



# 中华人民共和国金融行业标准

JR/T XXXXX—XXXX

## 证券期货业分布式关系型数据库稳定性治理指南

Guidance on stability governance for distributed relational database in  
the securities and futures industry

(征求意见稿)

XXXX-XX-XX 发布

XXXX-XX-XX 实施

中国证券监督管理委员会 发布



目 次

前言 ..... III

引言 ..... IV

1 范围 ..... 1

2 规范性引用文件 ..... 1

3 术语和定义 ..... 1

4 缩略语 ..... 3

5 概述 ..... 3

6 人员管理 ..... 4

    6.1 人员体系 ..... 4

    6.2 制度文件 ..... 4

    6.3 组织团队 ..... 4

7 数据库表规范 ..... 5

    7.1 数据表建立 ..... 5

    7.2 数据类型 ..... 6

    7.3 主键 ..... 7

    7.4 外键 ..... 7

    7.5 索引 ..... 7

    7.6 注释 ..... 8

8 编程规范 ..... 8

    8.1 DQL 操作 ..... 8

    8.2 DML 操作 ..... 9

    8.3 DDL 操作 ..... 10

    8.4 SQL 调优操作 ..... 10

    8.5 分布式事务编程 ..... 11

    8.6 异常处理 ..... 12

9 运维管理 ..... 12

    9.1 总体原则 ..... 12

    9.2 运维系统功能要求 ..... 12

    9.3 运维系统管理 ..... 14

    9.4 运维灾备演练 ..... 14

10 故障分类与检测 ..... 14

    10.1 故障分类 ..... 15

    10.2 故障分级 ..... 16

    10.3 故障检测 ..... 17

11 故障处理 ..... 18

    11.1 概述 ..... 18

    11.2 事件感知 ..... 18

    11.3 异常信息收集 ..... 18

    11.4 故障原因分析 ..... 19

|                            |    |
|----------------------------|----|
| 11.5 决策处理 .....            | 20 |
| 11.6 复盘与评估 .....           | 20 |
| 附录 A（规范性） .....            | 22 |
| A.1 数据表限制 .....            | 22 |
| A.2 指标阈值 .....             | 22 |
| A.3 灾备架构部署参考规范 .....       | 23 |
| A.4 运维演练技术规范 .....         | 23 |
| A.5 核心业务场景强制性指标基线 .....    | 23 |
| 附录 B（资料性）典型 SQL 调优示例 ..... | 25 |
| B.1 低效 SQL 请求优化 .....      | 25 |
| B.2 索引使用与失效场景 .....        | 26 |
| B.3 JOIN 优化 .....          | 27 |
| 参考文献 .....                 | 28 |

## 前 言

本文件按照GB/T 1.1—2020《标准化工作导则 第1部分：标准化文件的结构和起草规范》的规定起草。

本文件由全国金融标准化技术委员会证券分技术委员会（SAC/TC 180/SC4）提出。

本文件由全国金融标准化技术委员会（SAC/TC 180）归口。

本文件起草单位：中国证券登记结算有限责任公司、上海证券交易所、深圳证券交易所、中国金融期货交易所、中信证券股份有限公司、国泰海通证券股份有限公司、天弘基金管理有限公司、北京奥星贝斯科技有限公司、华为云计算技术有限公司。

本文件主要起草人：崔朝、和冲宇、卢向澄、鞠学祯、李兴民、吕国豪、刘佳航、韩凤宁、王帆、王明发、骆毅、刘传友、路瑞超、王栩、王鑫。

## 引 言

随着证券期货行业数字化转型的深入推进，业务系统对数据存储的高并发、高可用、强一致性和弹性扩展能力提出了更高要求，分布式关系型数据库是交易、清算、风控等关键系统的底层支撑，也是数据查询、统计、分析和存储类系统的重要组成，已在行业内广泛应用。在实践过程中，行业机构在分布式数据库的稳定性治理方面积累了丰富经验，但也暴露出标准不统一、规范缺失、处理流程混乱、跨团队协作效率低等问题，影响了系统的稳定性、安全性和可维护性。为统一技术与管理要求、规范开发行为、提升应急能力、保障业务连续性，亟需建立覆盖“设计→开发→运维”全生命周期的标准化体系。通过遵循本文件，可有效提升应用程序的稳定性与性能表现，助力企业构建健壮、灵活的金融数据架构，满足监管合规要求，为业务的持续创新与发展奠定坚实的技术基础。本文件融合了行业最佳实践，以促进证券期货业相关技术的规范化应用，推动行业信息化水平的提升。

# 证券期货业分布式关系型数据库稳定性治理指南

## 1 范围

本文件规定了证券期货行业在应用分布式关系型数据库过程中，涵盖应用开发和故障管理处理的全生命周期技术与管理要求，包括人员管理、数据表设计、利用结构化查询语言（SQL）编写数据库应用、建立故障分类与检测标准以及应急处理策略等内容。

本文件作为证券期货行业分布式关系型数据库系统应用建设、质量评估、代码审查、运行保障与故障处置工作的统一基础依据和参考标准。

本文件适用于证券期货行业内从事分布式关系型数据库应用设计、开发、运维、应急响应等工作的技术人员和管理人员，适用范围覆盖数据库系统从设计开发到运行维护、故障处理的全过程。

## 2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包含所有的修改单）适用于本文件。

GB/T 12991.1—2008 信息技术 数据库语言SQL 第一部分：框架  
GB/T 20273—2019 信息安全技术 数据库管理系统安全技术要求  
GB/T 36626—2018 信息安全技术 信息系统安全运维管理指南  
JR/T 0059—2024 证券期货业信息系统备份能力规范  
JR/T 0205—2020 分布式数据库技术金融应用规范 灾难恢复要求

## 3 术语和定义

下列术语和定义适用于本文件。

GB/T 12991.1—2008、JR/T 0203—2020与JR/T 0205—2020界定的以及下列术语和定义适用于本文件。为了便于使用，以下重复列出了JR/T 0203—2020某些术语和定义。

### 3.1

**事务 transaction**

一组以原子性、一致性、持久性和隔离性为特征的相关操作。

[来源：JR/T 0203—2020，3.2]

### 3.2

**租户 tenant**

共享同一数据库系统，但拥有独立资源和数据的逻辑隔离单元，可被视为独立的用户和组织。

### 3.3

**数据类型 data type**

可表示的值的集合。

[来源：GB/T 12991.1—2008，3.1.1.3]

### 3.4

**联机分析处理 On-Line Analytical Processing**

基于多维数据模型，通过复杂查询、数据聚合和快速响应技术，为决策支持、商业智能等需求提供实时数据分析的环境。

### 3.5

#### 索引 index

一种数据结构，用于提高数据库表中数据的检索效率。

### 3.6

#### 隔离级别 isolation level

用于控制并发事务之间相互作用程度的一种机制，表示事务在读取和修改数据时与其他事务的隔离程度。

### 3.7

#### 并发 concurrency

系统中多个计算任务在同一时间段内交替执行的能力。

### 3.8

#### 全表扫描 full table scan

数据库引擎在执行查询时，从头到尾地读取整张表中的所有数据行，逐一检查是否满足查询条件。

### 3.9

#### 连接池 connection pool

在数据库应用系统中，用于管理和复用数据库连接的资源池。连接池预先创建并维护一定数量的数据库连接，供应用程序按需获取和归还，以减少频繁建立和断开连接的开销，提升系统性能与资源利用率。

### 3.10

#### 实时信息系统 real-time information system

对业务连续运行要求很高的信息系统。

[来源：JR/T 0059—2024, 3.2]

### 3.11

#### 负载均衡 load balancing

通过特定的算法将同一任务分摊到多个操作单元上执行，使用健康检测机制保证调度合理性的技术。

[来源：GA/T 1726—2020, 3.1]

### 3.12

#### 热点行 hot row

在数据库中被频繁访问或更新的数据行。

### 3.13

#### 分片 sharding

指将数据库中的同一张表（或数据集）的数据，按照某种特定的规则（分片键），水平地拆分并存储到多个不同的物理数据库节点（服务器）上，从而将原本由单台机器承担的存储和计算压力分散到整个集群中。

