应用系统体系架构 — 作业2

# 学号： 522031910213 姓名： 朱涵 得分：

## A.

## 消息格式设计：

## 目前只有订单处理这一个WS端点 因此消息格式为简单的字符串 即订单处理成功 向前端发送“订单已处理成功！”的字符串 如果失败 发送“订单处理异常：{异常信息}"的字符串 以后需要引入其他端点（比如聊天室或是点赞通知） 可以使用json格式序列化的消息来封装更多信息

## 筛选客户端的方式设计：

## WS会维护一个sessionMap 使用userId作为键 session作为值 每当有新连接建立 根据session内的信息去数据库里查询对应的userId 并放入sessionMap中 调用发送消息方法时须指定userId参数 向指定用户的session发送消息 其他用户不会收到消息

## 为什么要选择线程安全的集合类型：

## Websocket服务器中 常常会有多个客户端的连接线程对于sessionMap进行操作（比如同时有两个用户建立新连接） 如果不是线程安全的 session的维护就会出现并发错误

## 另外 使用java封装好的线程安全类型集合 通常在并发场景下的读写性能也不错

## 我选择的类型为什么是安全的：

## 我选择了ConcurrentHashMap 其为HashMap的线程安全版本

## ConcurrentHashMap将哈希表分为许多段 每一个段内有很多桶 一个段维护自己的锁 使得多个线程可以执行并发写 并且任意多的线程可以执行并发读

## B.

## 如果数据库系统在事务执行过程中不断地将事务操作的结果执行落盘操作，会带来什么潜在问题？可以如何处理？

## 答：

## 即在事务执行过程中不断的把内存中的脏页落盘 采用“窃取”策略 虽然这种策略占用内存资源少 却会影响原子性 如果事务执行到一半出现系统崩溃 此时有一部分的操作已经落盘 而剩余操作还未落盘 事务的原子性被打破 数据也会不一致

## 可以使用Undo Log来保证事务的原子性 即做事务内的每一步操作前把数据项的旧值先写入日志 如若发送上述崩溃情况 重启系统后需要根据日志去撤销事务执行过的所有操作 来保证事务的原子性和一致性

## 如果数据库系统在事务执行提交后再将事务操作的结果执行落盘操作，会带来什么潜在问题？可以如何处理？

## 答：

## 若在事务执行提交后且在事务结果落盘前发生系统崩溃 事务的操作结果未能正确持久化 导致事务的持久性被打破

## 若采用“强制”策略 即每次事务提交都必须落盘 则可以先落盘后确认提交事务 确保事务的结果持久化

## 若采用“非强制”策略 即多次事务才进行一次落盘 则可以使用Redo Log来保证事务的持久性 即每一步操作前把数据项的新值写入日志 若发送上述崩溃情况 重启系统后线根据日志重做所有操作 来达到自动恢复事务结果的目的 保证事务的持久性和一致性