# 订单业务设计分析说明

## 1.酒店订单需求分析

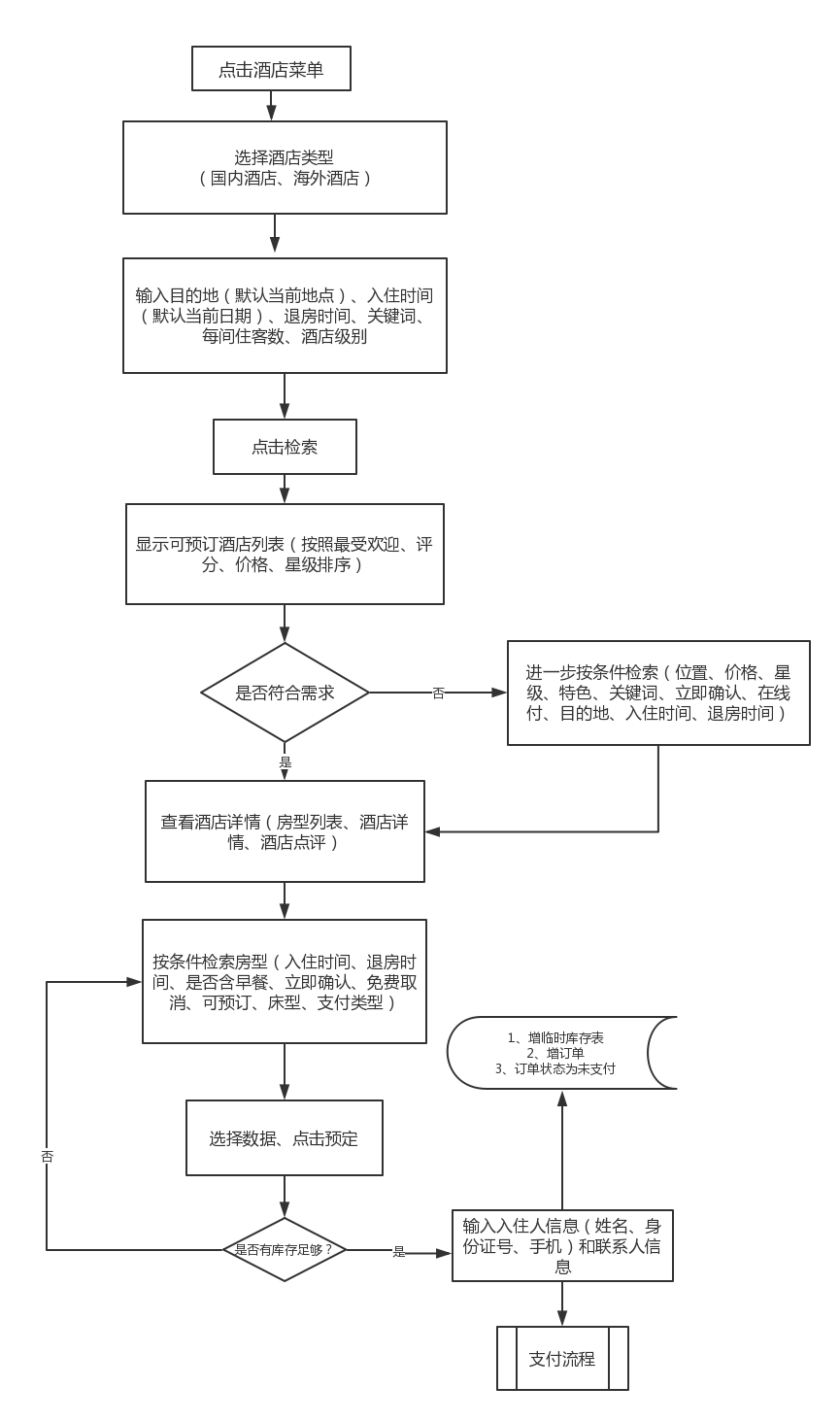
### 1.1酒店的订单操作

* 下订单前库存判断
* 下单成功，暂锁库存
* 订单超时未支付，退回库存
* 订单支付成功，扣减库存
* 订单支付成功，并超过退房日期，修改订单状态

### 1.2细化流程

1. 选择相应酒店
2. 根据酒店选择相应房间
3. 验证房间库存
4. 下单成功，锁定库存
5. 订单支付成功，更新库存数量
6. 超时未支付订单，释放库存
7. 订单支付成功，并超过

### 1.3系统流程图



### 1.4难点分析

* 酒店的库存不同于普通产品的库存，酒店的库存对应的是每一个房间的库存。
* 酒店房间的库存，是一个随着时间变化的库存，每一天对应房间的库存都需要重新计算。

房型A库存=房型A总库存量-房型A当日被预订的数量

* 对房间库存的判断，需根据用户输入的入住和退房日期，以及房型ID，根据以上的算法去计算库存。
* 用户输入的是一个大于等于1天的时间段，那么库存判断需要依据，这个时间段内，库存的最小量来进行计算。