### 3.14

#### 抖动 jitter

指网络中数据包传输延迟的变化程度，是衡量网络稳定性的重要指标。

### 3.15

#### 主键 primary key

指表中的一个或多个字段，其值用于唯一地标识表中的每一条记录，是数据库设计中用于确保数据

唯一性的约束条件。

4 缩略语

- 下列缩略语适用于本文件。
- SQL: 结构化查询语言 (Structured Query Language)
  - DDL: 数据定义语言 (Data Definition Language)
  - DQL: 数据查询语言 (Data Query Language)
  - DML: 数据操纵语言 (Data Manipulation Language)
  - RPO: 灾难恢复时允许的最大数据丢失时间 (Recovery Point Objective)
  - RT0: 业务系统允许的最长停机恢复时间 (Recovery Time Objective)
  - TPS: 每秒事务处理量 (Transactions Per Second)
  - ART: 平均响应时间 (Average Response Time)
  - NTP: 网络时间协议 (Network Time Protocol)
  - IO: 输入和输出 (Input Output)
  - ACID: 原子性、一致性、隔离性、持久性 (Atomicity Consistency Isolation Durability)
  - OLTP: 联机事务处理 (On-Line Transaction Processing)
  - OLAP: 联机分析处理 (On-Line Analytical Processing)
  - ETL: 抽取、转换、加载三部分核心数据处理技术 (Extract-Transform-Load)
  - CPU: 中央处理器 (Central Processing Unit)

5 概述

本文件规定了证券期货行业分布式关系型数据库使用的全生命周期稳定性治理规范,适用于采用分布式关系型数据库的行业机构在表设计、开发运维和故障管理三个阶段的技术实践。本文件主要内容包括:人员管理要求、数据库表设计要求、应用编程要求、运行维护要求,故障分类与检测要求、故障处理要求。其中,人员管理要求贯穿全生命周期各个阶段,数据库表设计要求适用于表设计阶段,编程与运维管理要求适用于开发运维阶段,故障分类与检测、故障处理要求适用于故障管理阶段。稳定性治理框架图如图1所示:

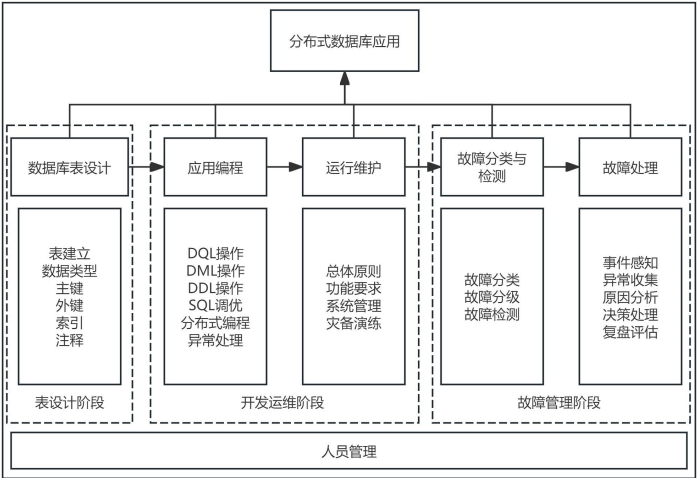


图 1 分布式数据库稳定性治理框架

## 6 人员管理

### 6.1 人员体系

- a) 机构应建立与分布式关系型数据库稳定性治理目标相适应的人员管理体系，明确组织架构、岗位职责、协作机制和能力要求；
- b) 人员管理体系应覆盖分布式关系型数据库的规划、选型、部署、运维、监控、优化及应急处置等全生命周期阶段；
- c) 机构应确保数据库管理相关岗位人员具备与其职责相匹配的专业能力，并定期开展培训与考核。

### 6.2 制度文件

机构应制定并维护数据库人员管理相关制度文件，宜包括：

- a) 岗位职责说明书，明确各角色的权限范围和责任边界；
- b) 人员准入与退出管理制度，规范数据库相关系统的访问权限生命周期管理；
- c) 培训与考核制度，定期组织分布式关系型数据库技术培训、应急演练及能力评估；
- d) 值班与应急管理制度，明确数据库故障场景下的值班安排、升级路径和响应时效要求。

人员管理相关的制度文件应定期评审和修订，当发生重大技术架构变更或组织架构调整时，应及时补充修订。

### 6.3 组织团队

#### 6.3.1 架构设计

为保障分布式关系型数据库系统的稳定性、安全性和高效运行，宜构建“决策层-执行层-支撑层”三级联动的组织架构，明确各层级职责边界与协作机制：

- a) 决策层宜由研发、测试、运维、质量保障及信息安全等相关职能部门的骨干人员组成，形成跨部门的联合治理工作组，协调跨部门工作的同时，审批重大架构变更、容灾演练计划及应急预案；
- b) 执行层负责分布式关系型数据库应用的落地与维护，建议建设“纵向+横向”的双团队架构模式。纵向团队由研发与运维部门的骨干组成，聚焦应用系统、数据库实例的纵向研发，横向团队由测试、研发及质量保障人员构成，致力于流程优化与工具建设；
- c) 支撑层由质量保障与信息安全的骨干人员组成，提供独立的安全监督与质量度量，确保合规性与数据完整性。

#### 6.3.2 能力要求

- a) 核心研发人员应掌握分布式关系型数据库原理等技术，熟悉数据库应用的业务逻辑；
- b) 运维人员具备在高压环境下快速定位故障根因的能力，熟练掌握故障隔离、流量切换及数据修复流程，同时具备使用运维开发平台、编写自动化脚本及分析运维数据的能力；
- c) 质量保障人员应具备一定的数据库软件设计、研发与测试能力，贯彻安全左移的稳定性治理思路，在应用架构设计初期即开始建立安全审计；
- d) 质量保障人员应具备数据完整性校验与数据一致性比对能力，能够针对大批量数据处理场景（如批量计算结果验证、历史数据回测校验等）设计并实施数据质量验证方案。

#### 6.3.3 团队管理

- a) 数据库权限应遵循“最小授权”与“职责分离”原则。严格划分数据库系统管理员、安全审计员与应用运维管理员的职责，原则上不允许同一人兼任多重角色，并按岗位实际需要授予最小必要权限；
- b) 建立生产环境、验证环境与开发测试环境分离的安全原则。开发人员不得直接操作生产环境数据库，确需查询生产环境数据的，须经审批后通过堡垒机或运维审计系统进行全程留痕操作；
- c) 数据库应用核心岗位应建立AB角互备机制，确保在关键人员缺岗时业务连续运行不受影响；
- d) 完善常态化培训机制，宜定期组织分布式数据库技术分享会、故障复盘会及行业前沿技术培训。

## 7 数据库表规范

### 7.1 数据表建立

#### 7.1.1 基础原则

- a) OLTP场景下，活跃业务数据与历史数据宜分表或分库隔离存储，并依据业务及合规要求定期清理、归档超期历史数据，控制主表体量，保障事务访问性能。OLAP场景下，不建议简单拆分当前与历史数据独立建表，优先采用分区、冷热分层进行管理；历史数据以长期留存为主，优先通过明细聚合降维、冷数据低成本存储优化资源，仅清理超保存期限且无分析回溯价值的历史数据；
- b) 表设计时，宜增加预留字段；
- c) OLTP场景下，避免大宽表和超大单行，单行宜控制在1兆字节以内。OLAP场景下允许宽表建模，单行长度按附录A.1执行；
- d) 证券期货业务涉及多语言大小写敏感场景，应明确字符集与排序规则，同时避免混合字符集；
- e) 库名与应用名称宜尽量一致；
- f) 字段允许适当冗余，以提高查询性能；
- g) 对交易流水等大表，设计时考虑分区策略（如按月分区）便于归档；
- h) 敏感字段（如客户身份证号）应加密存储，建表时预留加密字段或标记；
- i) 对于OLTP场景，不宜在分布式数据库中存放业务逻辑，包括存储过程、触发器、视图、自定义函数、定时任务等。对于OLAP场景，可根据分析需要使用视图和自定义函数（如窗口函数）封装通用分析逻辑，但仍不宜使用存储过程和触发器。

#### 7.1.2 数据分片设计要求

分布式关系型数据库通过数据分片将表的数据按规则分布到多个数据节点上，以实现负载均衡、提升并行处理能力和突破单机资源瓶颈。分片设计应综合考虑业务场景、查询模式、数据增长趋势和运维需求。

部分分布式关系型数据库支持设置分片数。主键宜包含分片键的所有列。对于业务主表，分片数量宜为提供读写服务的数据节点的2-3倍，便于负载均衡。分片后，单个分片宜不超过2000万条记录，物理容量不超过20吉字节。对于联机分析场景，分片数量宜在单机CPU总核数到提供读写服务的数据节点的CPU总核数之间。

分片条件要求如下：

- a) 如果单表记录数或数据量超过一定规模，经评估无法满足运行性能要求（包括TPS和交易响应时间），宜进行分片；
- b) 对于经常执行并行操作（如并行插入或并行修改）的表，宜进行分片，提高数据并行处理效率；
- c) 分片表的规则在表创建时需要指定；
- d) 分片设计需结合总体应用逻辑架构设计成果，满足架构清晰、性能优化的综合要求。

