

DBA面试题合集

来自： 小6互联网求职面试



马听

2024年01月30日 14:29



扫码加入  
查看更多优质内容

这里总结星球内分享过的，选出一些个人认为比较重要的一些DBA面试题，方便过完年看机会的同学直接复习。

会先贴上问题，朋友们可以先看问题，看自己能否答上来。然后在文章下方有解析。

面试题

- 1 主从切换时怎么检测 MySQL 主库是否有问题？
- 2 如果让你写 MySQL 高可用，会考虑怎么实现？
- 3 orchestrator 故障恢复原理
- 4 讲一下你熟悉的高可用方案，有哪些特点？
- 5 MySQL常见存储引擎的特点
- 6 MySQL四种事务隔离级别分别是什么？区别是？
- 7 MySQL事务ACID特性分别是什么，都是怎样实现的？
- 8 MySQL主从复制的原理
- 9 Binlog和Redo log的区别
- 10 常用备份工具，备份和恢复策略是怎样的？官方逻辑备份工具mysqldump
- 11 MySQL常见监控项
- 12 一条update会经历哪些过程
- 13 MySQL从库延迟高，怎么确定是哪条SQL导致的？
- 14 MySQL CPU 高负载怎么排查
- 15 怎么分析慢查询
- 16 事务繁忙的 MySQL，物理机宕机，启动后会发生什么？具体怎么判断哪些要回滚，哪些要提交
- 17 怎么判断主从延迟
- 18 一个实例活跃线程个数多少以下算合理？
- 19 你是怎样优化SQL的？怎样决定应该在哪个字段添加索引？
- 20 在数据库升级或者迁移项目中，你是如何确保数据完整性和最小停机时间的？
- 21 如何确保备份完整性？
- 22 大数据量场景，你是怎样优化数据库备份和恢复流程的？
- 23 你怎么看待云数据库？
- 24 Redis常见数据类型
- 25 Redis怎么处理数据持久化的？
- 26 Redis常用监控项
- 27 Redis主从复制配置，以及原理
- 28 MongoDB跟传统关系型数据库的主要区别是什么？
- 29 MongoDB的集合和文档是什么？
- 30 MongoDB怎样备份和恢复
- 31 MongoDB高可用方案有哪些？
- 32 你是怎样跟开发团队合作的，特别是数据库设计和性能优化方面
- 33 MySQL两阶段提交是什么？
- 34 落盘次数
- 35 为什么两阶段提交能保证数据安全呢？
- 36 假设没有两阶段提交，会有什么问题

解析

- 1 主从切换时怎么检测 MySQL 主库是否有问题？

每隔几秒执行一次select user();如果累计3次都获取不到结果，则切换；当然还有其他的一些方案，比如查询具体某一张表，或者执行update的方式。这个要看具体业务场景。

