

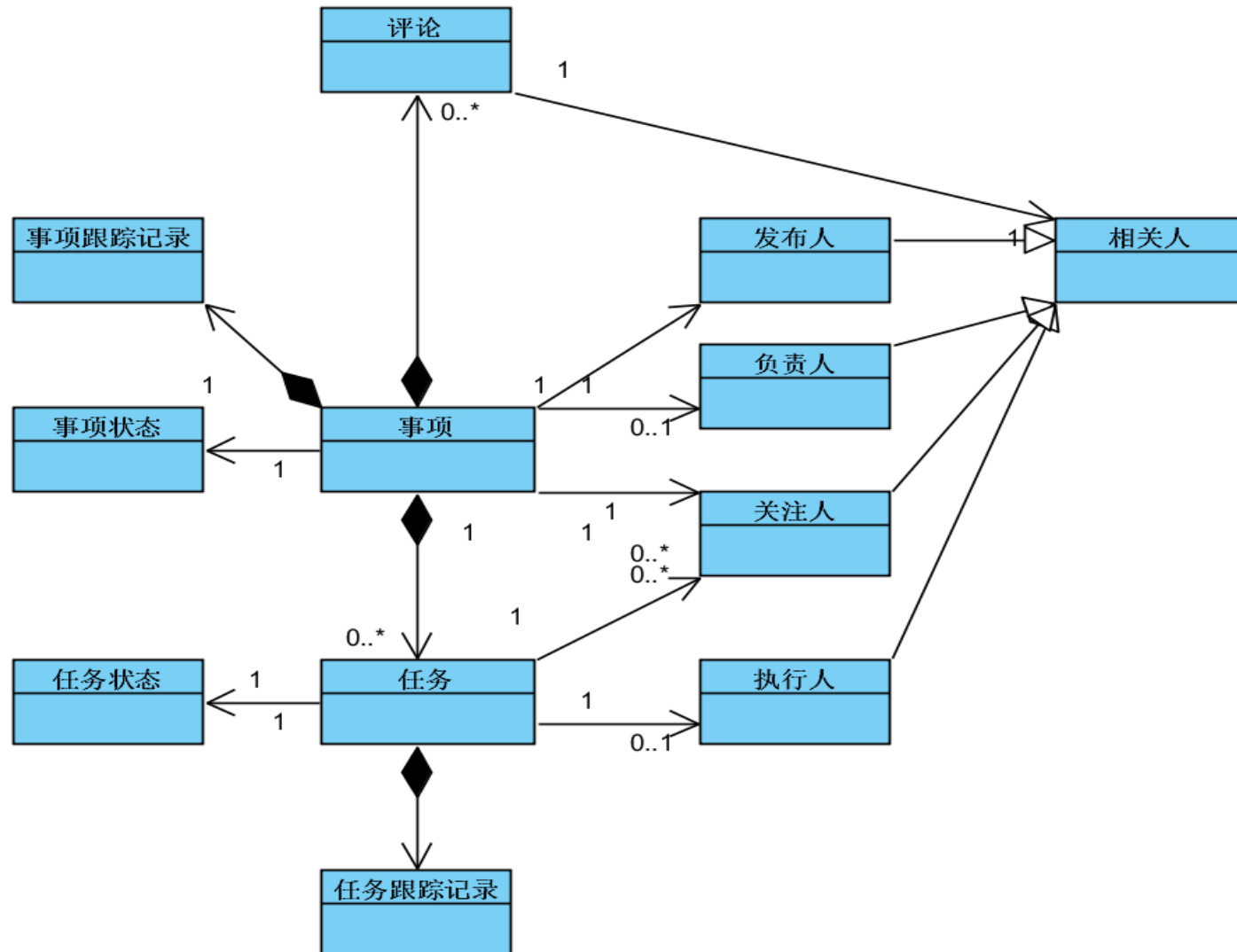
DDD实践中的一些思考

厦门云雾科技有限公司 王立

项目背景

公司为了便于对日常事项处理流程进行规范与跟踪管理，需要把管理办法固化到程序中，使得相关人员可以发布事项、分解任务、分配责任人、汇报进展、订阅进展、跟踪提醒、审计执行结果，并可以对事项和任务进行查询统计等，每个事项的参与方，根据自己在事项中扮演的角色，获取相关的操作权限，该项目涉及的业务实体包括：事项、任务、评论、操作日志等，业务角色包括发布人、负责人、执行人、关注人等，作为企业内部应用，对性能的要求不是很高。

