## 领域模型

* 仓库
* 子库
* 出库单
* 入库单
* 出库单明细
* 入库单明细
* 出入库状态
* 物料库存
* 条码批次库存
* 非条码批次库存
* 批次
* 批次价格明细

## 业务规则

### 仓库

* 仓库跟组织挂钩
  + 总部库存（GCSM 库存服务不负责，由 【WMS-仓库管理系统】管理，页面虽然有查询总部库存功能，但GCSM不维护总部库存数据，只提供查询WMS总部库存功能）
  + 战区库存（暂无）
  + 服务区域库存（包含京东RTC仓、安时达托管仓）
  + 一级服务商库存
  + 二级服务商库存（暂无）
  + 网点库存
* 一个仓库对应多个库区，入整机、备件、资产、赠品、服务用机等
* 一个库区对应多个子库，如 好料、坏料、品质料等

### 库存

* 物料代码库存
* 物料批次库存
  + 条码批次库存
  + 非条码批次库存
    - 用于先进先出

### 出入库

出库单状态流转扣减库存规则

入库单状态流转扣减库存规则

出入库单流转组合情况

### 批次

* 批次作用
  + 先入先出
  + 不同批次对应不同的价格
* 批次如何管理
  + 第一次入CSM库存新增批次
    - 总部WMS 发货后新增批次
    - 电商退货新增批次
    - 具体新增批次业务单据类型参考 https://odocs.myoas.com/sheets/ZzkLVJljQeSG103Q/iB090《库存交易说明 + 业务单据码表【持续维护】》
    - 第一次入库场景，创建入库单会新增批次
  + 后续继承批次
    - 条码创建出库单时就会有批次
    - 非条码发货后才会有批次
      * 非条码会对应批次先入先出