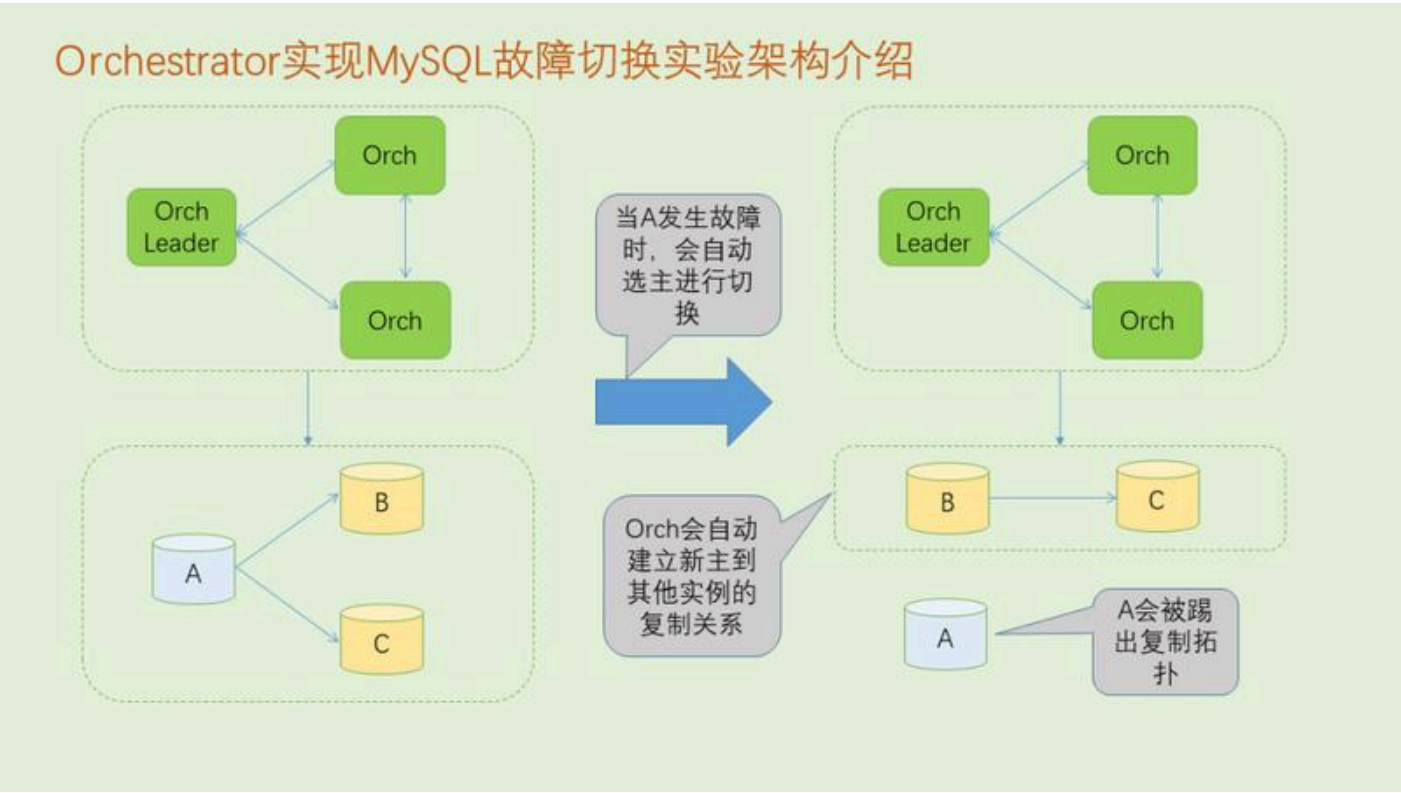
- 2 如果让你写 MySQL 高可用，会考虑怎么实现？

可以考虑从两个方向考虑，第一种，类似Orch的方案，也就是借助第三方工具去探测主库，当然，第三方工具可能有至少3台机器，当3台机器有两台认为主库宕机，则进行切换第二种，类似MGR的，就是节点之间互相探测，如果多数

节点认为主库宕机了，则切换而切换又涉及到选主，可以根据优先级，以及从库和主库的延迟，优先级高的，以及延迟底的，优先考虑作为新主

3 orchestrator 故障恢复原理

如图，Orch负责监听MySQL一主两从架构中三个MySQL实例（A、B和C）的状态，当A发生故障时，Orch会自动从B和C中选择一个新主进行切换，假设B为新主，Orch会自动创建B到C的复制关系，旧主会被踢出复制拓扑（如果旧主需要重新加入拓扑，需要手动与B或者C建立复制关系）。在切换之后，如果为了客户端能正常找到新主，可能会涉及到VIP修改、DNS修改等操作，可以放在PostFailoverProcesses的Hook中



4 讲一下你熟悉的高可用方案，有哪些特点？

这个建议是参考2中两种类型高可用方案，每一种选一个最具代表的，比如第三方工具投票Orch，自身集群投票的MGR。Orch的原理在3里面讲了，关于MGR，它的成员之间会互相发送检测消息，当服务器 A 在给定的时间内没有接收到服务器 B 的消息时，会发生超时并产生怀疑。之后，如果小组同意怀疑可能是真的，那么小组就会认定某个服务器确实出了故障，这意味着组中其他成员采取协调一致的决定来驱逐给定的成员。