分析模型



问题1：CQRS的精简与优化

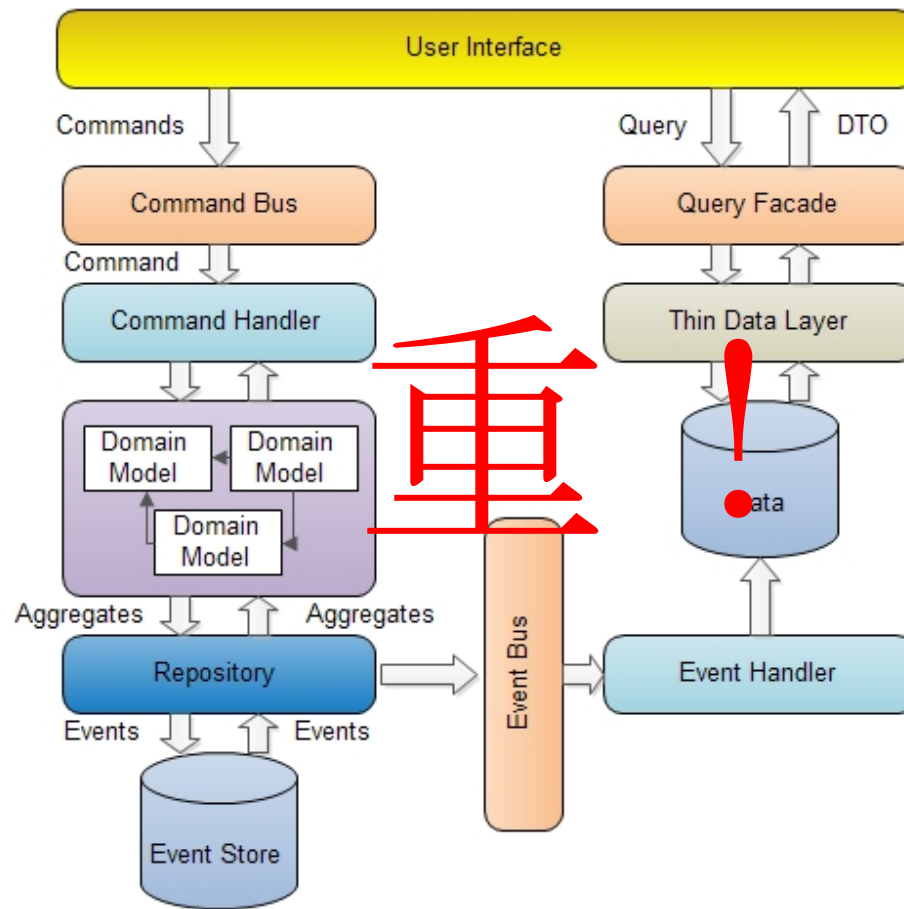
应用场景



问题与思考

- 1、列表的每个事项的进度需要统计所有任务的进度，是加载聚合进行计算还是直接从数据库查询统计？
- 2、每个事项需要根据当前登录者在事项中的角色，提供不同的操作入口，这类必须实时计算的结果，一定要DO参与吗？

