一个小小的批量插入，被面试官追问了6次
==========

嗨，你好呀，我是哪吒。

面试经常被问到“MyBatis批量入库时，xml的foreach和java的foreach，性能上有什么区别？”。

首先需要明确一点，**优先使用批量插入，而不是在Java中通过循环单条插入。**

很多小伙伴都知道这个结论，但是，为啥？很少有人能说出个所以然来。

就算我不知道，你也不能反反复复问我“同一个问题”吧？

1、MyBatis批量入库时，xml的foreach和java的foreach，性能上有什么区别？

批量入库时，如果通过Java循环语句一条一条入库，每一条SQL都需要涉及到一次数据库的操作，包括网络IO以及磁盘IO，可想而知，这个效率是非常低下的。

xml中使用foreach的方式会一次性发送给数据库执行，只需要进行一次网络IO，提高了效率。

但是，xml中的foreach可能会导致内存溢出OOM问题，因为它会一次性将所有数据加载到内存中。而java中的foreach可以有效避免这个问题，因为它会分批次处理数据，每次只处理一部分数据，从而减少内存的使用。

如果操作比较复杂，例如需要进行复杂的计算或者转换，那么使用java中的foreach可能会更快，因为它可以直接利用java的强大功能，而不需要通过xml进行转换。

孰重孰轻，就需要面试官自己拿捏了~

2、在MyBatis中，对于`<foreach>`标签的使用，通常有几种常见的优化方法?

比如避免一次性传递过大的数据集合到foreach中，可以通过分批次处理数据或者在业务层先进行数据过滤和筛选。

预编译SQL语句、优化SQL语句，减少foreach编译的工作量。

对于重复执行的SQL语句，可以利用mybatis的缓存机制来减少数据库的访问次数。

对于关联查询，可以考虑使用mybatis的懒加载特性，延迟加载关联数据，减少一次性加载的数据量。

3、MyBatis foreach批量插入会有什么问题?

foreach在处理大量数据时会消耗大量内存。因为foreach需要将所有要插入的数据加载到内存中，如果数据量过大，可能会导致内存溢出。

有些数据库对单条SQL语句中可以插入的数据量有限制。如果超过这个限制，foreach生成的批量插入语句将无法执行。

使用foreach进行批量插入时，需要注意事务的管理。如果部分插入失败，可能需要进行回滚操作。

foreach会使SQL语句变得复杂，可能影响代码的可读性和可维护性。

4、当使用foreach进行批量插入时，如何处理可能出现的事务问题？内存不足怎么办?

本质上这两个是一个问题，就是SQL执行慢，一次性执行SQL数量大的问题。

大多数数据库都提供了事务管理功能，可以确保一组操作要么全部成功，要么全部失败。在执行批量插入操作前，开始一个数据库事务，如果所有插入操作

都成功，则提交事务；如果有任何一条插入操作失败，则回滚事务。

如果一次插入大量数据，可以考虑**分批插入**。这样，即使某一批插入失败，也不会影响到其他批次的插入。

优化foreach生成的SQL语句，避免因SQL语句过长或过于复杂而导致的问题。

比如MySQL的INSERT INTO ... VALUES语法 通常比使用foreach进行批量插入更高效，也更可靠。

5、MyBati foreach批量插入时如何处理死锁问题?

当使用MyBatis的foreach进行批量插入时，可能会遇到死锁问题。这主要是因为多个事务同时尝试获取相同的资源（如数据库的行或表），并且每个事务都在等待其他事务释放资源，从而导致了死锁。

（1）优化SQL语句

确保SQL语句尽可能高效，避免不必要的全表扫描或复杂的联接操作，这可以减少事务持有锁的时间，从而降低死锁的可能性。

不管遇到什么问题，你就回答优化SQL，基本上都没毛病。

（2）设置锁超时

为事务设置一个合理的锁超时时间，这样即使发生死锁，也不会导致系统长时间无响应。

（3）使用乐观锁

乐观锁是一种非阻塞性锁，它假设多个事务在同一时间不会冲突，因此不会像悲观锁那样在每次访问数据时都加锁。乐观锁通常用于读取频繁、写入较少的场景。

（4）分批插入

如果一次插入大量数据，可以考虑分批插入。这样，即使某一批插入失败，也不会影响到其他批次的插入。

（5）调整事务隔离级别

较低的隔离级别（如READ UNCOMMITTED）可能会减少死锁的发生，但可能会导致其他问题，如脏读或不可重复读。

6、mybatis foreach批量插入时如果数据库连接池耗尽，如何处理?

（1）增加最大连接数

数据库连接池耗尽了，增加最大连接数，这个回答，没毛病。

（2）优化SQL语句

减少每个连接的使用时间，从而减少连接池耗尽的可能性。

万变不离其宗，优化SQL，没毛病。

（3）分批插入

避免一次性占用过多的连接，从而减少连接池耗尽的可能性。

（4）调整事务隔离级别

降低事务隔离级别可以减少每个事务持有连接的时间，从而减少连接池耗尽的可能性。但需要注意，较低的事务隔离级别可能会导致其他问题，如脏读或不可重复读。

（5）使用更高效的批量插入方法

比如MySQL的INSERT INTO ... VALUES语法。这些方法通常比使用foreach进行批量插入更高效，也更节省连接资源。

感觉每道题的答案都是一样呢？这就对喽，数据库连接池耗尽，本质问题不就是入库的速度太慢了嘛。

（6）定期检查并关闭空闲时间过长的连接，以释放连接资源。

就前面的几个问题，做一个小总结，你会发现，它们的回答大差不差。

通过现象看本质，批量插入会有什么问题？事务问题？内存不足怎么办？如何处理死锁问题？数据库连接池耗尽，如何处理？

这些问题的本质都是因为SQL执行慢，一次性SQL数据量太大，事务提交太慢导致的。

回答的核心都是：如何降低单次事务时间？

1. 优化SQL语句
2. 分批插入
3. 调整事务隔离级别
4. 使用更高效的批量插入方法
 原文链接: https://juejin.cn/post/7359900973991362597