分片键选择要求如下：

- a) 分片键尽量包含均匀分布的字段，推荐选择高基数字段，避免值的种类过少。分片键宜选择后续查询高频的字段；
- b) 关于分片键在多维业务查询场景下的选择，如证券账号和证券代码同时存在情况下，需根据业务使用频率和业务重要性等维度来考虑分片键的选择；
- c) 建议每张表上都必须设定主键，主键必须在建表语句中定义；
- d) 禁止使用自增列作为分片键；
- e) 在进行数据分片时，根据具体的业务场景选择合适的分片键。

分片策略建议如下：

- a) 哈希分片、键分片适合随机分布的数据，哈希算法会将分片列的数据进行重分布使得相近的值被打散到不同分片，如以账户作为哈希分片键；
- b) 列表分片适合分片值明确的情形，如账户信息表中以区域字段作为列表分片列；
- c) 范围分片适合处理相似的、与时间有关的数据（业务流水表），宜定期导入新数据、删除历史数据的场景；
- d) 对于数据量持续增加、需定期清理的表，宜结合数据清理策略进行分片；
- e) 对于需要经常按时间段成批删除数据的表，需要根据维护需要进行分片；

### 7.1.3 表格大小限制

单表限制及标识符长度限制可参考附录A.1。

### 7.1.4 连接要求

连接数据库相关要求如下：

- a) 应使用配套版本的驱动连接数据库服务器；
- b) 应使用统一的数据源配置；
- c) 使用连接池的时候宜设置一个最大的连接空闲时间；
- d) 前端程序连接数据库或者数据库代理层，要具备失败重连机制，且应有间隔时间；
- e) 程序端日志应记录连接数据库的标准以及连接的数据库信息；
- f) 对于有连接池的前端程序，应根据业务需要配置初始、最小和最大连接数；
- g) OLTP场景连接池宜配置较多连接数（与并发交易量匹配），侧重短连接快速响应；OLAP场景连接数通常较少但单连接执行时间长，宜适当增大连接超时时间和单连接最大存活时长，避免长查询被连接池回收。两类负载共用集群时，宜分别配置独立连接池。

## 7.2 数据类型

证券期货业用到的分布式数据库具有数据量大与数据类型多的特点，应当在开发过程中统一数据类型的相关设计要求，使其具有较高水平的扩展性和可用性。数据类型相关设计具体要求如下：

- a) 应使用与数据实际类型相对应的数据类型，以减少不必要的数据类型转换；
- b) 宜使用DATETIME类型存储时间戳数据（不依赖时区，存储的是实际的日期和时间）；
- c) 宜使用DECIMAL或整数类型存放涉及金额的数据。如果存储的数据范围超过DECIMAL的范围，应将数据拆分为整数和小数分开存放；
- d) 对于风险因子、计量系数等需要多级精度的金融计算中间结果，应明确小数精度要求并统一声明DECIMAL(M,D)的精度参数，避免不同系统间因精度差异导致计算结果不一致；
- e) DECIMAL类型的字段，宜使用DECIMAL(M,D)的方式声明；
- f) 如果DECIMAL(M,D)类型字段的D值为0，宜改为使用BIGINT或BIGINT UNSIGNED类型；

- g) 对于字符串字段, 宜使用VARCHAR字段类型;
- h) OLTP场景下, 所有字段宜定义为NOT NULL; OLAP场景下, 允许字段为空(NULL), 但应在数据模型文档中注明空值的业务含义;
- i) 宜将大字段、访问频率低的字段拆分到单独的表中存储;
- j) 对于自增列, 宜使用BIGINT或BIGINT UNSIGNED类型。

### 7.3 主键

主键相关设计的具体要求如下:

- a) OLTP场景下, 所有新增表应设置主键。OLAP场景下, 事实表和维度表宜设置主键, 追加写入型明细表可不设主键;
- b) 使用自增列作为主键时, 如果该表存在逻辑主键, 该逻辑主键宜设置为唯一索引;
- c) 主键的命名宜使用“PK\_字段名”, 关键词之间用下划线隔开。

### 7.4 外键

分布式数据库开发过程中, 不应使用外键。

### 7.5 索引

#### 7.5.1 设计原则

索引的设计应本着高效的原则, 具体的设计要求如下:

- a) 数据表索引请按照“分区索引”、“全局分区索引”、“全局索引”优先级递减原则创建使用。分区索引的适用场景为单分片高频查询和高写入吞吐场景, 全局分区索引适用场景为跨分片范围查询、排序查询、聚合查询场景, 全局索引的适用场景为全局唯一值、不带分片键的精确查询场景;
- b) 索引会降低更新类、写入类的性能, 数据表宜按需建索引;
- c) 对于多个字段的查询, 宜优先建立复合索引;
- d) 对于查询条件涉及的字段, 宜优先在区分度高的字段上建立索引;
- e) 索引创建须遵循按需评估、兼顾读写、联合优先的原则, 严禁未经业务场景分析, 对表的所有列逐一创建单列索引;
- f) OLTP场景宜通过索引保障点查和低延迟写入, 索引数量宜精简; OLAP场景以全表扫描和聚合为主, 索引数量宜从严控制, 优先依赖列存压缩和分区裁剪, 避免过多索引影响批量写入性能。

#### 7.5.2 不适合使用索引的场景

下列场景不宜使用索引:

- a) 对于数据表记录数较小, 不宜使用索引;
- b) 对于区分度较低的列(如“性别”), 不宜使用索引;
- c) 对于扫描范围占整个表比例较大, 不宜使用索引。

#### 7.5.3 适合使用索引的场景

对于以下场景, 适合使用索引:

- a) 当SQL语句返回的行数占整个表总行数的比例较小, 通过索引检索数据比全表扫描效率高时, 宜建立索引;
- b) 在频繁进行排序(ORDER BY)或分组(GROUP BY)操作的列, 宜建立索引;

- c) 在频繁使用DISTINCT关键字进行查询的列，宜建立索引；
- d) 进行表连接时，在连接字段上面宜建立索引；
- e) 业务上具有唯一特性的字段，即使是多个字段的组合，宜建成唯一索引。

#### 7.5.4 复合索引

在多个索引字段都匹配的情况下，区分度高的字段宜靠前。

#### 7.5.5 冗余索引

冗余索引是指在数据库中创建了功能相同或相似的索引，它们可能为相同的查询提供性能优化，但会带来额外的存储和维护成本。常见的冗余类型分为前缀冗余和包含冗余。冗余索引的相关要求如下：

- a) 主键字段不宜再建索引；
- b) 不宜再针对当前索引已覆盖到的前导字段再创建索引；
- c) 不宜将主键字段添加到索引字段的末尾。

#### 7.5.6 覆盖索引查询

宜减少检索的字段，尽可能使用到覆盖索引查询，减少IO，提升性能。

#### 7.5.7 前缀索引

对于字段长度较长的字符串列，如果需要建立索引，宜建立前缀索引。

### 7.6 注释

- a) 创建表时宜为表名和每个字段添加详细注释，描述其业务含义和使用规范，增强代码文档性；
- b) 宜在创建表之后使用ALTER语句加注释，便于后续对表的维护；
- c) 如果修改字段含义或对字段表示的状态追加时，宜及时更新字段注释。

## 8 编程规范

### 8.1 DQL 操作

#### 8.1.1 基本原则

证券期货类业务逻辑层次多，原则上避免跨业务系统直接访问数据表。

#### 8.1.2 结果集

对于SELECT COUNT语句，如果某字段含有NULL值，宜使用SELECT COUNT(\*)统计记录数。

#### 8.1.3 表连接

- a) 多表连接场景下，每个字段宜显式指定表名或表别名作前缀；
- b) 多表连接场景下，宜将过滤性较大的表选为驱动表；
- c) 通常情况下，在WHERE条件中进行过滤的表是驱动表，并将该驱动表放在FROM子句的开头位置以显式指定；
- d) OLTP场景下，多表连接个数不宜超过3个；OLAP场景下，表连接个数不宜超过8个，超过时宜通过预计算宽表或物化视图降低运行时关联复杂度；
- e) 需要JOIN的字段，数据类型应绝对一致。多表关联查询时，保证被关联的字段有索引。

#### 8.1.4 表嵌套

- a) 确定合理的嵌套深度。语句的嵌套层数不宜超过3层；否则，宜使用中间表和临时表，简化语句的复杂度；
- b) 明确父子关系。宜清晰定义注释每个表之间的父子关系，确保数据层次结构合理。

#### 8.1.5 数据范围（WHERE 条件）

数据范围的相关具体要求如下：

- a) 进行数据比较时，如果两边的数据类型不一致，宜在一方加上类型转换的函数；
- b) SQL语句中IN包含的值宜少于1000个；
- c) SQL语句中如果查询条件是数值列表，宜使用IN操作；
- d) 宜通过主键或唯一索引进行数据更新/删除。