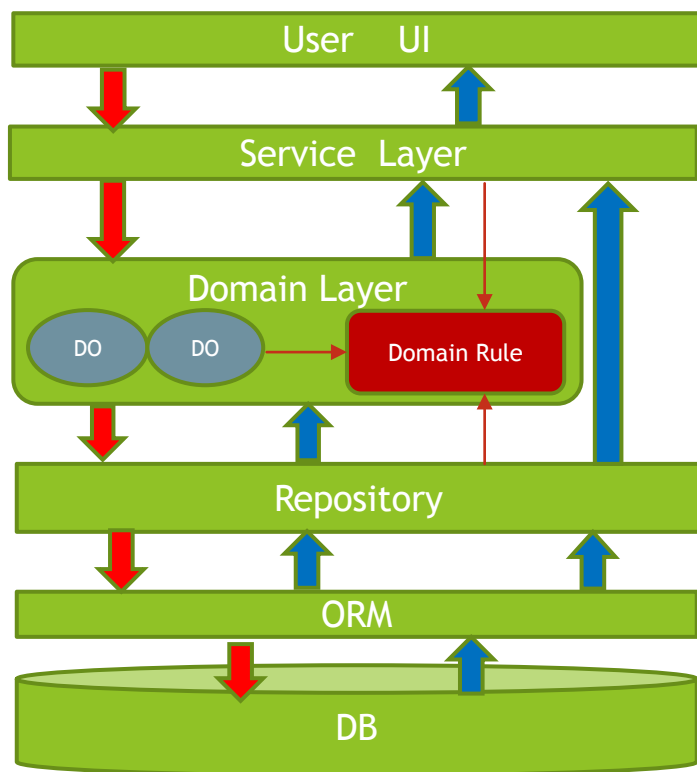
我们所熟知的CQRS



Event Source不是CQRS的必选项

它能解决第一个问题，但不能解决第二个问题

解决办法：轻量CQRS



- 通常查询可以绕过领域对象
- 只为写入操作提供领域对象。
- 实时记录事项聚合根的统计信息，使得查询结果可以不再需要聚合根实时统计。
- 必须实时计算的信息，既可以从领域对象中得到，也可以从仓储中直接得到，因为领域规则，可被剥离为独立的部分，被DO、服务层、仓储层共享。

读写分离的意义：
让领域模型和关系模型扬长避短
发挥作用！

粗箭头只代表数据的传输方向，不代表层的依赖关系

问题2、聚合根真的必须成为唯一的入口吗？








应用场景

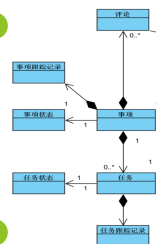


问题与思考

唯一聚合
根入口，
接口臃
肿...

于是接口分离

- >  CommonTask.java 194
- >  CommonThing.java 194
- >  ExecutedTask.java 198
- >  ExecutedThing.java 195
- >  ManagedTask.java 203
- >  ManagedThing.java 65
- >  PublishedThing.java 124