5 MySQL常见存储引擎的特点

- InnoDB，支持事务、行锁、MVCC
- MyISAM，不支持事务，表锁设定
- MEMORY，全在内存中，读取快，重启会丢数据
- CSV，数据文件就存储为CSV格式，常用在数据导出或者临时存储的场景

6 MySQL四种事务隔离级别分别是什么？区别是？

- Read uncommitted（读未提交，简称：RU）：所有事务都可以看到其它未提交的事务的执行结果。可能会出现脏读。
- Read Committed（读已提交，简称：RC）：一个事务只能看见已经提交事务所做的改变。因为同一事务的其它实例在该实例处理期间可能会有新的 commit，所以可能出现幻读。
- Repeatable Read（可重复读，简称：RR）：这是MySQL的默认事务隔离级别，它确保同一事务使用同样的查询语句读取数据时，会看到同样的数据行。消除了脏读、不可重复读，默认也不会出现幻读。
- Serializable（串行）：这是最高的隔离级别，它通过强制事务排序，使之不可能相互冲突，从而解决幻读问题。

7 MySQL事务ACID特性分别是什么，都是怎样实现的？

- atomicity（原子性）：一个事务要么全执行，要么全都不执行；通过Undo Log来实现的。
- consistency（一致性）：在事务开始和完成时，数据都必须保持一致状态；通过Redo Log和Undo Log来实现一致性
- isolation（隔离性）：事务处理过程中的中间状态对外部是不可见的；通过锁来实现写和写之间事务的隔离性，通过MVCC来实现读和写的隔离性

durability（持久性）：事务完成之后，它对于数据的修改是永久性的，即使发生了断电也能恢复数据。Redo Log实现了MySQL的持久性

8 MySQL主从复制的原理

主库必须开启二进制日志

当主库有写操作时（比如insert、update、delete），会记录到主库的Binlog中

从库通过IO线程读取主库的Binlog里面的内容，传给从库的Relay Log（中继日志）

从库的sql线程负责读取它的relay log里的信息并应用到数据库中

9 Binlog和Redo log的区别

Redo Log是在InnoDB层产生的。Binlog不单单记录InnoDB的修改，也记录其他任何存储引擎的修改。

Redo Log 是物理逻辑格式的日志，记录的是每页的修改。Binlog 是一种逻辑日志，记录的是对应的变更SQL。

在事务进行中会不断地写入Redo Log，而Binlog只在事务提交时写入一次

Redo Log是循环写的，比如一共有3个redo log文件，会依次往三个文件中写入redo log，如果第3个文件写满了，又会回到第一个Redo log文件写入，而Binlog是写到一定大小会切换到下一个，不会覆盖之前的日志。