#### 8.1.6 联合

- a) 如果不需要对结果集进行去重、排序，应使用UNION ALL代替UNION，并且UNION分支尽量控制在5个以内；
- b) 使用UNION/UNION ALL时，不宜超过10张表；
- c) 确保联合查询中包含有效的连接条件，不宜使用笛卡尔积，即两个表的记录数相乘的结果集；
- d) 选择合适的连接类型。

#### 8.1.7 分组

- a) 如果涉及字段建立了索引，为有效利用索引，GROUP BY的字段顺序宜与索引顺序一致；
- b) 宜保证穷尽性原则，每个记录归属于一个分组；
- c) 宜保证互斥性原则，每个记录只能属于一个分组；
- d) 宜提前识别并处理异常值，选择剔除、修正或单独分组分析。

#### 8.1.8 排序

- a) 如果表连接后有排序操作，在满足业务的条件下，宜使用驱动表的索引字段进行排序；
- b) 选择合适的字段，例如高区分度的字段进行排序；
- c) 注意对数据库中NULL值设定规则或者提前排除。

#### 8.1.9 分页

- a) 宜使用合理的分页方式，先快速定位需要获取的记录的主键范围，然后再关联，以提高分页效率；
- b) 宜使用通用的分页方案，编写可移植的分页代码，提高代码的兼容性。

### 8.2 DML 操作

#### 8.2.1 数据写入

- a) INSERT语句宜显式指定列名，宜使用“INSERT INTO 表名(列名1, 列名2, ...) VALUES(值1, 值2, ...)”；
- b) 业务上对某个字段频繁地重复写，宜使用写缓存机制；
- c) OLAP场景允许单次批量写入大批量数据(如百万行级ETL导入)，宜采用批量INSERT或LOAD DATA方式，并适当放宽单次事务行数限制；

- d) 大批量数据导入（如全市场行情、历史风险因子）后，应对导入数据的行数、关键字段完整性及业务逻辑一致性进行校验，校验通过后方可用于下游业务计算。

### 8.2.2 数据清理

对于删除全表或某个分区（LIST、RANGE）数据的操作，宜使用TRUNCATE功能来实现。

### 8.2.3 性能优化

- a) 应定期数据清理，通过定期删除不再使用的数据、优化查询语句、合理使用索引等方式，提升数据库的响应速度；
- b) 应做好缓存优化，采用缓存预加载、设置缓存过期时间等策略，提高缓存的命中率；
- c) 应控制单次操作粒度，对于涉及大量数据的批量操作，OLTP场景必须在应用层进行分批处理，原则上不建议一次性提交大规模的事务操作，以避免长事务导致的锁表、回滚段溢出或主从同步延迟。OLAP场景下，批量导入和聚合计算允许使用大事务，但仍宜设置合理的超时时间和资源上限。

### 8.3 DDL 操作

DDL是SQL的重要组成部分，用于定义和管理数据库的结构。DDL的主要操作包括创建数据库对象、修改数据库对象、删除数据库对象和其他操作。其具体的应用要求如下：

- a) 针对表级的DDL操作，宜评估停机时间窗口；
- b) 对同一个表的多次ALTER操作宜合并为一次操作；
- c) DDL操作宜与该表的其他SQL错开执行；
- d) 如需替换索引，宜建新索引，再删除旧索引，且新旧索引名称不相同；
- e) 数据订正时，删除和修改记录时，宜先SELECT，避免出现误删除，确认无误才能执行更新语句；
- f) 宜对DDL操作实施实时监控，应预设回滚策略以应对可能的风险，确保操作的可靠性和可恢复性；
- g) 耗时较长的DDL操作的执行时机宜选择业务低峰期，减少对在线业务的影响；
- h) 并行度宜做针对性调整。根据系统资源状况和业务负载，合理调整DDL操作的并行度，如数据回填的并行度和限速，以优化执行效率。

### 8.4 SQL 调优操作

在SQL调优中，针对慢SQL的分析可参考如下步骤：

- a) 通过全局SQL审计表、SQL跟踪（TRACE）和计划缓存视图查看SQL执行性能信息，初步查找SQL请求的流程中导致耗时大或消耗资源多（如内存、磁盘IO等）的SQL；
- b) 通过执行EXPLAIN命令查看优化器针对给定SQL生成的逻辑执行计划，确定可能的调优方向。单条SQL的执行性能往往与该SQL的执行计划相关；
- c) 收集SQL中涉及到的表、列、谓词等对象的统计信息。统计信息是代价模型中选取最优执行计划的关键，优化器可以利用统计信息来优化计划的选择策略；
- d) 找到具体的慢SQL，为了使某些SQL的执行时间或资源消耗符合预期，常见的优化方式如下：
  - 1) 对SQL做等价改写生成最佳执行计划；
  - 2) 针对多表访问的SQL，还需要关注多表间的联接问题，通过优化访问路径、联接顺序和联接算法等实现查询优化。
- e) 采取SQL性能优化策略改写或是替代该慢SQL语句，标准规范如下，包括但不限于：

- 1) 宜采用批量SQL语句，减少与数据库交互次数。例如：INSERT INTO表名(X,Y,Z) VALUES (X1,Y1,Z1), (X2,Y2,Z2), (X3,Y3,Z3)...;
- 2) 能用GROUP BY替代时宜避免使用DISTINCT。数据库引擎对GROUP BY的优化有时优于DISTINCT，尤其在查询字段包含索引列时，而DISTINCT需要额外创建临时表；
- 3) 宜用UNION ALL替代OR。当查询条件中有OR且涉及索引列时，使用UNION ALL可以让数据库分别对每个条件使用索引，避免全表扫描。而OR可能导致索引失效，特别是当两个条件涉及不同列时；
- 4) 宜避免对索引列使用函数操作。函数操作会破坏索引的有序性，导致全表扫描。如果分布式数据库版本支持，可以考虑创建函数索引；
- 5) 宜对大型子查询使用EXISTS替代IN。IN对小型数据集表现良好，但当子查询涉及大型数据集或关联查询时，EXISTS的性能更优；
- 6) 宜使用覆盖索引。覆盖索引包含查询所需的所有列，数据库无需回表读取数据页；
- 7) 对于涉及大批量数据聚合计算的场景（如风险指标批量计算、历史数据回测等），宜通过预计算宽表、物化视图或批处理中间表减少运行时计算复杂度，避免单条复杂SQL长时间占用集群资源；
- 8) 宜使用LIMIT限制ORDER BY结果集；
- 9) 宜使用JOIN关联表格减少子查询次数；
- 10) 禁止使用SELECT \*，即使表本身只有一两列，且都需要，也应该明确在SELECT中写出列名；
- 11) 宜控制索引的数量。表中新增数据时，需要同时为其创建索引，而索引需要额外的存储空间并造成性能消耗；
- 12) OLAP场景下，宜优先使用窗口函数替代多层子查询和自关联，提升复杂分析查询的可读性和执行效率，宜通过视图封装高频分析逻辑，减少重复SQL编写。

## 8.5 分布式事务编程

### 8.5.1 总体原则

- a) 最小化使用原则：在证券期货业务系统中，应尽最大可能通过业务拆分、数据本地化、异步解耦等方式避免跨节点强一致性事务。仅在无法规避且业务逻辑明确要求原子性（如资金划转与持仓变更同步）时，方可启用分布式事务机制；
- b) 合规优先原则：所有涉及客户资产、交易记录、清算结果等关键数据的操作，必须满足事务一致性和可审计性的规定；
- c) 可观测性内建原则：任何分布式事务实现必须内嵌完整的上下文追踪、日志记录和状态回溯能力，确保满足监管审计和故障复盘要求。

### 8.5.2 事务处理

考虑到证券期货交易过程中的一致性，分布式数据库应具有高安全性与数据高一致性，事务处理过程中的具体要求如下：

- a) 如果涉及事务处理，应在完成事务所有的DML操作后执行提交操作（COMMIT）；在异常处理时，应包含回滚操作（ROLLBACK）；
- b) 单个事务更新、插入、删除的记录数，OLTP场景下应小于10万行；OLAP场景下按8.2.1c)条执行；
- c) 为避免死锁，联机交易之间、联机与批量之间、并行运行的批量之间，宜使用相同的次序对表进行更新；

- d) 对于只读事务，宜在SELECT结束后尽快执行提交操作；
- e) 对于联机交易，事务宜尽可能小，逻辑上一次原子操作才宜放入一个事务；
- f) 事务内不宜包含线程阻塞操作；
- g) UPDATE要注意控制修改的行数，确保事务不会太大；
- h) OLAP场景下，宜采用批量提交或隐式事务机制处理常规数据加载，仅在需要保证多语句原子性的复杂ETL环节，才显式开启并提交事务。