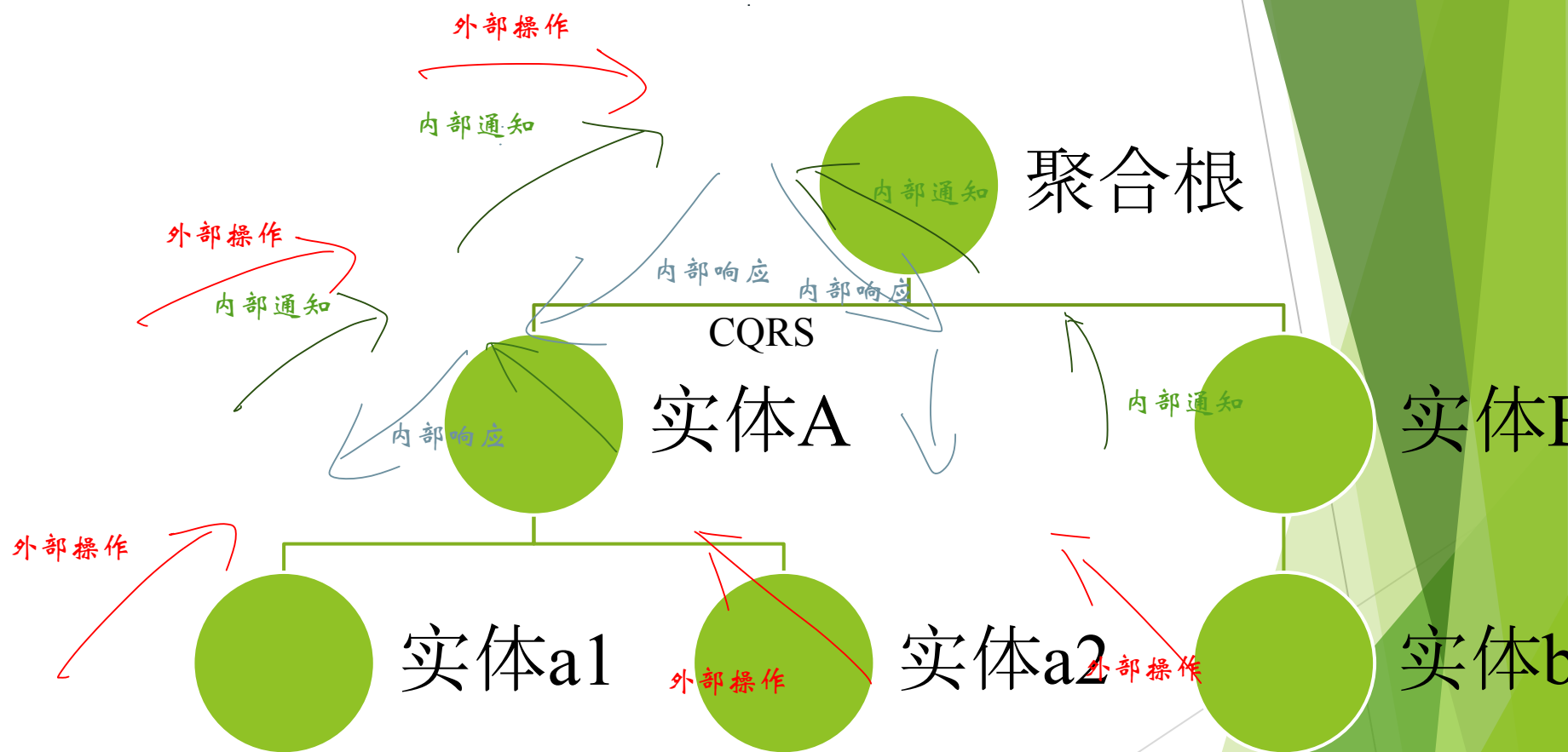
但还是要实例化整个聚合

为了改个任务标题，我还得造这么多对象？？

聚合根作为唯一的入口，是为了使用门面模式来封装聚合内部的复杂对象关系和保护数据一致性

那么“领域驱动”必须总是从聚合根开始操作聚合吗？

解决办法：观察者模式



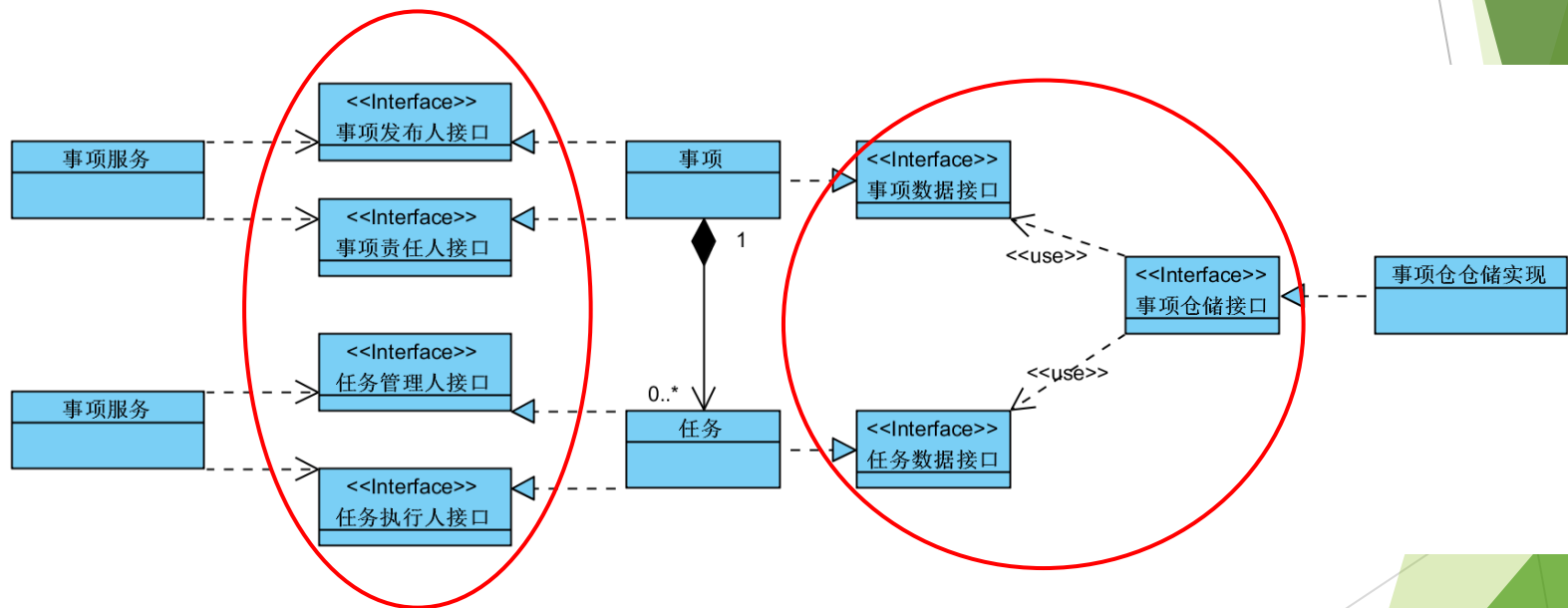
问题3：限制直接使用DO并管理接口

应 用 场 景

每个领域对象的客户端既有服务层的调用也有仓储层的调用，而且，任务对象的操作者包括责任人和执行人，不同使用者的接口需要隔离，避免不安全的操作。