10 常用备份工具，备份和恢复策略是怎样的？官方逻辑备份工具mysqldump

多线程逻辑备份工具mydumper

物理备份工具XtraBackup

Clone Plugin

份和恢复策略：每天凌晨物理全备，其他时间Binlog备份，定期验证备份可恢复性，恢复过程规范化，脚本化。

11 MySQL常见监控项

系统相关

系统cpu\_iowait

系统cpu负载

系统内存交换

系统文件描述符

系统网络流量（进出）

磁盘利用率

MySQL 状态相关

MySQL 进程状态

QPS

TPS

慢查询数量

锁相关

表锁

行锁

死锁

锁等待

连接相关

最大连接数

活跃连接

主从

io sql线程

从库read\_only

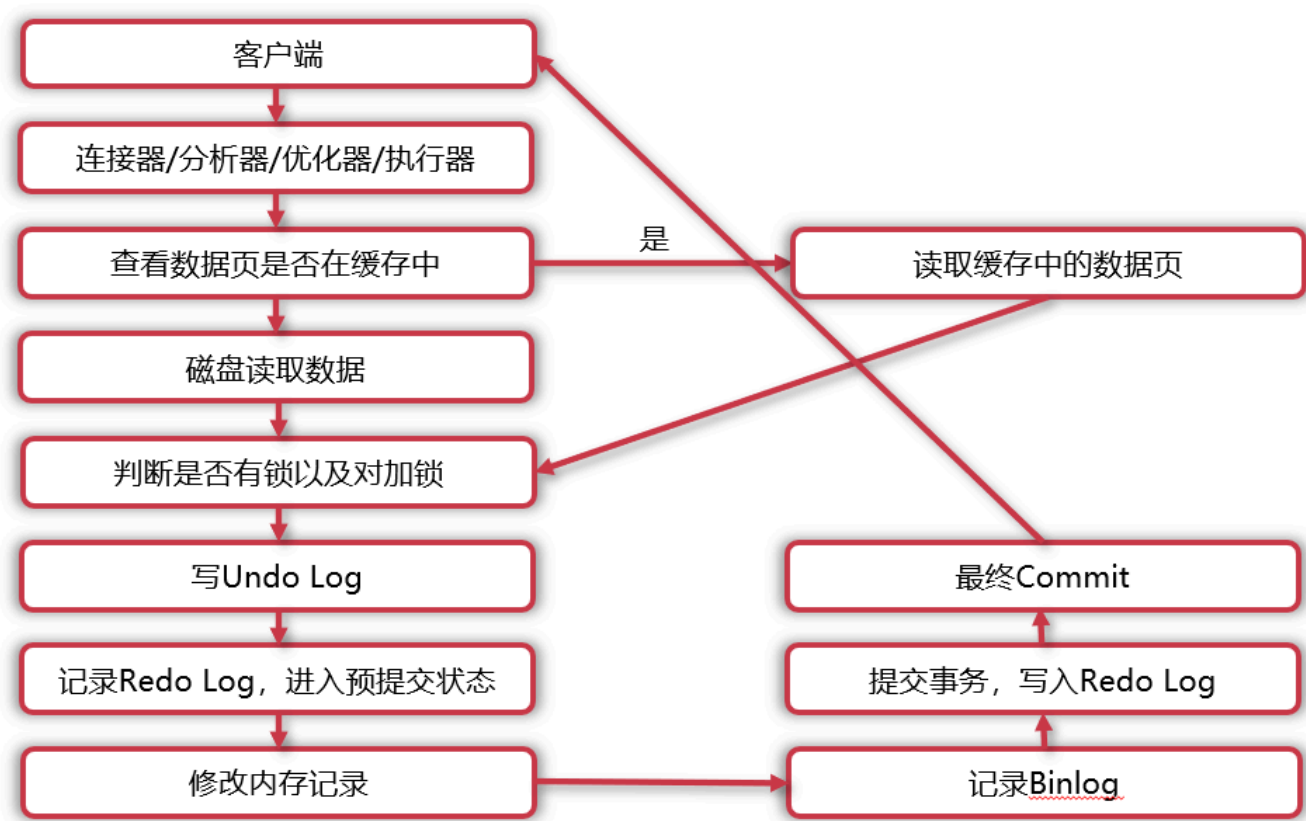
主从延迟

主从切换

业务相关  
主键自增值  
表数据量

12 一条update会经历哪些过程

# 一条更新语句的生命历程



首先是连上连接器，负责跟客户端建立连接、获取权限、维持和管理连接  
分析器会先做词法分析。需要确定SQL语句中的字符串分表是什么，代表什么？  
再做语法分析，判断输入的SQL是否满足MySQL语法。  
再经过优化器，决定是否走索引，走哪个索引，或者决定关联表的顺序  
执行器负责具体的执行，  
进入到存储引擎层，  
首先查看数据页是否在缓存中，如果在，就读取数据，如果不在，就从磁盘中读取  
判断要操作的记录是否存在锁，如果存在，就等待  
如果不存在，就对要修改的记录行加锁  
再写Undo Log  
记录Redo Log，进入预提交状态  
修改内存中的记录  
记录Binlog  
提交事务，写入Redo Log，  
最终Commit

