mq与mysql的redo log

怎么保证mq消息的可靠性?

所谓保证消息可靠性就是要保证消息不丢失

你要保证消息不丢，是不是得先知道消息啥时候会丢？那消息啥时候会丢呢?

- 第1，是不是从生产者发送消息给mq这个过程会丢。
- 第2，mq自己也可能会把消息弄丢，比如mq宕机重启了。
- 第3，mq把消息发送给消费者也可能会丢，或者消费者拿到消息还没来得及消费就出现了异常或者消费者重启了或者线程崩溃了等等

有3个点会丢。那应该怎么解决呢?

- 首先从生产者发送到mq这里，可以使用ack机制。当mq收到消息，给生产者回复一个ack即可。
- 其次, mq自己把消息弄丢了怎么办？开启mq的持久化机制，把消息持久化到磁盘中，这样即便mq宕机重启，也还是能从磁盘中恢复消息。然后就是做mq集群，保证高可用，避免mq宕机呀
- 最后，消费者还没消费，消息就丢了。那就使用消费者的ack机制，当消费者消费完消息再给mq发送ack。mq收到ack后再删除消息。否则mq就重发消息。

编程的本质就是抽象, 我们将这些技术推广到其他中间件或者其他技术设计中!

把mq消息持久化，抽象一下。这个场景就是怎么让内存中的数据不丢失。解决方案把内存中的数据保存到磁盘。

将其推广到redis中，就变成怎么保证redis数据不丢失，如果redis把数据丢了怎么办呢?

- 所以就有了rdb aof这种持久化机制，rdb和aof又各有优劣，所以就有了混合持久化机制。
- 除了持久化，是不是还要保证redis高可用，所以就有了redis主从, 哨兵，集群。

那再想想mysql，mysql好像数据本来就是保存在磁盘中的

- 但是当我们执行insert插入一条数据，数据是直接被插入到磁盘中了吗?
  - 好像不是吧。数据会被放入到内存中的buffer pool中，然后在某个时间，被写到磁盘中。
- 那么问题又来了，那数据在buffer pool但还没有被保存到磁盘中时，mysql宕机了，那数据不就丢失了，那事务完整性不就被破坏了。怎么办?
  - 所以就有了redo log呀，mysql每次把某个磁盘页修改的数据记录在redo log中，然后刷盘redo log不就行了。
  - 如果buffer pool中的数据未刷盘就宕机了，那么就可以读取redo log来恢复数据。
  - 这也就是为什么说redo log能保证事务的完整性，让mysql能够做到崩溃回复。
- 那你有没有想过为什么修改buffer pool之后不直接刷盘呢，而去写redo log，去刷盘redo log?
  - 因为buffer pool刷盘是随机io，刷盘的时候需要找到某个磁盘页，然后修改，然后再去找另外一个磁盘页，再修改，本来磁盘操作就慢，随机io的话就更慢了

    - 但是redo log只记录在那个磁盘位置做了怎样的修改，刷盘时，只需要往redo log日志文件后追加就行了，这就是顺序io，不需要找这个磁盘页找那个磁盘页，磁盘的磁头只需要沿着一个方向移动就行了。
- 这就是从设计者的角度思考, 为什么这么设计?
- 那如果问你，为什么redo log里面要记录数据页的物理修改，你是不是就知道了。再问你为什么随机io比顺序io慢，你是不是也能答上来了。

如果把mq这里的ack机制再抽象一下呢？当A服务给B服务发送数据，需要保证B服务收到或者使用完数据后A服务再删除数据。这就是场景. 那么解决方案就是ack机制.

说到这儿你想到了什么？tcp的ack机制.

那么请问先有tcp还是先有mq，肯定先有tcp呀。所以你猜猜mq的ack机制跟谁学的？我也不知道，但是我猜就是跟tcp学的

那你为什么不能学？所以我们在做业务需求的时候, 遇到这种场景是不是就可以使用ack机制和超时重传呢?