限制直接使用DO

除了仓储，其它层不能直接使用DO



规范DO的创建和获取

- ▶ 只能通过仓储获取DO的接口，并且方法的意图必须明确：

loadTaskFromRepositoryForManager:从仓储获取用于管理的任务接口

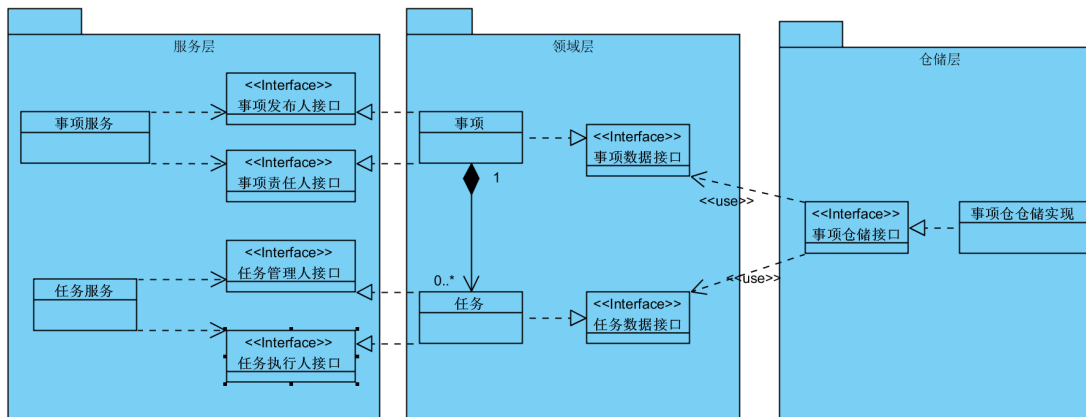
loadTaskFromRepositoryForExcutor:从仓储获取用于执行的任务接口

load方法的内部，会调用领域对象的不同构造函数，为了避免构造函数太多导致混乱，领域对象也可以不提供public的构造函数，只提供意义明确的静态方法来获取对应的实例。