## 8.6 异常处理

分布式关系型数据库需要对结果进行紧急的处理，使得整个系统具有较高的完善度，设计结果处理的编程要求如下：

- a) 数据库调用等IO操作应捕获异常并处理；
- b) 在资源使用完毕后，应及时释放，避免资源耗尽的风险（包括异常场景的处理）。常见资源包括连接、语句、结果集/游标；
- c) 应用开发过程中出现错误码，需要根据实际选择的分布式数据库进行手册查询得到错误原因并针对性处理，处理结果与过程记录在日志中。

## 9 运维管理

### 9.1 总体原则

运维管理是保障分布式关系型数据库持续稳定运行的核心环节。证券期货业信息系统对数据库的可用性、一致性、安全性和可审计性要求极高，运维工作可参考遵循以下总体原则：

- a) 保障系统高可用性和业务连续性；
- b) 严格实施最小权限访问控制，强化身份认证与操作审计；
- c) 所有运维操作应通过标准化流程与授权机制执行，禁止未经审批的直接干预；
- d) 建立覆盖节点状态、事务延迟、资源利用率、复制一致性等维度的全栈监控体系；
- e) 统一运维工具、配置模板、操作手册与应急处置流程；
- f) 对于同时承载OLTP和OLAP负载的分布式数据库集群，应通过资源组或租户隔离机制，将交易类负载与分析类负载分配至不同资源组，分别设定CPU、内存、IO配额，防止分析查询挤占交易资源；
- g) 对于周期性高负载批处理任务（如风险指标批量计算、全市场历史数据回溯、多场景压力仿真等），应通过资源配额或任务调度机制限制其对联机业务的影响，并在批处理窗口期间加强核心交易接口响应时间的监控。

### 9.2 运维系统功能要求

#### 9.2.1 系统监控

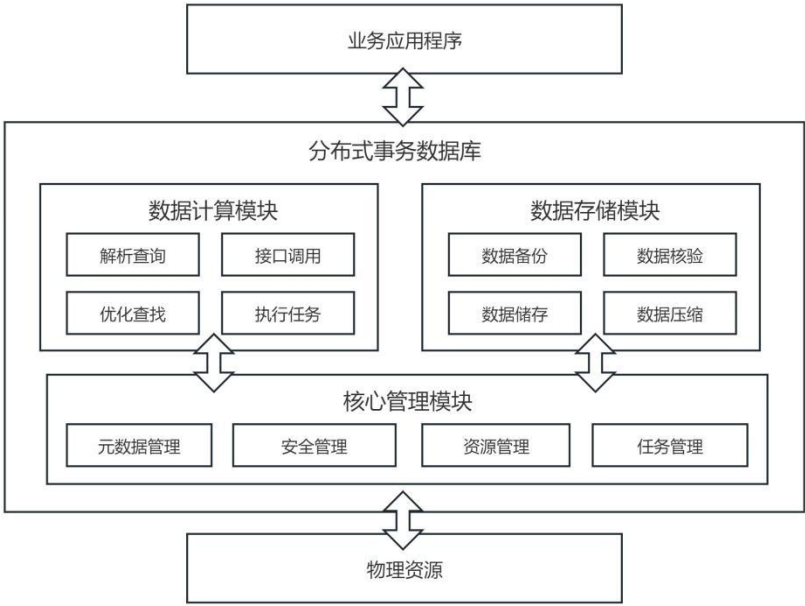


图2 分布式关系型数据库模块结构图

证券期货业涉及的运维系统中应包含一套全面、实时、智能、合规的监控系统，覆盖全部的核心指标，指标宜覆盖分布式关系型数据库的主要模块（如图2），保障分布式关系型数据库系统的高可用性、高性能、数据一致性、安全合规性，满足证券期货业务对交易连续性、数据完整性、风险可控性的严苛要求。监控指标应按业务重要性划分最小监控集，核心业务指标须纳入强制监控范围，非核心指标可按需配置。关键指标须设置多级阈值告警（预警、严重、紧急，阈值指标具体参考附录A.2），其他阈值确定宜采用历史基线分析结合业务容忍度评估的方法，并设置对应持续时间门槛，以避免抖动误报。告警信息宜自动推送至值班人员并记录处置闭环。各监控指标宜定期审查，可根据系统负载、硬件能力调整监控阈值。

监控具体指标的范围包括但不限于：

- a) 基础设施：服务器、网络设备、存储系统、容器/虚拟化平台资源使用与调度情况；
- b) 数据库实例：节点状态、连接数、锁等待、查询性能、事务处理状态、缓存命中；
- c) 分布式特性：数据分片与负载均衡、跨节点事务、数据复制延迟、集群拓扑；
- d) 数据库接口：核心交易接口数据库响应时间、数据校验。

不同级别告警应设定差异化处置时限，预警建议30分钟内响应，严重建议15分钟内响应，紧急建议5分钟内响应。

9.2.2 备份恢复

证券期货业运维系统应具有关键数据库备份与恢复机制，这是保障金融信息系统安全、稳定、连续运行的重要基础。

备份机制要求如下：

- a) 核心交易、结算、风控等关键数据宜使用全量备份、增量备份和日志备份；
- b) 核心数据的存储与灾备架构设计宜遵循业务分级、指标驱动、成本适配的原则。在满足“本地+异地”多副本存储的基础上，根据业务系统对恢复时间目标和恢复点目标的容忍度，选择匹配的灾备架构模式，可参考附录A.3；

- c) 数据库备份有效性验证频率应根据《证券期货业信息系统备份能力规范》重要性分级实施。核心数据（对应备份能力第6级）每周宜至少进行1次有效性验证，重要数据（对应备份能力第4-5级）每月宜至少进行1次有效性验证，一般数据（对应备份能力第3级及以上）每季度宜至少进行1次有效性验证；
- d) 核心业务数据应进行归档处理，全面收集信息系统的运行日志。数据存储备份的保留期限、位置应依据《中华人民共和国网络安全法》规定执行。

恢复机制参考要求如下：

- a) 宜使用实例级、库级、表级、行级（基于时间点或事务ID）的细粒度恢复；
- b) 支持时间点恢复，精度不宜低于1秒；
- c) 在主备切换或集群节点故障时，自动触发恢复流程，无需人工干预；
- d) 恢复过程中必须确保分布式事务的原子性、一致性、隔离性、持久性（ACID），采用两阶段提交或基于日志同步机制保障全局一致性；
- e) 对于风险计量等场景涉及到的批量计算任务产生的中间结果和最终结果数据，宜建立独立的备份与版本管理机制，支持按批次回溯和结果比对，确保计算结果可复现；
- f) 恢复操作应支持安全回退，若恢复失败或验证异常，可快速回滚至恢复前状态。

### 9.2.3 变更管理

运维系统宜具备数据库系统变更管理能力。变更管理系统宜确保所有对分布式关系型数据库系统的结构、配置、数据、代码、权限及关联组件的变更，均在受控、可审计、低风险的前提下实施，保障系统安全性，满足证券期货行业要求。

## 9.3 运维系统管理

运维系统管理规范包括但不限于：

- a) 所有运维活动（包括巡检、配置、扩容、备份、恢复等）应通过标准化工单流程发起、审批与执行；
- b) 运维操作应保留完整日志，包括操作人、时间、命令、输入参数及执行结果，日志保存期限不少于180天；
- c) 运维工具应通过安全评估，宜具备操作审计、命令拦截、敏感操作两次确认等能力；
- d) 所有数据库配置变更（包括参数调整、拓扑变更、版本升级）应在测试环境充分验证后，方可按变更窗口实施；
- e) 运维自动化脚本须纳入版本管理，经测试验证后方可上线，并定期复审有效性。

## 9.4 运维灾备演练

灾备演练应根据业务系统的重要性等级进行分级、分频次的实施，以确保资源投入与风险等级相匹配。演练规范如下：

- a) 核心系统专项切换演练每年至少1次，针对核心系统某个关键模块或链路进行接管演练。全面灾备演练每3年至少1次，模拟整个核心业务系统从生产中心到灾备中心的完整切换与回切；
- b) 重要业务系统每年至少1次，模拟切换演练；
- c) 一般业务系统每3年至少1次。