13 MySQL从库延迟高，怎么确定是哪条SQL导致的？

1、从库show processlist;因为已经发现从库延迟了，假如是某个大事务导致的，那说明在主库已经提交了，正在从库执行，这个时候就可以找到具体的SQL语句；2、主库慢查询，这种导致主从延迟的，很可能在主库就已经是慢查询了，当然有时候，可能因为一些DDL操作导致的延迟，要慢查询日志能打印DDL，需要开启log\_slow\_admin\_statements，可以记录管理语句（比如alter table语句、create index语句等）3、如果前面两种方式还没找到，就解析最近的Binlog，看是否是因为主库增删改并发大  
降低延迟的处理办法：1、假如是大事务，比如一条delete删除上千万数据，那就拆分成小事务，每次删1000行，业务低峰操作2、如果是主库并发大，那就在从库开启多线程复制3、还有就是可以调整从库的innodb\_flush\_log\_at\_trx\_commit和sync\_binlog，都临时调整成不为1的值，从库延迟也能得到缓解。4、当然，有些公司为了节约成本，从库配置很差，就可以考虑提高配置。

14 MySQL CPU 高负载怎么排查

查看进程列表：用top查看哪些进程占用了大量CPU资源

检查MySQL连接：在MySQL中，执行show processlist，查看是否有正在执行的很占CPU资源的SQL，以前就遇到过，研发执行全表delete，导致CPU跑慢的情况

检查慢查询日志：查看慢查询日志，看从CPU高负载开始到现在，所有的慢查询，可以确定是不是慢查询导致的，如果有慢查询，就需要考虑是否要添加索引，是否要优化SQL等

观察其他指标：比如是否有锁等待，QPS和TPS是否有暴涨的情况等。

## 15 怎么分析慢查询

explain：查看SQL执行类型，扫描行数和是否走索引等；

show profile：确定 SQL 执行过程具体在哪个过程耗时比较久；

trace：查看优化器如何选择执行计划。

## 16 事务繁忙的 MySQL，物理机宕机，启动后会发生什么？具体怎么判断哪些要回滚，哪些要提交

在事务执行过程中，首先会将操作写入redo log，并刷新到磁盘。宕机恢复后，MySQL会通过检查redo log来确定哪些事务已经完成但尚未提交，这些事务将会被提交。

当MySQL发现某个事务在宕机时尚未完成，它会利用undo log来回滚这个事务。

## 17 怎么判断主从延迟

Seconds\_Behind\_Master

一种常规的方法就是 show slave status 查看 Seconds\_Behind\_Master，这个参数表示从库延迟的秒数。如果是0，表示可能没有延迟。这里为什么是可能呢？

当从库正在主动处理更新时，此字段显示从库上的当前时间戳与从库上当前正在处理的事件的主库上记录的原始时间戳之间的差异。

当副本上当前没有处理任何事件时，该值为 0

在某些情况下，Seconds\_Behind\_Master 并不一定准确。比如网络中断时，Seconds\_Behind\_Master = 0，并不能代表主从无延迟。因此，有比这个更准确的一种方法：对比位点或 GTID。

对比位点

如果是基于位点的复制，则判断 Master\_Log\_File 跟 Relay\_Master\_Log\_File 是否相等，如果 Relay\_Master\_Log\_File 落后 Master\_Log\_File，则表示主从存在延迟。

