服务器出问题，目前部分恢复

35 结语：ShardingSphere 总结及展望

终于到了专栏的最后一讲。今天，我们将对整个 ShardingSphere 课程进行总结和展望。作为一款在业界领先的分布式数据库中间件，ShardingSphere 受到越来越多人的追捧，它可以为我们提供多项核心功能，并帮忙我们构建完整的分库分表解决方案。

首先，我们还是总结一下专栏中讲解过的 ShardingSphere 核心功能，然后再梳理我在写作过程中的一些思考和心得，最后，我会向你讲解 ShardingSphere 4.X 版本至未来 5.X 版本的演进变化。

ShardingSphere 核心功能

ShardingSphere 官网展示了数据分片、分布式事务、数据库治理等三大块核心功能，对于这些功能，我分别在本专栏的第四部分、第五部分、第六部分都进行了详细介绍，你可回顾重温一遍。

1.数据分片

数据分片是 ShardingSphere 的基本功能。ShardingSphere 支持常规的基于垂直拆分和水平拆分的**分库分表**操作。在分库分表的基础上，ShardingSphere 也实现了基于数据库主从架构的**读写分离机制**，而且这种读写分离机制可以和数据分片完美地进行整合。

另一方面，作为一款具有高度可扩展性的开源框架，ShardingSphere 也预留了分片扩展点，开发人员可以基于需要实现分片策略的定制化开发。

2.分布式事务

分布式事务用于**确保分布式环境下的数据一致性**，这也是 ShardingSphere 区别于普通分库分表框架的关键功能，并且该功能使得分布式事务能够称为一种分布式数据库中间件。

ShardingSphere 对分布式事务的支持首先体现在**抽象层面**上。ShardingSphere 抽象了一组标准化的事务处理接口，并通过分片事务管理器 ShardingTransactionManager 进行统一管理。同样，在**扩展性**上，我们也可以根据需要实现自己的 ShardingTransactionManager 从而对分布式事务进行扩展。在事务类型上，ShardingSphere 也同时支持强一致性事务和柔性事务。

当具备数据分片和分布式事务功能之后，相当于就可以基于 ShardingSphere 实现日常的分库分表操作了。但这还不够，因为我们需要对系统中的数据库资源，以及服务的运行时状态

进行跟踪和监控。因此，ShardingSphere 中也提供了多种有助于我们进行数据库治理的技术体系。

3.数据库治理

如果你一直在学习我们的专栏，相信你已经知道使用 ShardingSphere 的主要手段就是利用它的配置体系。关于**配置信息的管理**，我们可以基于配置文件完成配置信息的维护，这在 ShardingSphere 中都得到了支持。

更进一步，在ShardingSphere 中，它还提供了配置信息动态化管理机制，即可支持数据源、表与分片及读写分离策略的动态切换。而对于系统中当前正在运行的数据库实例，我们也需要进行动态的管理。在具体应用场景上，我们可以基于注册中心完成数据库实例管理、数据库熔断禁用等治理功能。

一旦 ShardingSphere 被应用到生产环境，开发和运维人员都需要关注通过 ShardingSphere 所执行的 SQL 语句的执行情况，以及 ShardingSphere 内核的运行时状态。在 ShardingSphere 中，使用 OpenTracing API 发送性能追踪数据。而在 SQL 解析与 SQL 执行等核心环节，ShardingSphere 都会把采集到的运行时数据通过标准协议提交到链路跟踪系统供我们进行分析和监控。

关于数据库治理的最后一项核心功能是**数据脱敏**。严格意义上讲，与其说数据脱敏是一项数据库治理功能，不如说它更多的是一项面向业务场景的特定功能。数据脱敏是业务系统中确保数据访问安全的常见需求，我们需要实现对原文数据进行加密并存储在数据库中。而在用户查询数据时，它又从数据库中取出密文数据并解密，最终将解密后的原始数据返回给用户。

ShardingSphere 对这一数据脱敏过程实现了自动化和透明化，开发人员无须关注数据脱敏的实现细节。

ShardingSphere 课程总结

总结完介绍的 ShardingSphere 各项核心功能，我们再来总结整个专栏所讲解内容的特色和与其他专栏之间的差异化。这里，我分为以下三大亮点。

本专栏的一大亮点在于**提供了完整的案例代码**来介绍 ShardingSphere 中的上述功能。

这个案例系统足够简单，可以让你从零开始就能理解和掌握其中的各项知识点；同时这个案例系统又足够完整，涉及的各个核心功能我们都提供了相关的配置项和示例代码，供大家在日常开发过程中进行参考。