## 2.订单数据库设计

### 2.1酒店表

酒店表：用于保存酒店信息

|  |  |  |
| --- | --- | --- |
| 列 | 类型 | 说明 |
| id | bigint | Auto Increment 主键 |
| hotelName | varchar | 酒店名称 |
| countryId | bigint | 酒店所在城市 |
| provinceId | bigint | 酒店所在省份 |
| cityId | bigint | 酒店所在城市 |
| address | varchar | 酒店地址 |
| details | text | 酒店详细信息 |
| facilities | text | 酒店设施 |
| hotelPolicy | text | 酒店政策 |
| hotelType | int | 酒店类型 |
| hotelLevel | int | 酒店级别 |
| isGroupPurchase | int | 是否支持团购 |
| creationDate | datetime | 创建时间 |
| createdBy | bigint | 创建人 |
| modifyDate | datetime | 修改时间 |
| modifiedBy | bigint | 修改人 |

### 2.2房型表

房型表：用于保存酒店房型的数据库表

|  |  |  |
| --- | --- | --- |
| 列 | 类型 | 说明 |
| id | bigint | Auto Increment 主键 |
| hotelId | bigint | 酒店ID |
| roomTitle | varchar | 房间标题 |
| roomPrice | decimal | 房间价格 |
| roomBedTypeId | bigint | 房间床型 |
| isHavingBreakfast | int | 是否有早餐 |
| payType | int | 支付类型(1:在线付 2:到店付 3:不限) |
| satisfaction | decimal | 满意度 |
| isBook | int | 是否预订 |
| isCancel | int | 是否取消 |
| isTimelyResponse | int | 是否及时确认 |
| creationDate | datetime | 创建时间 |
| createdBy | bigint | 创建人 |
| modifyDate | datetime | 修改时间 |
| modifiedBy | bigint | 修改人 |

### 2.3原始库存表

原始库存表用于保存，房型的原始库存

|  |  |  |
| --- | --- | --- |
| 列 | 类型 | 说明 |
| id | bigint | Auto Increment |
| productType | int | 商品类型(0:旅游产品 1:酒店产品 2:机票产品) |
| productId | bigint | 商品id |
| store | int | 商品库存 |

### 2.4实时库存表

实时库存表用于保存，房型对应的每一天的实时库存，如果用户输入的日期，在实时库存表内没有记录，则需要根据原始库存表往实时库存表插入记录。

|  |  |  |
| --- | --- | --- |
| 列 | 类型 | 说明 |
| id | bigint | Auto Increment、主键 |
| hotelId | int | 酒店ID |
| roomId | bigint | 房型ID |
| recordDate | datetime | 记录日期 |
| store | int | 库存数 |
| creationDate | datetime | 创建时间 |
| createdBy | bigint | 创建人 |