## 技术设计

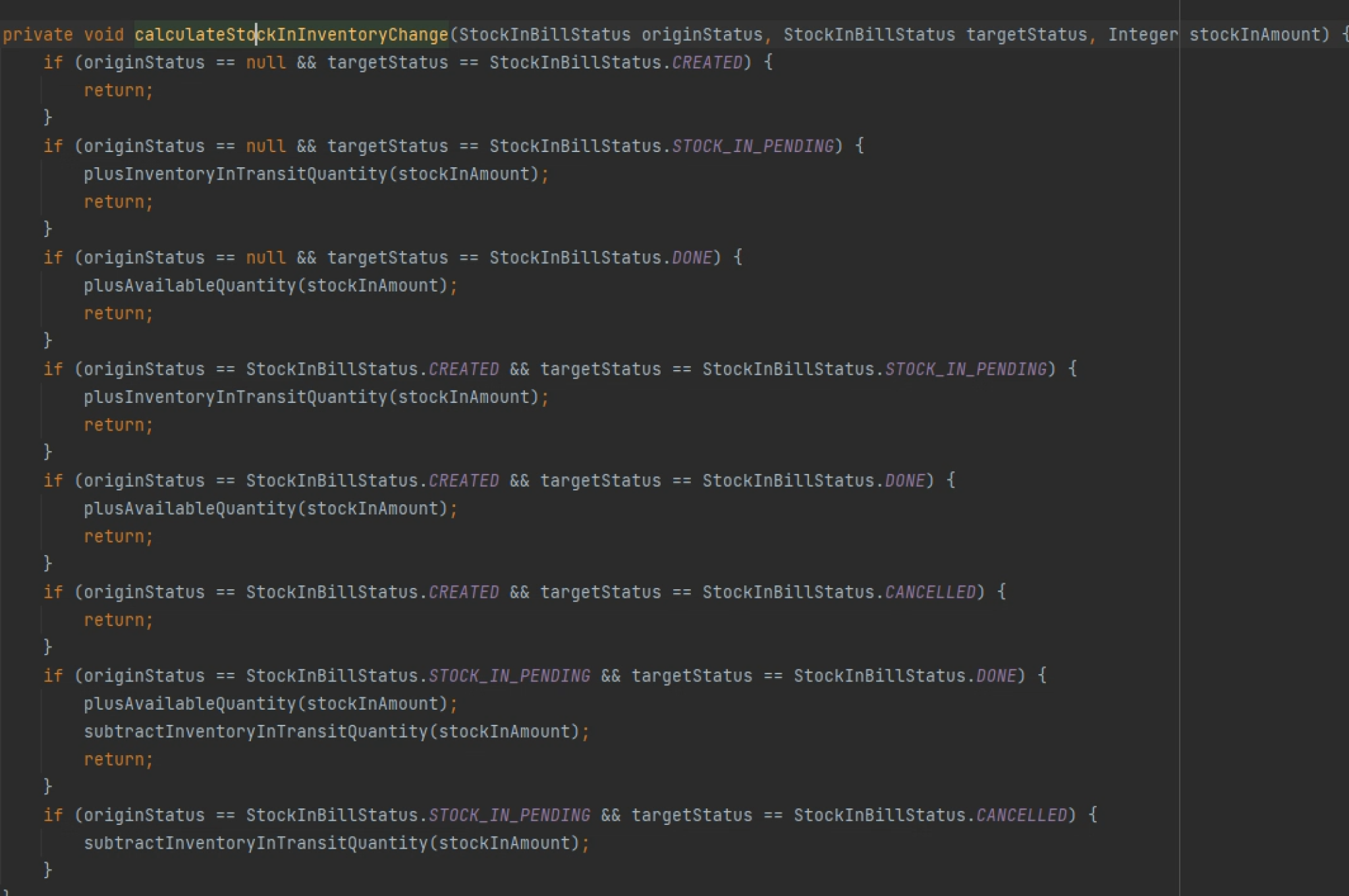
### 库存操作

#### 库存操作API 如何设计，如何使用？

* 按照虚拟资源拆分API
  + 子库间出入库（有实物出入库，有发货、收货
  + 库内转移（同一个仓库，不同子库移动）
  + 库存调整（单边出入库，盘亏盘盈）
  + 再针对批量、工单消耗和改单按场景拆分
    - 批量场景
    - 工单消耗和改单

#### 出入库单状态变更如何驱动库存更新？

* 入库单状态变更



* 出库单状态变更(类似上图)

#### 如何保证库存更新不丢失？

* 数据库原子更新操作 update set amount = amount +- 1
* 针对负库存，添加判断条件 update set amount = amount +- xx (where amount +- xx >= 0) if yy

UPDATE edInventoryInTransitQuantity},

gcsm\_inventory.deliver\_in\_transit\_quantity = gcsm\_inventory.deliver\_in\_transit\_quantity +

#{inventoryChange.addedDeliverInTransitQuantity},

gcsm\_inventory.total\_amount = gcsm\_inventory.total\_amount + #{inventoryChange.addedTotalAmount},

gcsm\_inventory.update\_by = #{updateBy.updateBy},

gcsm\_inventory.update\_time = #{updateBy.updateTime}

WHERE gcsm\_inventory.sub\_warehouse\_code = #{inventoryUpdateQueryDTO.subWarehouseCode} AND

gcsm\_inventory.material\_code = #{inventoryUpdateQueryDTO.materialCode}

<if test="!inventoryUpdateQueryDTO.negativeAvailableInventoryAmountAllowed">

AND gcsm\_inventory.available\_quantity + #{inventoryChange.addedAvailableQuantity} >= 0

</if>

#### 如何防止更新库存导致数据库死锁？

* 库存更新时按出入库明细的物料编码进行排序，然后再更新库存

#### 如何避免短时间内同一单据多次操作库存？

* 分布式锁

通过businessType 和 businessCode 添加分布式锁，防止短时间内操作两次库存

public <C extends InventoryOperationCommandDTO> void operate(Consumer<C> inventoryOperation, C inventoryOperationCommandDTO) {