所有演练活动均应遵循“计划-执行-检查-处理”的闭环管理流程。其他演练内容参见附录A.4。

## 10 故障分类与检测

## 10.1 故障分类

### 10.1.1 硬件故障

硬件本身产生的故障，其中多数单节点问题目前的分布式数据库可以自行恢复，但是一些情况依然需要通过应急手段进行主动干预。常见的硬件类故障分为以下几类：

- a) 服务器硬件故障导致的宕机；
- b) 机柜/机房级电力故障；
- c) 单节点网卡故障、机柜/机房网络抖动等；
- d) 集群负载均衡设备故障。

### 10.1.2 软件错误

软件错误是指在数据库开发过程中因为编程逻辑或设计缺陷导致后续应用过程中出现故障的问题，常见的软件错误包括但不限于以下几类：

- a) 资源管理错误；
- b) 并发控制错误；
- c) 数据库软件缺陷；
- d) 事务处理错误。

### 10.1.3 性能瓶颈

当业务层的访问量由于证券交易活动等原因突然上涨，或者有新业务发布上线时，或者有重大变更之后，往往可能导致数据库访问超过前期设计容量。此时就会表现为性能下降、响应时间变长、连接超时等一系列问题。对于分布式数据库，性能容量不足通常表现为以下几种情况：

- a) SQL查询导致的集群资源占用率过高；
- b) 集群节点磁盘IO过高；
- c) 集群节点网卡负载过高；
- d) 租户请求队列积压；
- e) 租户内存写满；
- f) 线程数量满溢；
- g) 集群节点日志盘空间满；
- h) 集群节点数据盘空间满。

### 10.1.4 网络问题

由于证券期货交易涉及的分布式数据库需要频繁地进行网络数据的处理和交互，网络的稳定性将对分布式数据库的顺畅应用起到关键的作用，网络问题导致的分布式数据库故障可分为以下几类：

- a) 网络延迟；
- b) 网络中断；
- c) 网络拥塞；
- d) 分区故障。

### 10.1.5 人为因素

人为因素是导致故障的一个常见因素，可分为以下几类：

- a) 配置错误：配置错误是指在软件、硬件或网络系统中，由于参数设置不正确、缺失或与实际情况不符合，导致系统无法正常运行或实现预期功能的技术问题。配置错误多数源于人为失误，具体可分为：
  - 1) 参数配置不当；
  - 2) 配置文件错误。
- b) 操作错误：操作错误一般是个体在执行具体操作过程中因疏忽、判断失误或技术缺陷导致的无意识错误行为，具体可分为：
  - 1) 版本安装不兼容；
  - 2) 执行了错误的命令；
  - 3) 误删除或修改数据；
  - 4) 错误授权。
- c) 恶意攻击：恶意攻击在分布式数据库的运行过程中会造成严重的数据库安全性和稳定性危机，具体可分为以下几类：
  - 1) 数据篡改与破坏类攻击；
  - 2) 拒绝服务类攻击；
  - 3) 数据窃取类攻击；
  - 4) 物理攻击。

#### 10.1.6 分布式故障

分布式关系型数据库在应用过程中存在分布式体系带来的问题，典型的分布式故障分为以下几类：

- a) 时钟偏移；
- b) 主从角色异常；
- c) 分布式事务悬挂；
- d) 复制延迟持续扩大；
- e) 元数据服务异常；
- f) 配置漂移。

#### 10.2 故障分级

为规范分布式数据库故障的应急响应流程，合理调度运维资源，根据故障对证券期货业务的影响程度、波及范围及恢复紧迫性，将故障划分为五个等级（P0-P4）：

a) P0级（特别重大故障）：核心交易业务中断、核心分析/监管报表数据产出中断、数据丢失或严重不一致，导致全市场或主要交易参与人无法正常交易。需立即响应（7×24小时），要求秒级发现，分钟级恢复；

b) P1级（重大故障）：核心业务性能严重下降（如响应时间超过阈值5倍以上）、非核心业务中断，或关键组件冗余失效（单点运行）。需紧急响应，要求分钟级发现并介入；

c) P2级（较大故障）：局部功能异常、非关键业务性能抖动，或存在潜在风险（如磁盘使用率超过80%），暂不影响核心交易连续性。需在工作时间内快速响应；

d) P3级（一般故障）：一般性错误日志、非核心配置项异常、备份任务失败等，不影响业务运行。按常规运维流程处理；

e) P4级（轻微故障/隐患）：系统存在的潜在隐患、性能趋势预警或合规性检查不达标。纳入定期巡检整改计划。

对于周期性批处理任务（如定期风险计量等场景），若任务未能在规定业务窗口内完成，可根据其对下游业务的影响程度参照上述等级进行分级。

10.3 故障检测

10.3.1 检测系统

在故障处理的过程中，维护人员很大程度上依赖检测系统进行故障巡检，越早越全面的故障检测将对后面的故障处理起到积极的作用。检测系统应当做到准确、全面，快速地检测与预警可能出现的故障与隐患。因此检测系统结构宜从上到下进行设计达到所需的检测纵深。负责检测的系统设备通常与运维系统相结合，宜包含运维控制层、智能分析层、数据采集层和基础设施层，运维控制层负责统一的调度，分析层进行指标分析，排查隐患，数据采集层将详细数据进行备份并实时监控，具体结构要求与功能如图3：

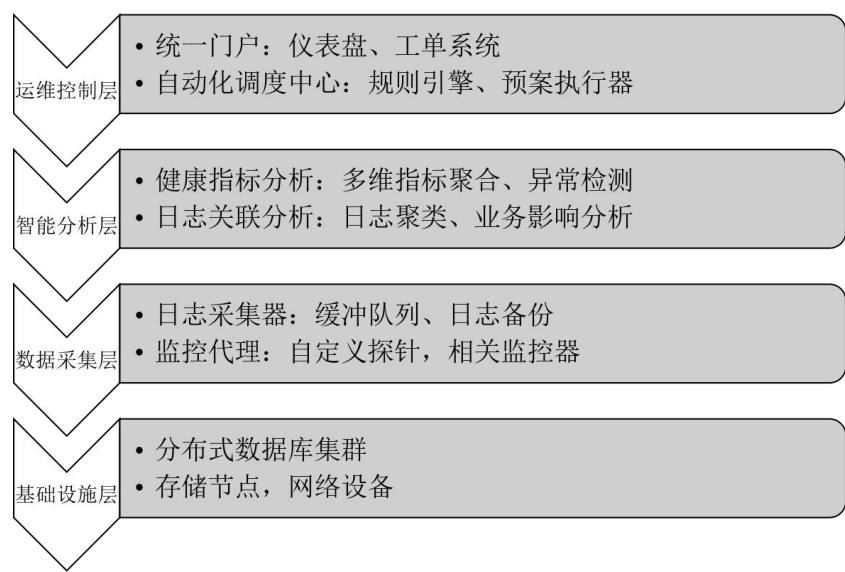


图 3 故障检测系统层级结构图

整个系统的参考设计要求如下：

- a) 宜引入自动化、智能化的性能监控工具；
- b) 宜增加识别运行态的会话数、锁等待、高频SQL和慢SQL等多维度指标，及时发现资源不足、性能瓶颈的情况，并通过可视化运维面板展示系统各个节点的健康状况和性能状态，及时发现和处理问题；
- c) 宜建设统一的分布式日志系统，统一收集每个节点的操作日志和通信日志，利用集中化日志分析工具，快速定位和排查故障；
- d) 应建立详细的故障处理预案，包括节点故障时的自动切换、数据恢复策略、备用节点激活流程等，确保数据库集群在发生故障时能够迅速恢复服务；
- e) 应选择业界广泛验证，有成功应用案例的分布式数据库版本，灾难恢复时允许的最大数据丢失时间（RPO）和业务系统允许的最长停机恢复时间（RTO）应满足高可用性需求。

10.3.2 检测渠道

故障的发现与检测，通常分为主动发现和被动感知两类，一般分为以下几个渠道：

- a) 业务监控报警（应用层监控）；
- b) 数据库监控报警；
- c) 数据库日常巡检。

检测系统应至少具备上述3种故障检测能力。

检测系统需要做好告警管理工作，相关规范如下：

- a) 宜建立告警抑制机制，当多个告警由同一原因触发时，仅保留根因告警，抑制衍生告警；
- b) 同一指标在5分钟内重复超阈值，仅触发首次告警，避免告警风暴；
- c) 告警超过处置时限未处理，自动升级通知更高层级责任人；
- d) 同一指标持续恶化，自动提升告警级别并通知；
- e) 告警记录应保存不少于180天，支持合规审计要求。