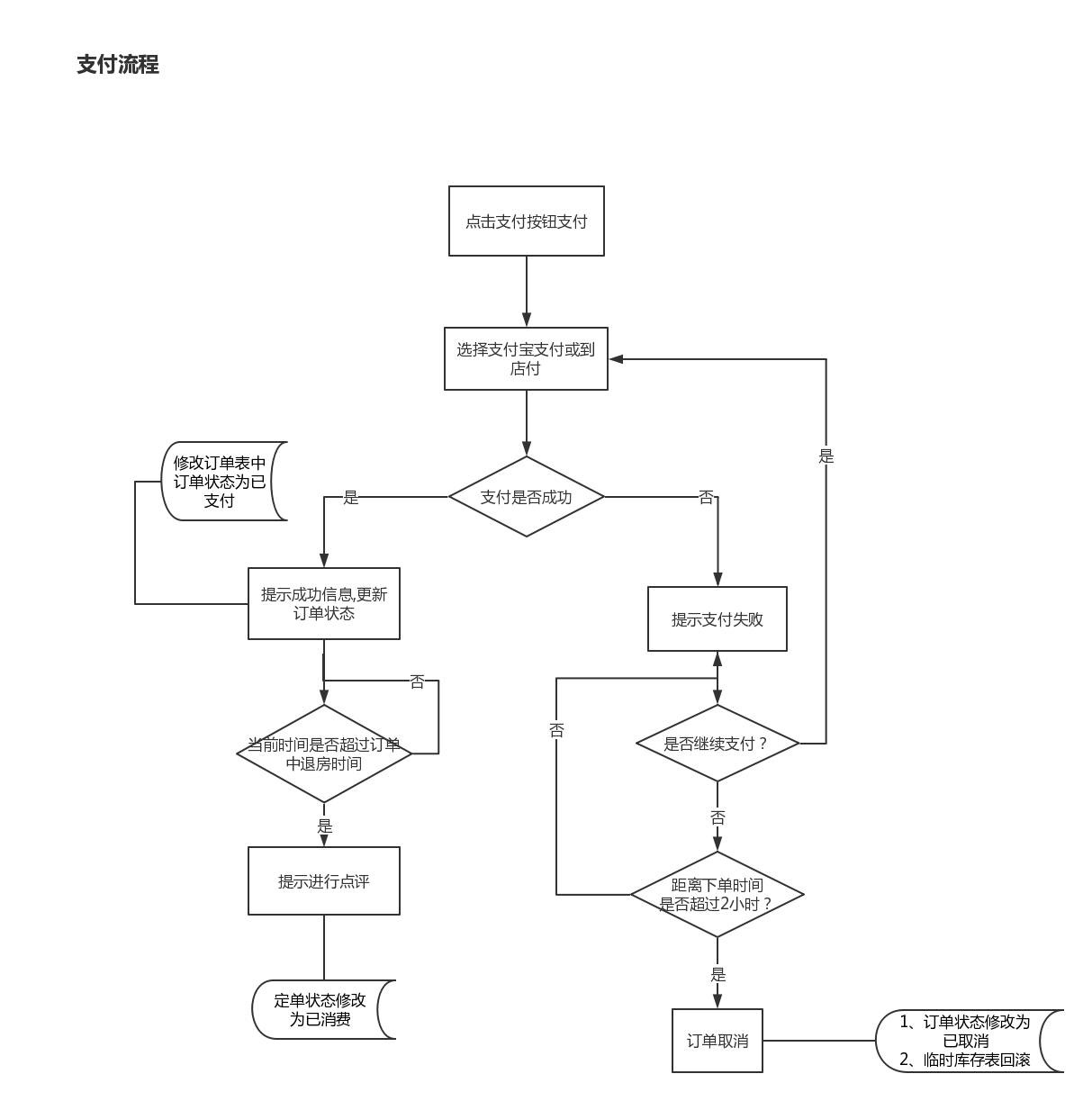
### 2.5订单表

|  |  |  |
| --- | --- | --- |
| 列 | 类型 | 说明 |
| id | bigint | Auto Increment、主键 |
| userId | bigint | 用户ID |
| orderType | int | 订单类型(0:旅游产品 1:酒店产品 2：机票产品) |
| orderNo | varchar | 订单编号 |
| tradeNo | varchar | 交易编号 |
| hotelId | bigint | 酒店ID |
| hotelName | varchar | 酒店名称 |
| roomId | bigint | 酒店房间ID |
| count | int | 预订房间数量 |
| bookingDays | int | 预订天数 |
| checkInDate | datetime | 入住日期 |
| checkOutDate | datetime | 退房日期 |
| orderStatus | int | 订单状态（0：待支付 1:已取消 2:支付成功 3:已消费） |
| payAmount | decimal | 支付金额 |
| payType | int | 支付类型（支付方式:1:支付宝 2:微信 3:到店付） |
| noticePhone | varchar | 联系电话 |
| noticeEmail | varchar | 联系邮件 |
| specialRequirement | text | 特殊要求 |
| isNeedInvoice | int | 是否需要发票（0：不需要 1：需要） |
| invoiceType | int | 发票类型（0：个人 1：公司） |
| invoiceHead | varchar | 发票抬头 |
| linkUserName | varchar | 联系人姓名 |
| bookType | int | 0:WEB端 1:手机端 2:其他客户端 |
| creationDate | datetime | 创建日期 |
| createdBy | bigint | 创建人 |
| modifyDate | datetime | 修改日期 |
| modifiedBy | bigint | 修改人 |

注：因为本次爱旅行项目只做酒店相关订单，故在订单表中保存了hotelId、hotelName等酒店信息。如果后需要做其它产品，则订单表需要进行进一步拆分。

## 3.库存算法

### 3.1支付流程图



### 3.2算法逻辑分析

输入：入住日期、退房日期、房型ID、订房数量

* 根据入住日期、退房日期、房型ID在实时库存表查询对应房型的在这个时间段内的库存是否都存在记录。
* 如果都存在记录，则进行最小库存筛选，如果最小库存小于订房数量，则库存满足，反之则库存充足。
* 如果不存在对应某一个或几个日期的库存，则在实时库存表插入相应记录，继续执行步骤2。

注意：下单成功后，在未支付前，不对实时库存表做减操作，只在订单表里插入相应的记录。所以当日库存的算法公式为：

当日库存=实时库存表的库存数量-当日订单中对应该房型未支付的订单所占的库存。

## 4.订单编码设计

### 4.1库存检查设计

问题：当用户输入入住和退房日期后，如何判断这个时间段内，实时库存表是否有对应记录？并在没有相应记录的情况下插入记录。

解决方案：

* 使用Java程序编写判断程序：程序实现不难，但是在执行查询和插入操作的时候多次连接数据库，性能低下，但是易于维护。
* 使用存储过程实现：程序实现较Java代码有一定的难度，代码不易维护，但是代码执行效率高。