其中

Master\_Log\_File 表示 IO 线程正在读取的主库 binlog 文件名

Relay\_Master\_Log\_File 表示SQL 线程最近执行的事务对应的主库 binlog 文件名

或者判断 Read\_Master\_Log\_Pos 跟 Exec\_Master\_Log\_Pos 是否相等，如果后者落后前者很多，则表示延迟比较高。

其中

Read\_Master\_Log\_Pos 表示IO 线程正在读取的主库 binlog 文件中的位点

Exec\_Master\_Log\_Pos 表示 SQL 线程最近读取和执行的事务对应的主库 binlog 文件中的位点

对比GTID

如果开启了 GTID 复制，则可以对比 Retrieved\_Gtid\_Set 和 Executed\_Gtid\_Set 是否相等，如果 Executed\_Gtid\_Set 落后很多，则表示存在延迟。

其中

Retrieved\_Gtid\_Set：从库收到的所有日志的 GTID 集合；

Executed\_Gtid\_Set：从库已经执行完的 GTID 集合。

## 18 一个实例活跃线程个数多少以下算合理？



注意，这里是活跃线程个数，不是连接数，活跃线程个数一般建议是建议为机器核数的两倍值以下；  
因为活跃线程往往才会消耗CPU；  
通常也建议设置最大的活跃线程个数，MySQL可以用innodb\_thread\_concurrency参数来控制最大的活跃线程个数。

19 你是怎样优化SQL的？怎样决定应该在哪个字段添加索引？

根据执行计划explain，分析，是不是全表扫描，是不是filesort，另外就是关联查询使用了Block Nested-Loop Join或者Hash Join，需要在关联字段添加索引

20 在数据库升级或者迁移项目中，你是如何确保数据完整性和最小停机时间的？

一般是会提前配置一个老库到新库的同步，如果数据量比较大，就在迁移前一两天完成新老库的数据一致性校验，比如pt-table-checksum，而在迁移前，就只校验主键ID，这样会快很多，然后再进行迁移，这样停机的时间会很短。

21 如何确保备份完整性？

一般我们最主要的是使用两款备份工具，逻辑备份工具mysqldump和物理备份工具Xtrabackup，mysqldump的逻辑是读取MySQL所有的数据，并转换成SQL语句来创建备份文件，为了保证完整性，对于非事务表，mysqldump会锁定这些表，完成备份，确保备份过程中数据不会改变，对于InnoDB这类事务存储引擎，mysqldump提供了一种一致性备份的选项，使用事务来确保备份期间数据的一致性。Xtrabackup的逻辑是直接复制数据文件和事务日志，Xtrabackup在进行备份时会记录备份过程中正在改变的数据页，这样，即使在备份过程有数据变更，这些变更也会被记录，等数据文件传完之后，再应用这些变更。这样就可以保证备份期间的数据一致性。

22 大数据量场景，你是怎样优化数据库备份和恢复流程的？

通常建议使用物理备份工具备份，比如Xtrabackup，它可以直接复制数据文件，速度快，对MySQL的影响也小；如果有些场景一定要逻辑备份，可以使用多线程逻辑备份工具mydumper另外可以考虑使用增量备份；大数据量场景，恢复也很耗时间，可以增加服务器资源，优化配置，使用SSD等方式来加快恢复过程。另外就是要定期进行恢复测试，保证备份可用

23 你怎么看待云数据库？

云数据库有很多优势，比如高可用性，灵活的扩展性，并且可以快速适应业务需求的变化

24 Redis常见数据类型

string，字符串类型，一个key对应一个value  
hash，键值对集合  
list，字符串列表，按插入顺序排队  
set， string 类型的无序集合，集合中元素是唯一的  
zset，有序集合，它类似于SET，但每个元素都会关联一个分数（score），分数用来对元素进行排序。有序集合允许你按照分数顺序访问和检索元素，这使得它在需要排序或排行榜功能时非常有用  
HyperLogLog，用于估计集合基数的数据类型，可用在用户日活、月活统计。  
bitmap，用于存储位数据，支持位运算操作，适用于布尔值和计数器，最大的优势就是存储数据时，可以极大的节省空间  
GEO，存储地理位置数据，支持地理位置相关操作