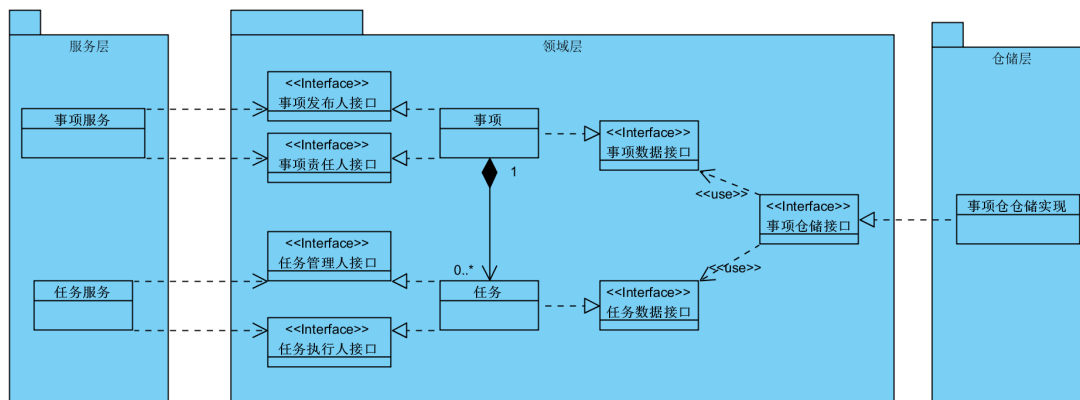
- ▶ 保存动作也只操作特定接口：

saveTask (TaskForRepository task) :使用TaskForRepository接口保存任务

接口管理



VS

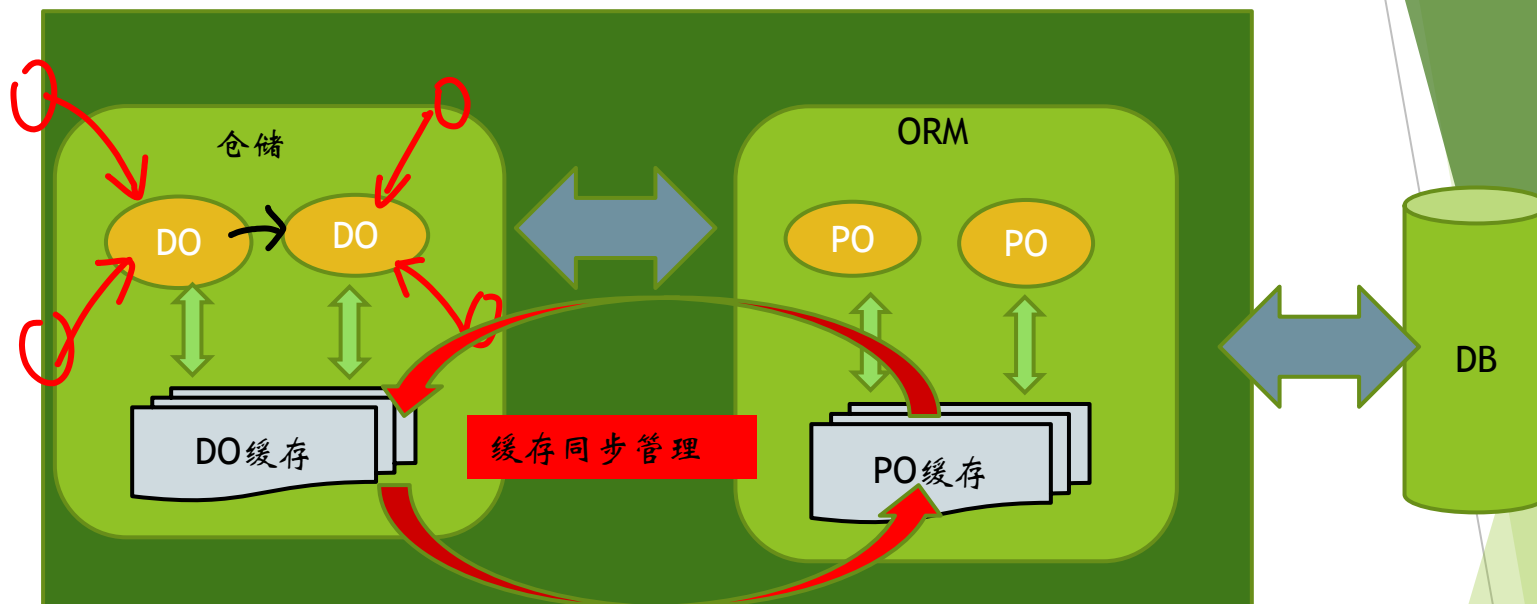


问题4、领域对象的缓存

应用场景