String inventoryLockKey = REDIS\_KEY\_PREFIX + ":" + inventoryOperationCommandDTO.getBusinessType() + ":" + inventoryOperationCommandDTO.getBusinessCode() + ":" + INVENTORY\_LOCK\_KEY;

if (!lockClient.acquireLock(inventoryLockKey, new ExpiredTime(TIME\_TO\_LIVE, TimeUnit.SECONDS))) {

throw new AbortedException("acquire inventory lock error");

}

try {

inventoryOperation.accept(inventoryOperationCommandDTO);

} finally {

lockClient.releaseLock(inventoryLockKey);

}

}

具体见 InventoryOperationLockService.java

* 库存操作幂等性

根据 businessType 和 businessCode 作为幂等标识

通过 businessType 和 businessCode 查出对应的出入库单，根据出入库单状态作为判断条件，如果已经达到对应的状态，就返回成功

@Override

public StockOutBillDTO stockIn(InventoryOperationCommandDTO inventoryOperationCommandDTO) {

Function<InventoryOperationCommandDTO, StockOutBillDTO> stockIn = (operationCommandDTO) -> {

InventoryBill latestInventoryBill = inventoryBillService.findLatestInventoryBill(operationCommandDTO);

if (latestInventoryBill.isDone()) {

return StockOutBillConverter.INSTANCE.toDTO(latestInventoryBill.getStockOutBill());

}

InventoryOperationValidator.validateInventoryBillTransit(latestInventoryBill);

inventorySubWarehouseCheckService.validateSubWarehouseTypeNotCheckForNonStockIn(latestInventoryBill.getPairSubWarehouseCodes());

return inventoryOperationService.updateInventoryBillToDoneStatus(latestInventoryBill);

};

return inventoryOperationLockService.operate(stockIn, inventoryOperationCommandDTO);

}

具体可以参考 InventoryTransfersBetweenWarehouseServiceImpl#stockIn()

* 库存批量操作幂等性设计

添加 InventoryBillBatchRecord，通过单据号+批量操作号 做幂等

@Override

@Transactional(rollbackFor = Exception.class)

public List<StockOutBillDTO> batchInventoryOperate(BatchInventoryOperationCommandDTO batchInventoryOperationCommandDTO) {

Function<BatchInventoryOperationCommandDTO, List<StockOutBillDTO>> batchOperate = (commandDTO) -> {

Optional<InventoryBillBatchRecord> record = inventoryBillBatchRecordDomainRepository.findByBusinessCodeAndBatchOperationNumber(commandDTO.getBusinessCode(), commandDTO.getBatchOperationNumber());

if (record.isPresent()) {

return stockOutBillDomainRepository.findAllByCodes(record.get().getStockOutBillCodes())

.stream()

.map(StockOutBillConverter.INSTANCE::toDTO)

.collect(Collectors.toList());

}

if (!CollectionUtils.isEmpty(commandDTO.getRollbackBatchOperationNumbers())) {

changeWorkOrderInventoryWithCatchable(commandDTO);

}

return consumeWorkOrderInventory(commandDTO);

};

return inventoryOperationLockService.operate(batchOperate, batchInventoryOperationCommandDTO);

}

具体可以参考 WorkOrderInventoryOperationServiceImpl#batchInventoryOperate()

### 批次

#### 什么时候创建批次？

根据业务单据类型判断是否需要创建批次（原则就是第一次GCSM入库）

创建入库单时判断是否需要创建批次

#### 什么时候继承批次？

已经从总部发货并入库GCSM库存，货物后续流转时需要继承批次

库存操作时都会创建入库单、出库单，出库时一定会有批次，入库时需要继承对应的批次

所以在创建出库单时需要返回对应的批次信息，这样创建入库单时就可以继承出库物料的批次