select、delete、usage、grant option 等，容许有个别不支持
测试环境	分布式数据库环境
前置条件	分布式数据库产品部署正常且正常运行
测试步骤	1） 通过客户端用管理员用户登录分布式事务数据库集群； 2） 执行备注中 SQL 语句或其他自定义 SQL 语句，对用户进行权限测试
预期结果	1） SQL 执行成功； 2） 返回正确结果

测试项目	身份认证
测试目的	是否支持对接入数据库用户进行身份认证工作（用户名密码验证）
测试环境	分布式数据库系统测试环境
前置条件	1. 分布式数据库系统运行正常； 2. 创建用户 user1 并设置密码；

测试步骤	1. 开启客户端 1，输入正确的 user1 用户名和密码访问集群； 2. 开启客户端 2，输入错误的用户名或密码访问集群；
预期结果	1. 客户端 1 成功访问集群，客户端 2 报错，不能访问集群 2. 支持通过用户名和密码的方式对接入数据库的用户进行身份认证

测试项目	操作审计
测试目的	是否支持对数据库内操作进行审计工作
测试环境	分布式数据库系统测试环境
前置条件	1. 分布式数据库系统运行正常； 2. 开启审计功能
测试步骤	1. 验证分布式数据库具备新增删除用户、访问权限控制、用户权限变更、登录退出等审计信息； 2. 执行任意的 SQL 任务，在审计日志文件中查看跟踪日志信息。日志信息内容包括：执行时间、操作的用户、连接的 IP、执行操作的数据库和 SQL 语句等； 3. 验证分布式数据库具备启动、停止、更改数据库参数，DDL 操作等审计信息
预期结果	1. 分布式数据库具备操作的审计验证功能； 2. 审计日志中记录详细的用户操作信息，打印的审计内容准确，清晰； 3. 审计内容支持灵活的配置方式

测试项目	流量控制能力
测试目的	• 验证是否能够配置数据库的流量上限
测试环境	分布式数据库系统测试环境
前置条件	1. 分布式数据库系统运行正常； 2. 配置客户端最大连接数为 5
测试步骤	1. 开启 5 个客户端，连接分布式数据库执行相关操作； 2. 开启第 6 个客户端，连接分布式数据库
预期结果	1. 步骤 1 事务正常，步骤 2 报错； 2. 分布式数据库具备流量控制功能，且功能正确

附 录 A
测试参考代码
（资料性附录）

A.1 数据类型测试参考脚本代码示例

a） CREATE TABLE TYPE001（COL INTEGER）；
b） CREATE TABLE TYPE002（COL INT）；
c） CREATE TABLE TYPE003（COL SMALLINT）；
d） CREATE TABLE TYPE004（COL BIGINT）；
e） CREATE TABLE TYPE005（COL REAL）；
f） CREATE TABLE TYPE006（COL DOUBLE）；
g） CREATE TABLE TYPE007（COL FLOAT（30））；
h） CREATE TABLE TYPE008（COL DECIMAL（18，10））；
i） CREATE TABLE TYPE009（COL NUMERIC（18，10））；
j） CREATE TABLE TYPE010（COL DATE）；
k） CREATE TABLE TYPE011（COL TIMESTAMP）；

A.2 操作符测试参考脚本代码示例

a） SELECT * FROM TYPE016 WHERE COL0=COL1；
b） SELECT * FROM TYPE016 WHERE COL0<>COL1；
c） SELECT * FROM TYPE016 WHERE COL0<COL1；
d） SELECT * FROM TYPE016 WHERE COL0<=COL1；
e） SELECT * FROM TYPE016 WHERE COL0>COL1；
f） SELECT * FROM TYPE016 WHERE COL0>=COL1；

A.3 函数测试参考脚本代码示例

a） SELECT ABS（COL）FROM TYPE017；
b） SELECT AVG（COL）FROM TYPE017；
c） SELECT COUNT（COL）FROM TYPE017；
d） SELECT SUM（COL）FROM TYPE017；
e） SELECT MAX（COL）FROM TYPE017；
f） SELECT MIN（COL）FROM TYPE017；

A.4 DML测试参考脚本代码示例

a） CREATE TABLE T1101（COL INT，COL1 CHAR（10））；
b） CREATE TABLE T1102（COL INT，COL1 CHAR（10））；
c） INSERT INTO T1102 VALUES（100，'ABCD'）；

