# StockDashboard 前后端实现与数据来源审查指引

本文汇总 http://localhost:5200/stock/dashboard 路径对应的前端、后端代码位置与数据库表字段来源，便于代码 Review 与数据校验。

## 一、t\_ticket 实体/Mapper 字段映射

* 实体文件：ticket-show/src/main/java/com/ticketsystem/show/entity/Ticket.java
* 表名：t\_ticket
* 字段映射：
  + id -> id（主键，自增）
  + show\_id -> showId
  + session\_id -> sessionId
  + name -> name
  + price -> price（BigDecimal）
  + total\_count -> totalCount
  + remain\_count -> remainCount
  + limit\_count -> limitCount
  + status -> status（0 未开售、1 售票中、2 已售罄、3 停售）
  + create\_time -> createTime（自动填充 INSERT）
  + update\_time -> updateTime（自动填充 INSERT\_UPDATE）
  + is\_deleted -> isDeleted（逻辑删除）
* Mapper 文件：ticket-show/src/main/java/com/ticketsystem/show/mapper/TicketMapper.java
  + 主要 SQL：
    - 锁定库存（扣减 remain\_count）
    - UPDATE t\_ticket  
      SET remain\_count = remain\_count - #{quantity},  
       update\_time = CURRENT\_TIMESTAMP  
      WHERE id = #{ticketId}  
       AND remain\_count >= #{quantity};
    - 释放库存（回加 remain\_count）
    - UPDATE t\_ticket  
      SET remain\_count = remain\_count + #{quantity}  
      WHERE id = #{ticketId};
    - 确认扣减（仅更新时间戳，不再更改数量）
    - UPDATE t\_ticket  
      SET update\_time = CURRENT\_TIMESTAMP  
      WHERE id = #{ticketId};

## 二、StockDashboard 前端表格各列字段映射对照

* 前端页面：ticket-frontend/src/views/StockDashboard.vue
* 前端调用：ticket-frontend/src/api/ticket.ts 的 ticketApi.getAllTicketStock()
  + 路径：GET /ticket/stock/all（经网关转发至 ticket-show 服务）
* 后端聚合返回结构（由 TicketServiceImpl.getAllTicketStockInfo() 组装）：
  + 每条记录：{ ticket: Ticket, stock: TicketStock | null }
  + stock 为 null 时，前端显示 0 或“无库存”
* 字段对照（前端列 -> 后端字段 -> 数据来源表）
  + 票档ID -> row.ticket.id -> t\_ticket.id
  + 票档名称 -> row.ticket.name -> t\_ticket.name
  + 价格 -> row.ticket.price -> t\_ticket.price
  + 总库存 -> row.stock.totalStock -> t\_ticket\_stock.total\_stock
  + 可用库存 -> row.stock.availableStock -> 服务端计算 total\_stock - locked\_stock - sold\_stock（TicketServiceImpl 设置）
  + 锁定库存 -> row.stock.lockedStock -> t\_ticket\_stock.locked\_stock
  + 已售库存 -> row.stock.soldStock -> t\_ticket\_stock.sold\_stock
  + 库存状态（售罄/紧张/充足）-> 前端根据 availableStock 判断
  + 库存占用率（进度条）-> 前端计算 (totalStock - availableStock) / totalStock
* 页头统计（KPI）
  + 座位档数 -> stockData.length -> 记录条数
  + 总库存 -> sum(row.stock.totalStock) -> Σ t\_ticket\_stock.total\_stock
  + 可用库存 -> sum(row.stock.availableStock) -> Σ (total - locked - sold)
  + 锁定库存 -> sum(row.stock.lockedStock) -> Σ locked\_stock
  + 已售库存（用于图表） -> sum(row.stock.soldStock) -> Σ sold\_stock

## 三、临时 SQL 校验脚本（对齐看板数据）

用于快速校验“看板展示值”与数据库当前数据是否一致（建议与页面刷新在同一时刻执行）。

### 1) 明细校验（逐票档行对齐）

目的：获得与前端表格对应的字段集合

SELECT  
 t.id AS ticket\_id,  
 t.name AS ticket\_name,  
 t.price,  
 s.total\_stock,  
 (s.total\_stock - s.locked\_stock - s.sold\_stock) AS available\_stock,  
 s.locked\_stock,  
 s.sold\_stock,  
 s.version,  
 s.update\_time  
FROM t\_ticket t  
LEFT JOIN t\_ticket\_stock s ON s.ticket\_id = t.id  
ORDER BY t.price ASC, t.id ASC;

对照点： - 前端“总库存”= total\_stock - “可用库存”= available\_stock - “锁定库存”= locked\_stock - “已售库存”= sold\_stock

### 2) 汇总校验（对齐页头四个 KPI）

目的：校验总计是否与看板头部统计一致

SELECT  
 COUNT(\*) AS total\_tickets,  
 COALESCE(SUM(s.total\_stock), 0) AS sum\_total\_stock,  
 COALESCE(SUM(s.total\_stock - s.locked\_stock - s.sold\_stock), 0) AS sum\_available\_stock,  
 COALESCE(SUM(s.locked\_stock), 0) AS sum\_locked\_stock,  
 COALESCE(SUM(s.sold\_stock), 0) AS sum\_sold\_stock  
FROM t\_ticket t  
LEFT JOIN t\_ticket\_stock s ON s.ticket\_id = t.id;

对照点： - 座位档数 -> total\_tickets - 总库存 -> sum\_total\_stock - 可用库存 -> sum\_available\_stock - 锁定库存 -> sum\_locked\_stock - 已售库存（用于图表） -> sum\_sold\_stock

### 3) 单票档快速核查（按前端选中的票档ID）

SELECT  
 t.id AS ticket\_id,  
 t.name,  
 t.price,  
 s.total\_stock,  
 (s.total\_stock - s.locked\_stock - s.sold\_stock) AS available\_stock,  
 s.locked\_stock,  
 s.sold\_stock,  
 s.version  
FROM t\_ticket t  
LEFT JOIN t\_ticket\_stock s ON s.ticket\_id = t.id  
WHERE t.id = :ticketId;

## 补充说明

* 前端自动刷新每 10 秒一次（StockDashboard.vue 中 setInterval(fetchStockData, 10000)），核对时建议在同一时刻刷新页面并执行 SQL，避免“未刷新”导致的轻微出入。
* “初始化库存”按钮调用 POST /api/ticket/stock/init，后端执行 INSERT ... ON DUPLICATE KEY UPDATE 写入 t\_ticket\_stock；前端随后延时 500ms 再刷新一次以等待事务提交。

## 便捷扩展（可选）

* 可在后端临时新增一个校验接口，直接返回“汇总校验”SQL 结果，前后端一键对齐。
* 或准备 DataGrip/Workbench 的查询模板，评审时一键执行对比。