25 Redis怎么处理数据持久化的？

RDB：把内存中的数据记录到磁盘中，相当于对全量数据做一个快照  
AOF：记录Redis收到的每一条写命令

26 Redis常用监控项

连接失败监控  
客户端连接数  
配置的最大内存  
最大内存策略  
角色监控  
复制状态监控  
延迟监控  
从库是否设置只读  
QPS  
网络总出/入量  
每秒输出/入量  
内存使用率  
内存碎片率  
缓存命中率  
key数量  
大key  
热key  
慢查询

27 Redis主从复制配置，以及原理

从实例执行  
replicaof 192.168.12.161 7001

复制原理：  
建立链接  
当执行完replicaof命令之后，从库给主节点发送psync命令，如果主节点设置了 requirepass 参数，则需要密码验证，从节点必须配置正确的 masterauth 才能通过验证，如果验证失败复制将停止。

全量复制  
首次同步时，主实例会执行bgsave生成RDB文件，再传到从库，从库收到RDB文件后，会先清空当前数据库，然后加载RDB文件。从节点从开始接收 RDB 到接收完成期间，如果有新的写操作，主实例会在把操作放在复制缓冲中，记录从节点开始接收 RDB 到接收完成期间主库收到的写操作  
当主实例发送完RDB文件后，就会把复制缓冲中的修改操作发给从库。从库再执行这些操作

命令持续复制  
当从节点接收到所有数据后，则完成了复制的建立流程。接下来主库会把每一次修改操作发送给从库，保证主从数据一致。

28 MongoDB跟传统关系型数据库的主要区别是什么？

MongoDB基于文档的数据模型，数据以 JSON-like（BSON）格式存储。这种模型提供了更大的灵活性，允许在同一集合（类似于表）中存储不同结构的文档。不需要预先定义模式

29 MongoDB的集合和文档是什么？

集合可以理解为MySQL的表  
文档可以理解为MySQL的数据行

30 MongoDB怎样备份和恢复

mongodump/mongorestore

mongoexport/mongoimport

### 31 MongoDB高可用方案有哪些？

副本集和分片集群

### 32 你是怎样跟开发团队合作的，特别是数据库设计和性能优化方面

沟通非常重要，需要确保数据库设计符合项目的需求，我一般会参与早期的设计讨论，确保数据表的高效和可扩展性，在性能优化方面，我会部署完整的数据库监控系统，并与开发团队共享这些数据，共同发现和解决潜在的问题，另外也会根据公司实际情况制定数据库最佳实践，这些沟通协作，可以避免很多数据库的问题

### 33 MySQL两阶段提交是什么？

第一阶段：写Redo Log

第二阶段：写BinlogRedo Log 的Commit

### 34 落盘次数

如果innodb\_flush\_log\_at\_trx\_commit设置为1  
并且sync\_binlog也设置为1

会涉及多次落盘，  
Redo log在Prepare阶段持续落盘；  
Redo log、Binlog在提交阶段的落盘。

### 35 为什么两阶段提交能保证数据安全呢？

再来聊下。其实两阶段提交的目的也是为了崩溃恢复：如果MySQL在1和2中间，崩溃了因为Binlog还没写，Redo log也没提交，那MySQL启动之后，事务回滚就行如果MySQL在2和3中间，崩溃了就需要做一些判断1 如果Redo log是完整的，则事务直接提交2 如果Redo log里的事务只是Prepare状态，就需要判断Binlog是否完整：Binlog完整，就提交事务，Binlog不完整，就回滚事务。

### 36 假设没有两阶段提交，会有什么问题

假设先Redo log落盘了，Binlog后落盘，比如Redo Log落盘之后，MySQL崩溃了，因为Redo log记录在磁盘中，可以把数据恢复回来，但是Binlog因为没完成落盘，那binlog就少一条记录，从库就会少一次变更假设先Binlog落盘，再Redo Log落盘，如果写完Binlog，MySQL崩溃了，因为Redo Log没全部完成记录，那这个事务就不会生效，但是Binlog中多出一条记录，那从库可能就会多了一次变更。