d） CREATE TABLE T1103（COL INT DEFAULT 99，COL1 CHAR（10））；

e） INSERT INTO T1101 VALUES（1，'TESTING'）；

f） SELECT * FROM T1101；

g） INSERT INTO T1101 SELECT * FROM T1102；

h） SELECT * FROM T1101；

i） INSERT INTO T1103 VALUES（DEFAULT，'DATABASE'）；

j） SELECT * FROM T1103；

k） UPDATE T1101 SET COL1='ABC ABC' WHERE COL>10；

l） SELECT * FROM T1101；

m） DELETE FROM T1101 WHERE COL>10；

n） SELECT * FROM T1101；

o） DROP TABLE T1101；

p） DROP TABLE T1102；

q） DROP TABLE T1103；

A.5 表连接测试参考脚本代码示例

a） CREATE TABLE T81（bookname char（10），price float）；

b） INSERT INTO T81 VALUES（'delphi'，50.11）；

c） CREATE TABLE T82（bookname char（10），author char（10））；

d） INSERT INTO T82 VALUES（'java'，'tom'）；

e） SELECT AUTHOR，PRICE FROM T81 INNER JOIN T82 ON
T81.BOOKNAME=T82.BOOKNAME；

f） SELECT AUTHOR，PRICE FROM T81 LEFT OUTER JOIN T82 ON
T81.BOOKNAME=T82.BOOKNAME；

g） SELECT AUTHOR，PRICE FROM T81 RIGHT OUTER JOIN T82 ON
T81.BOOKNAME=T82.BOOKNAME；

h） SELECT AUTHOR，PRICE FROM T81 INNER JOIN T82 ON
T81.BOOKNAME=T82.BOOKNAME；

i） SELECT AUTHOR，PRICE FROM T81 INNER JOIN T82 ON
T81.BOOKNAME<>T82.BOOKNAME；

j） SELECT AUTHOR，PRICE FROM T81 INNER JOIN T82 ON
T81.BOOKNAME<T82.BOOKNAME；

k） SELECT AUTHOR，PRICE FROM T81 INNER JOIN T82 ON
T81.BOOKNAME<=T82.BOOKNAME；

l） SELECT AUTHOR，PRICE FROM T81 INNER JOIN T82 ON
T81.BOOKNAME>T82.BOOKNAME；

m） SELECT AUTHOR，PRICE FROM T81 INNER JOIN T82 ON T81.BOOKNAME>=T82.BOOKNAME；

n） DROP TABLE T81；

o） DROP TABLE T82；

A.6 子查询测试参考脚本代码示例

a） CREATE TABLE T901（col int，col1 char（10））；

b） INSERT INTO T901 VALUES（1，'t1-one'）；

c） CREATE TABLE T902（col2 int，col3 char（20））；

d） INSERT INTO T902 VALUES（10，'t2-first'）；

e） CREATE TABLE T903（id CHAR（5）NOT NULL，name CHAR（20），total NUMERIC（10，2）DEFAULT 10.00，addr CHAR（10））；

f） INSERT INTO T903 VALUES（'B0A01'，'Huale'，50，'BEIJING'）；

g） CREATE TABLE T904（itemid CHAR（8）NOT NULL，itemname CHAR（10），price NUMERIC（10，2），warehouse CHAR（10），id CHAR（5））；

h） INSERT INTO T904 VALUES（'C0001'，'TV'，3000，'WUHAN'，'B0A01'）；

i） SELECT id，name，addr FROM T903 WHERE id =（SELECT id FROM T904 WHERE itemid='C0001'）；

j） SELECT id，name，addr FROM T903 WHERE total<（SELECT price FROM T904 WHERE itemid='C0001'）；

k） SELECT id，name，addr FROM T903 WHERE total<=（SELECT price FROM T904 WHERE itemid='C0001'）；

l） SELECT id，name，addr FROM T903 WHERE total>（SELECT price FROM T904 WHERE itemid='C0001'）；

m） SELECT id，name，addr FROM T903 WHERE total>=（SELECT price FROM T904 WHERE itemid='C0001'）；

n） SELECT id，name，addr FROM T903 WHERE total<>（SELECT price FROM T904 WHERE itemid='C0001'）；

o） DROP TABLE T901；

p） DROP TABLE T902；

q） DROP TABLE T903；

r） DROP TABLE T904；

A.7 临时表测试参考脚本代码示例

a） create database if not exists test；

b） use test；

c） drop temporary table if exists temp；

d） create temporary table temp（a int）；

e） insert into temp values（5）；

f） select * from temp；

g） insert into temp select * from temp；

h） drop temporary table temp；

i） create temporary table temp（a int，b varchar（20））；

j） show create table temp；

k） insert into temp values（1，'a1'），（2，'2a'），（3，'3b'），（4，'4c'），（50，'abcdefgh'）；

l） select * from temp order by a；

m） update temp set a=a+1；

n） select * from temp order by a；

o） delete from temp；

p） select * from temp order by a；

q） drop temporary table temp；

r） create temporary table temp（a int，b varchar（20））；

s） show create table temp；

t） insert into temp values（1，'a1'），（2，'2a'），（3，'3b'），（4，'4c'），（50，'abcdefgh'）；

u） select * from temp order by a；

v） truncate table temp；

w） select * from temp order by a；

x） drop temporary table temp；

A.8 索引测试参考脚本代码示例

a） drop table if exists t；

b） create table t（a varchar（255），b char（9））；

c） create fulltext index in_1 on t（a）；

d） create fulltext index in_2 on t（b）；

e） update index in_1 on t；

f） update index in_2 on t；

g） update index in_1 on t；

h） update index in_2 on t；

i） drop table if exists t2；

j） create table t2 like t；

k） insert into t2 select a，b from t；

l） update index in_1 on t2；

m） update index in_2 on t2；

n） drop table if exists t；

o） drop table if exists t2；