mysql 一个表中存多少个字段儿合理_20个MySQL高性能架构设计原则（收藏版）

原创 weixin_39697660 于 2020-12-23 17:11:03 发布 2117 收藏 5 版权

文章标签： mysql 一个表中存多少个字段儿合理

原文链接：https://www.modb.pro/db/25251 (复制链接至浏览器，即可查看)

常常会问道，怎样使用MySQL能达到高性能。以下内容是结合其他技术同仁的总结和自我实践整理的20个开源数据库设计原则。

一、开源 数据库架构 设计原则1.1. 技术选型选择成熟的平台和技术，同时是最熟悉的，能做到极致的，用好不用坏，用熟不用生。目前业界的MySQL主流分支版本有Oracle官方版本的MySQL、Percona Server、MariaDB。

1.2. 高可用选择高可用解决方案的探讨本质上探讨的是低宕机时间解决方案，可以理解为高可用的反面是不可用，绝大部分情况下数据库宕机才会导致数据库的不可用。随着技术发展，开源数据库方面很多高可用组件(主从复制，半同步，MGR，MHA，Galera Cluster),对应场景，只有适合的，没有万能的，需要理解每个高可用优缺点。

1.3. 表设计表设计方面目前为止一直坚持和提倡原则：

- 单表数据量
  所有表都需要添加注释，单表数据量建议控制在 3000 万以内
- 不保存大字段数据
  不在数据库中存储图片、文件等大数据
- 表使用规范
  拆分大字段和访问频率低的字段，分离冷热数据
  单表字段数控制在 20 个以内
- 索引规范
  1.单张表中索引数量不超过 5 个
  2.单个索引中的字段数不超过 5 个
  3.INNODB 主键推荐使用自增列，主键不应该被修改，字符串不应该做主键，
  如果不指定主键，INNODB 会使用唯一且非空值索引代替
  4.如果是复合索引，区分最大的字段放在索引前面
  5. 避免冗余或重复索引：合理创建联合索引(避免冗余)
  6. 不在低基数列上建立索引，例如‘性别’
  7. 不在索引列进行数学运算和函数运算
- 字符集utf8mb4(偏生字，表情符)

1.4. 优化原则

1.5. 复制方式MySQL复制方式提供异步方式，半同步方式，全局事务强一致性，binglog同步。需要不同业务系统间 或 两个数据库间进行同步。异步方式可以防止故障和效率问题的蔓延，扩大化，但强一致性会更复杂，并发，事务大小都有求限制。

1.6. 分离原则区分核心的业务，重要业务，渠道，内部业务的业务系统，对不同的系统设置不同的架构。为核心业务设置 最佳为分库，多活 专用高速公路，其他业务可以做读写分离，缓存

1.7. 扩展性对于系统来说扩展性很重要，尽量做到水平扩展。避免过度依赖纵向扩展，同时具备纵向，横向扩展的能力，例如无状态应用应该多套负载均衡多活部署，数据库分库架构。

1.8. 读写分离

- 读多写少场景(10%写 90%读)
- 复制存在延迟，业务对延迟不敏感的
- 实现方式：通过应用代码配置读写分离，
  通过中间代理方式路由只读库 &
  业务和数据库为一个单位

1.9. 分库分表

- 当表中数据记录的数量超过3000万条，再好的索引也已经不能提高数据查询的速度，这时需要将表拆分成更多的小表，增加性能，增加弹性，避免发生垮库进行操作。
- 引入中间价要考虑性能代价,聚合需求。
- 分库原则尽量在app 上层进行分库，就是流量
- 分多少合适：可用性和性能满足TPS
- 路由：写入配置文件 或则 插表 或则 zookeeper

1.10. 归档原则历史数据定期进行归档 或则 移到其他大数据平台。能让轻量级数据库更多缓存有用的数据。在MySQL分区表里 注意要避免分区锁，只能写读的场景。

1.11. 连接池的要求长链接，自动重链，延时和异常记录， 弹性链接，检测满，异常告警，进阶要求是记录所有访问情况，可以扩展出很多能力。应用和数据库连接池设置，数据库允许的连接数设置，常见问题。A)应用的数据库连接池设置偏小，一旦数据库相应慢(新上线应用，缺少索引 等)则应。用排队严重，甚至雪崩，而遗憾的是数据库能力还远为用尽。B)不具备失效及时发现和重新链接数据库能力。C)隔离级别设置：RR 和 RC 下不同的表现。

1.12. 应用解耦通过应用访问数据库而不是直接访问，重要业务不能依赖低保障级别的系统，应用层重要业务和普通业务解耦，关键业务要独立。

1.13. 组件失效免疫能力单一应用，单一硬件，甚至单一基础设施，单一站点容灾，业务影响，故障恢复能力，要季度级别进行演练。

1.14. 为关键组件减负特别是数据库访问,数据库成本最高，扩展性最难，可用性保障最难，恢复难度和时间最大。减负：能不用就不用，使用最简单，成本最低的语句，避免大事务，慎用两阶段事务。

1.15. 建议灰度数据库减少发布时变更数据库对全局的影响,只有应用程序灰度是不够的，还要有专门的灰度数据库。在分库、读写分离架构下，一套含数据库的完整应用架构，变的很自然。所为灰度环境就是生产环境，生产数据，所影响的也是生产环境，只是范围比测试环境更广，更真实。其实就是小范围的生产环境。类似于游戏内测。

1.16. 建立高仿真架构体系

- 数据库，操作系统升级：应用是否适应，性能会变好， 还是变坏
- 应用上线发布，系统变更(列如换平台)，提前判断业务影响和性能瓶颈
- 应对突发交易量，例如双十一，性能极限在哪里，瓶颈在哪里。

1.17. 容灾保障高可用是运维核心要求，容灾是最后屏障例如 双活比单活好，MGR比复制架构好，重要系统要做好高可用，容灾建设。

1.18. 多中心建设冗余是基础，多中心建设是为了提升容灾能力和扩展能力，并保障业务。

1.19. 应用和数据库是一个整体应用和运维人员一起，解决应用解耦，数据库解耦，追账补数，业务监控，应用路由，故障切换等。可用性，效率，故障恢复等方面都要一起参与。

1.20. 性能提升开源数据库使用应该合理且有效的结合周边的其他类型数据库，做到性能最大化。比如：Redis，MongoDB，ES，ClickHouse等。

二、总结最适合的架构是结合软件特性和业务场景，又能取得成本收益平衡。大数据情况下可以是利用读写分离，分库分表，但要选择合适的。不适合分库的应该考虑 竭尽所能把核心库做小，然后通过垂直扩展来扩容。用尽各种技术， 高可用 和 容灾手段保证其可用。