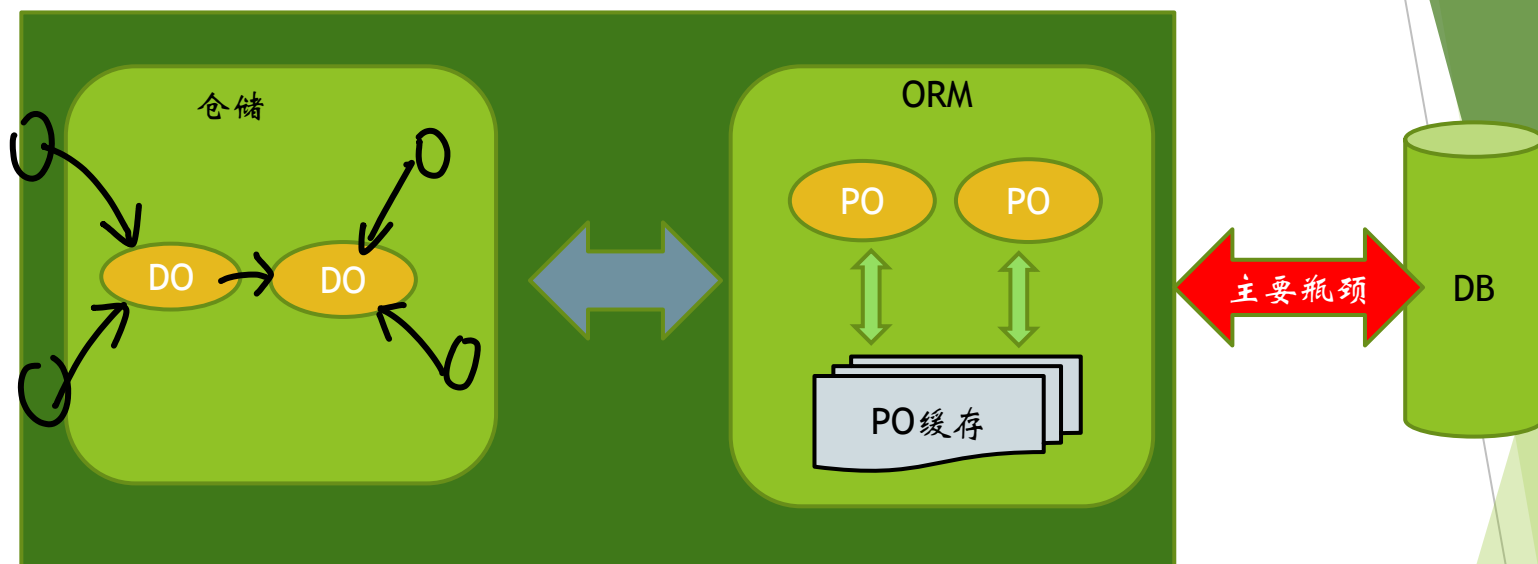
任务对象需要频繁引用聚合的其它成员，构建聚合的开销比较大，需要缓存。

问题与思考



- ▶ 二级缓存的管理复杂性问题。
- ▶ 缓存的序列化和反序列化的复杂性问题。

解决办法



创建领域对象的主要开销多数情况下主要是获取数据的过程，而不是装配领域对象的过程，所以，一般情况下默认不缓存领域对象，只缓存PO，除非装配领域对象本身的开销也很大。

谢谢！