### 10.3.3 检测频率

在证券期货行业交易中涉及的分布式数据库，宜建立故障等级与检测频率的联动机制。相应的检测频率参考如下：

- a) 关键指标（P0/P1级）：直接影响核心交易、清算、风控等关键业务连续性的指标，如主节点存活状态、事务提交延迟、数据同步中断、连接池耗尽等，指标基线参考附录A.5。一旦异常，可能在数秒至数分钟内引发重大故障，这类指标的检测频率不低于每秒1次；
- b) 常规指标（P2级）：影响系统性能或局部功能，但尚不导致业务中断的指标，如慢查询数量、CPU/内存使用率、副本延迟、锁等待等。异常时可能演变为较大或一般故障。这类指标的检测频率不低于每分钟1次；
- c) 深度检查指标（P3/P4级）：用于系统健康度评估、容量规划或合规审计的非实时指标，如历史性能趋势、日志归档完整性、备份有效性、配置合规性等。异常通常不立即影响业务，但长期忽视可能导致风险累积。根据业务需求，每天或者每周至少进行1次。

## 11 故障处理

### 11.1 概述

分布式关系型数据库故障的处理流程顺序为事件感知、异常信息收集、故障原因分析、决策处理、故障复盘与评估几个步骤。

### 11.2 事件感知

故障应急事件的感知可参考章节10。

无论以哪种方式发现，事件/故障应急的第一步都应该是故障信息同步（包括当前影响以及接手处理人），以及异常报错信息的快速收集，为下一步的分析和应急处理提供依据。

### 11.3 异常信息收集

异常信息宜按照以下要求的维度收集并整理：

- a) 异常表现：一般第一层是业务相关的信息，如用户无法登录、交易超时、批量超时等。异常信息收集的第一步是需要从业务告警、监控信息中定位寻找到数据库相关的告警信息。例如从应用的日志、中间件、客户端等报出的数据库相关异常信息等；
- b) 异常时间点：确认故障/异常开始的具体时间点，以及持续时间；
- c) 异常范围：确认异常发生在单个应用还是全部应用，对应的影响范围是全局，还是某个集群或租户。对于全局范围的影响，往往先从基础设施异常入手排查。对于局部的故障异常，需要确认具体的集群、租户、数据库、代理详细信息；
- d) 报错相关日志：收集异常触发期间报错日志信息，一般来自应用日志以及客户端日志；

- e) 是否有变更/发布：确认故障时间点前后业务是否有发布或变更，如果有需要收集具体信息。在实际场景中，大量的故障和异常往往都是由变更导致的；
- f) 对关联业务的影响：确认当前故障的影响面，如果持续，是否会影响上下游其他系统。

## 11.4 故障原因分析

### 11.4.1 数据库自身的巡检和告警的故障

一般指向为具体的数据库事件。在开始排查任何应急事件之前，将最基础的异常情况尽早排除，避免无效排查。影响分布式数据库正常工作的最基础的软硬件因素主要包括但不限于：

- a) 网络时间协议（NTP）时钟是否同步；
- b) 是否服务器宕机；
- c) 是否有日志磁盘、数据磁盘空间满；
- d) 机房网络是否抖动；
- e) 负载均衡设备/组件是否故障；
- f) 数据库日志是否存在严重错误；
- g) 数据库活动会话是否存在锁等待；
- h) 数据库活动会话是否存在长时间运行的SQL；
- i) 数据库的每秒事务处理量（TPS）、平均响应时间（ART）是否存在抖动；
- j) 分布式集群节点是否失联或心跳中断；
- k) 副本是否存在多数不可用或日志同步滞后；
- l) 分区是否存在迁移失败；
- m) 全局时间戳是否存在时钟漂移；
- n) 分布式事务管理器是否存在阻塞或超时。

这些因素会在数据库自身巡检中暴露，无需进一步排查，而应用发现的故障需要进一步规范化地排查找到应用层故障的根本原因。

### 11.4.2 应用发现的故障

一般来说从应用层暴露出的数据库相关错误和异常，通常分为以下几大类，包括但不限于：

- a) 应用连接池满；
- b) 应用队列积压；
- c) 应用请求超时；
- d) 应用建连失败；
- e) 应用写入失败；
- f) 应用锁冲突。

各类故障排查与处理规范如下：

- a) 应用连接池满/队列积压/请求超时故障，检测信号为应用连接池使用率持续接近满负载，或特定分区QPS、RT异常升高。排查及处理建议如下：
  - 1) 排查慢SQL：检查是否存在执行计划异常导致响应变慢、引发重试的SQL。紧急状态下实施请求限流或熔断，后续优化执行计划；
  - 2) 排查节点资源：监控数据库节点CPU和内存水位，确认是否因资源瓶颈导致任务排队或写入阻塞。必要时进行资源扩容；

- 3) 排查分布式状态：检查是否存在分区负载不均、Leader频繁切换或跨节点事务协调延迟。紧急状态下限流异常SQL，后续建立弹性扩缩容机制。
- b) 应用建连失败故障，检测信号为TCP连接超时或拒绝，代理线程池耗尽等，排查建议如下：
  - 1) 代理建连失败排查：检查网络链路通畅性；排查负载均衡设备及组件状态；分析代理服务线程资源是否耗尽；确认代理路由表是否更新及目标分区是否无主；
  - 2) 服务器建连失败排查：确认网络链路正常；检查数据库服务器CPU/IO使用率及异常SQL占用情况；确认目标存储节点在线且具备可用Leader副本。
- c) 应用写入失败故障，检测信号为租户内存使用率过高，日志盘剩余空间过低，长事务数量突增等，排查建议如下：
  - 1) 排查内存与事务：确认是否因批量写入、长事务/悬挂事务阻碍转储，或其他模块占用过多导致内存耗尽。紧急状态下终止长事务，后续调整内存配额、优化提交频率或实施写入限流；
  - 2) 排查存储与共识：检查日志盘空间是否不足；确认多副本共识协议是否达成多数派确认（排除网络分区或副本宕机）；
  - 3) 排查时钟服务：检查全局时间戳服务是否异常或时钟漂移是否超出容限。
- d) 应用锁冲突故障，检测信号为LOCK\_WAIT状态会话占比高，平均锁等待时间长。排查建议如下：
  - 1) 排查SQL性能：确认是否因SQL响应慢导致锁持有时间延长。若是，需解决SQL性能问题；
  - 2) 排查热点竞争：确认是否因高并发修改同一行数据引发锁等待堆积；
  - 3) 排查隐式冲突：检查是否因副本延迟读取旧版本数据，或事务重试机制不当放大锁竞争。

### 11.5 决策处理

通过上述故障诊断过程，剩下的应对就是实施对应的决策处理动作，决策处理的原则包括：

- a) 快速恢复，减少业务被影响的时长；
- b) 保证数据不丢失；
- c) 保证数据一致性；
- d) 最小化影响范围：尽可能限制故障对整个系统的影响。

具体的操作要求如下：

- a) 明确记录并复核下发的每个应急变更操作，作为如果恢复失败后继续深入诊断的参考信息；
- b) 应尽可能记录备份现场信息，包括异常期间的观测器日志、核心文件等；
- c) 相关的应急恢复操作，不宜破坏集群数据，保证数据一致性为前提；
- d) 当应急动作执行完毕，业务已经恢复后，应将一些参数配置的变更进行回滚，并对问题/故障进行复盘；
- e) 基于应急期间保留的现场信息，后续应展开问题的根因排查工作；
- f) 应急变更操作应做好回撤预案，以便操作失误后及时恢复。

### 11.6 复盘与评估

- a) 故障信息收集
  - 1) 时间记录：准确记录故障发生的时间、持续时间；
  - 2) 日志收集：收集数据库日志、系统日志、网络日志等相关的日志信息；
  - 3) 现场信息：记录故障现场的状态，包括错误信息、性能指标等；
  - 4) 用户反馈：收集故障造成影响的用户反馈，了解故障的影响程度。
- b) 故障影响评估

- 1) 业务影响：评估故障对业务的影响范围、持续时间和整体损失；
  - 2) 系统影响：评估故障对系统性能、可用性、数据完整性等方面的影响；
  - 3) 用户影响：评估故障对用户体验、满意度方面的影响。
- c) 故障处理回顾
- 1) 处理步骤：详细记录故障处理过程中的每一步操作；
  - 2) 处理效率：评估故障处理的响应时间、解决时间及效率；
  - 3) 处理效果：评估故障处理的效果，是否彻底解决问题，是否有遗留问题。
- d) 改进措施制定
- 1) 技术改进：包括自动化恢复机制、系统优化、代码改进、监控增强等方面提升故障处理能力；
  - 2) 人员培训：针对运维或其他相关人员进行技术培训，提高其技术水平和故障处理能力；
  - 3) 文档更新：进一步完善各团队内部的故障处理预案与操作手册，尽量避免同一类故障再次造成影响。

附录 A  
(规范性)

A.1 数据表限制

A.1.1 单表限制

表 A.1 单表配置限制表

| 配置项   | 最大限制    |
|-------|---------|
| 行长度   | 1.5 兆字节 |
| 列数    | 4096 列  |
| 索引个数  | 128 个   |
| 索引总列数 | 512 列   |
| 索引长度  | 16 千字节  |
| 主键总列数 | 64 列    |
| 主键长度  | 16 千字节  |
| 分区个数  | 8192 个  |