本专栏的**最核心内容是 ShardingSphere 的源码解析**，这部分内容占据了整个专栏 $\frac{2}{3}$ 的篇幅，可以说是课程的精髓所在。

一方面，我们给出了微内核架构，以及分布式主键的设计理念和实现方法，更重要的是，我们对 ShardingSphere 中介绍的各项核心功能都从源码出发给出了详细的设计思想和实现机制。

另一方面，针对数据分片，我们剖析了其中所涉及的解析引擎、路由引擎、改写引擎、执行引擎、归并引擎和读写分离。而对于分布式事务和数据库治理，我们也结合应用场景分析了各个技术组件的底层原理，确保你能够不仅知其然，更能知其所以然。

本专栏在设计上也还有一个亮点，就是源码解析各个课时中的 **“从源码解析到日常开发”** 这部分内容。

本课程的一大目标，是通过系统化地讲解框架源码，帮忙你深入理解 ShardingSphere 实现原理，但这并不是唯一目标，我更希望你能从中收获实践技能，做到学有所用。

所以我在每一课时的“从源码解析到日常开发”部分，都会根据该课时内容梳理若干条工程实践。这些工程实践，有些是设计思想的提炼，有些是工具框架的应用技巧，还有一些则是可以直接应用到业务开发过程中的模板代码。

我希望你能通过对 ShardingSphere 这款优秀开源框架的学习，能够掌握好系统架构设计和实现过程中的方法和技巧，并把这些工程实践应用到日常的开发工作中。

从 ShardingSphere 4.X 到 5.X

最后，我们来对 ShardingSphere 的发展做一些展望。

本课程应用的是 ShardingSphere 4.X，而在当下，ShardingSphere 的开发团队正在紧锣密鼓地开发 5.X 版本。5.X 版本是 ShardingSphere 发展过程中的一个大版本，所涉及的内部功能与其对外 API 也将面临大规模的优化和调整。

同时，5.X 版本也添加了多项新的核心功能，让 ShardingSphere 生态圈更加丰富。到目前为止，5.X 版本已经设计和实现了包括弹性伸缩和影子库压测在内的多项核心功能，让我们一起分别看一下这两个功能。

1.弹性伸缩功能

5.X 版本首先要介绍的是它的弹性伸缩功能，对应的模块名称为 ShardingSphere-Scaling。随着业务规模的快速变化，我们可能需要对现有的分片集群进行弹性扩容或缩容。这个过程看似简单，实现起来却非常复杂。

为此，ShardingSphere 给出了一站式的通用型解决方案。这个方案**支持各类用户自定义的分片策略，并能减少用户在数据伸缩及迁移时的重复工作及业务影响**。弹性伸缩功能实际上在 4.1.0 版本时便已经开始引进导入，但它一直处于 alpha 开发阶段，所提供的也只是基础伸缩功能。在后续的规划中，ShardingSphere 计划通过半自动伸缩、断点续传和全自动伸缩等多个里程碑阶段来完成整个功能体系。

2.影子库压测

5.X 版本引入的第二个功能是影子库压测，这个功能的背景来自如何对系统进行全链路压测。在数据库层面，为了保证生产数据的可靠性与完整性，**需要做好数据隔离，将压测的数据请求打入影子库，以防压测数据写入生产数据库，从而对真实数据造成污染。**

在 ShardingSphere 中，我们可以通过数据路由功能将压测所需要执行的 SQL 路由到与之对应的数据源中。与数据脱敏一样，ShardingSphere 实现影子库压测的开发方式也是配置一个影子规则。

此外，ShardingSphere 还在规划和实施强一致多副本等功能，让我们一起期待这些功能早日发布。

作为业内关于 ShardingSphere 的第一门系统化专栏，《ShardingSphere 核心原理精讲》凝练着我基于 ShardingSphere 进行数据分库分表和治理工作的多年实践经验，整个专栏从酝酿到启动，再到上线也历经了小半年的时间，伴随着这个过程，我把 ShardingSphere 的源代码系统地梳理了一遍，并对内部的设计思想和实现原理也做了详细的提炼和总结。

总体而言，ShardingSphere 是一款代码质量非常高的开源框架，尤其是其中关于对 JDBC 规范的兼容、分片引擎的阶段化执行过程，以及各种辅助性的服务编排和治理等诸多功能，都让我的工作受益良久。

相信这些宝贵的“知识财富”也能一直伴随你，让你的职业生涯越走越远，越走越广。最后，祝大家在各自的岗位上都能够更上一层楼！

上一页