A.1.2 标识符长度限制

表 A.2 标识符长度限制表

| 配置项  | 最大限制   |
|------|--------|
| 集群名  | 128 字节 |
| 用户名  | 64 字节  |
| 租户名  | 63 字节  |
| 数据库名 | 128 字节 |
| 表名   | 64 字节  |
| 列名   | 128 字节 |
| 索引名  | 64 字节  |
| 视图名  | 64 字节  |
| 别名   | 255 字节 |

A.2 指标阈值

表 A.3 关键指标阈值限制表

| 指标名称    | 预警阈值 | 严重阈值 | 紧急阈值 |
|---------|------|------|------|
| CPU 使用率 | >70% | >85% | >95% |
| 内存使用率   | >75% | >85% | >95% |
| 磁盘使用率   | >70% | >85% | >90% |

|            |             |             |             |
|------------|-------------|-------------|-------------|
| 数据库活跃连接数   | >最大连接数的 60% | >最大连接数的 80% | >最大连接数的 90% |
| 平均响应时间     | >500 毫秒     | >1000 毫秒    | >3000 毫秒    |
| 慢查询数量      | >10 个       | >50 个       | >100 个      |
| QPS/TPS 跌幅 | 比正常值下跌 20%  | 比正常值下跌 40%  | 比正常值下跌 60%  |
| 主从复制延迟     | >5 秒        | >30 秒       | >60 秒       |
| 分布式事务回滚率   | >0.1%       | >0.5%       | >1%         |

注：上表中平均响应时间、慢查询数量、分布式事务回滚率三项仅适用于OLTP场景。OLAP场景的阈值由机构根据业务窗口自行设定。

### A.3 灾备架构部署参考规范

表 A.4 部署策略参考表

| 业务等级 | 建议架构       | RTO    | RPO    |
|------|------------|--------|--------|
| 核心级  | 同城双活、两地三中心 | <5 分钟  | =0     |
| 重要级  | 同城主备       | <30 分钟 | <5 分钟  |
| 一般级  | 异地备份       | <4 小时  | <24 小时 |

### A.4 运维演练技术规范

表 A.5 演练频率参考表

| 系统级别 | 主备演练        | 时间点恢复演练   | 全链路回退验证       |
|------|-------------|-----------|---------------|
| 核心系统 | 每年至少 1 次    | 每半年至少 1 次 | 每次时间点恢复演练必须包含 |
| 重要系统 | 每年至少 1 次    | 每年至少 1 次  | 建议包含，可简化流程    |
| 一般系统 | 每 3 年至少 1 次 | 按需开展      | 可选            |

### A.5 核心业务场景强制性指标基线

表 A.6 指标基线参考表

| 业务场景 | 指标类别   | 强制性指标基线           |
|------|--------|-------------------|
| 核心交易 | 交易接口性能 | 99%的情况下响应时间<50 毫秒 |
|      | 事务处理   | 提交成功率>99.99%      |
|      | 数据同步   | 主从同步延迟<1 秒        |
| 清算业务 | 批量处理   | 批量完成时间<业务窗口的 80%  |
|      | 数据一致性  | 清算数据一致性 100%      |
|      | 资金安全   | 资金流水完整性校验通过率 100% |
| 风控业务 | 风险计量   | 风险指标计算延迟<业务窗口要求   |
|      | 实时监控   | 监控数据丢失率=0         |
|      | 模型更新   | 风控模型加载成功率=100%    |

|      |        |                         |
|------|--------|-------------------------|
| 数据分析 | 批量查询吞吐 | 核心报表查询完成时间<业务窗口要求       |
|      | 资源隔离   | 分析负载导致交易类 99%响应时间劣化<20% |

## 附 录 B

### （资料性）

### 典型 SQL 调优示例

#### B.1 低效SQL请求优化

##### B.1.1 避免全表扫描与不必要的列查询

使用 `SELECT *` 会查询出不需要的列往往比较低效，其浪费 IO、内存和网络资源，且可能导致覆盖索引失效。例如：

```
SELECT * FROM trade_order WHERE user_id = 'U123456';
```

优化写法为仅查询业务所需的特定字段。例如：

```
SELECT order_id, symbol, trade_qty, trade_price
FROM trade_order
WHERE user_id = 'U123456';
```

##### B.1.2 深分页优化

在数据量巨大时，传统的 `LIMIT M,N` 会导致大量的数据被扫描和丢弃，造成性能急剧下降。随着 `offset` 的增大，查询速度会越来越慢。例如：

```
SELECT order_id, trade_time
FROM trade_order
ORDER BY trade_time DESC
LIMIT 100000, 10;
```

可以采用延迟关联的方式进行优化，即先通过子查询利用索引获取主键，再关联原表获取数据，大幅减少回表的数据量。例如：

```
SELECT t1.order_id, t1.trade_time
FROM trade_order t1
INNER JOIN (
SELECT id FROM trade_order ORDER BY trade_time DESC LIMIT 100000, 10
) t2 ON t1.id = t2.id;
```

对于有自增主键或有序字段的场景，也可通过记录上一次查询的最大 ID 进行过滤。例如：

```
SELECT order_id, trade_time
```

```
FROM trade_order

WHERE id > 1000000 -- 上一页最后一条记录的 ID

ORDER BY id LIMIT 10;
```

## B.2 索引使用与失效场景

### B.2.1 避免对索引列进行运算或使用函数

对索引列进行计算或函数操作会导致索引失效，引发全表扫描。低效写法总是在索引列 `create_time` 上使用 `YEAR()` 函数，导致索引失效。例如：

```
SELECT * FROM trade_order WHERE YEAR(create_time) = 2023;
```

优化写法为将计算逻辑转移到常量侧，利用索引进行范围扫描。例如：

```
SELECT * FROM trade_order

WHERE create_time >= '2023-01-01 00:00:00'

AND create_time < '2024-01-01 00:00:00';
```

### B.2.2 正确处理模糊查询

模糊查询中通配符的位置直接影响索引的使用效率。低效写法常常是左模糊或全模糊匹配无法使用 B-Tree 索引的前缀匹配特性。例如：

```
SELECT * FROM trade_order WHERE symbol LIKE '%AAPL';
```

优化写法为尽量使用右模糊匹配，以有效利用索引。例如：

```
SELECT * FROM trade_order WHERE symbol LIKE 'AAPL%';
```

### B.2.3 遵循联合索引最左前缀原则

联合索引的使用必须遵循最左前缀原则，否则索引将失效。假设存在联合索引 `idx_user_status_time (user_id, status, trade_time)`。低效写法为跳过索引最左列 `user_id`，直接查询 `status`，这会导致索引失效，例如：

```
SELECT *

FROM trade_order

WHERE status = 'FILLED';
```

优化写法为从最左列开始查询，或包含最左列的组合查询。例如：

```
SELECT *

FROM trade_order

WHERE user_id = 'U123456'
```

```
AND status = 'FILLED' ;
```

### B.3 JOIN优化

#### B.3.1 小结果集驱动大结果集

在 JOIN 操作中,应尽量用过滤后数据量小的结果集作为驱动表,以减少外层循环的次数。如果 user 表数据量远大于 order 表,以 user 为驱动表效率较低。例如:

```
SELECT u.user_name, o.order_amount
FROM user u
INNER JOIN order o ON u.user_id = o.user_id
WHERE u.status = 'ACTIVE' ; -- 假设活跃用户很多
```

优化写法为如果特定时间段的订单量远小于活跃用户数,以 order 为驱动表更优。

```
SELECT u.user_name, o.order_amount
FROM order o
INNER JOIN user u ON o.user_id = u.user_id
WHERE o.trade_time BETWEEN '2026-01-01' AND '2026-01-02' ;
```

#### B.3.2 为关联条件增加索引

确保 JOIN 条件的字段上建立了索引,是高效关联查询的基础。优化建议为在被驱动表的关联字段上建立索引。例如当利用 user\_id 进行关联的时候在 order 表的 user\_id 字段上建立索引:

```
CREATE INDEX idx_order_user_id ON order(user_id);
```

## 参 考 文 献

- [1] GA/T 1726—2020 负载均衡产品安全技术要求
  - [2] JR/T 0203—2020 分布式数据库技术金融应用规范 技术架构
  - [3] JR/T 0204—2020 分布式数据库技术金融应用规范 安全技术要求
  - [4] JR/T 0250—2022 证券期货业数据安全管理与保护指南
  - [5] JR/T 0295—2023 证券期货业信息安全运营管理指南
-