基于应用优先原则，爱旅行项目采用第二种解决方案。

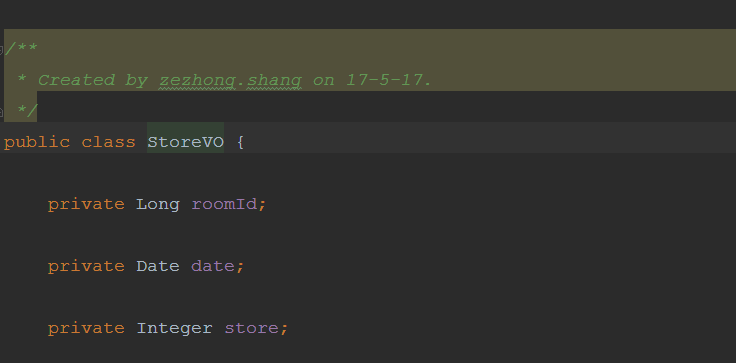
**存储过程代码如下：**

|  |
| --- |
| CREATE DEFINER=`root`@`localhost` PROCEDURE `pre\_flush\_store`(  startTime datetime,  endTime datetime,  roomId1 BIGINT,  hotelId1 BIGINT  )  BEGIN  DECLARE  tempTime datetime;  DECLARE  store1 INT;  DECLARE  count1 INT;  set tempTime=startTime;  WHILE (  date\_format(tempTime, '%Y-%m-%d') <= date\_format(endTime, '%Y-%m-%d')  ) DO  SELECT  COUNT(id) INTO count1  FROM  itrip\_hotel\_temp\_store  WHERE  roomId = roomId1  AND date\_format(recordDate, '%Y-%m-%d') = date\_format(tempTime, '%Y-%m-%d');  IF (count1=0) THEN  SELECT  store INTO store1  FROM  itrip\_product\_store  WHERE  productId = roomId1 AND productType = 1;  INSERT INTO itrip\_hotel\_temp\_store (  hotelId,  roomId,  recordDate,  store,  creationDate  )  VALUES  (  hotelId1,  roomId1,  tempTime,  store1,  NOW()  );  END IF;  set tempTime=date\_add(tempTime, INTERVAL 1 DAY);  END while;  END |

### 4.2返回库存列表设计

当用户输入入住时间、退房时间、房型ID、以及对应数量后，系统首先会检查实时库存表有无对应记录，如果没有，则在实时库存表插入对应的记录。当完成以上操作后，返回相应的库存列表。

1.库存列表对应的接收数据的VO设计（省略GET,SET方法）



2.对应Mapper文件的SQL编写

|  |
| --- |
| SELECT A.roomId,A.recordDate,A.store from ( SELECT  store.roomId,  store.recordDate,  DATE\_FORMAT(store.recordDate,'%Y-%m-%d'),  store.store - (  CASE  WHEN *SUM*(ord.count) IS NULL THEN  0  ELSE  *SUM*(ord.count)  END  ) AS store  FROM  itrip\_hotel\_temp\_store store LEFT JOIN itrip\_hotel\_order ord ON store.roomId = ord.roomId AND ord.orderStatus = 0 AND DATE\_FORMAT(store.recordDate,'%Y-%m-%d') BETWEEN  DATE\_FORMAT(ord.checkInDate, '%Y-%m-%d')  AND  DATE\_FORMAT(ord.checkOutDate,'%Y-%m-%d') WHERE store.roomId = #{roomId} GROUP BY store.roomId,store.recordDate) AS A WHERE A.recordDate BETWEEN  DATE\_FORMAT(#{startTime}, '%Y-%m-%d') AND DATE\_FORMAT(#{endTime}, '%Y-%m-%d')  ORDER by A.store ASC |

### 4.3订单编号的生成规则

爱旅行项目中订单号组成包括：机器码 +日期+（MD5）（商品IDs+毫秒数+1000000的随机数）

* 机器码是为了解决分布式系统的ID可能重复的问题，（比如：在分布式的情况下同时运行两台爱旅行服务器）
* 机器码是自己定义的 1000001或者O1000001 ，O代表Order的意思，都是自己定义的。
* 把机器码写在配置文件中，生成订单的时候取这个机器码的值就行了

那么：第一台服务器里配置：sysConfig.machineCode=D1000001

第二台服务器里配置：sysConfig.machineCode=D1000002

以此类推

1、日期：20170516142638 = 2017-05-16 14:26:38

2、MD5加密括号里面的内容

### 4.4定时程序实现未支付订单

对于超时（下单两小时）未支付的订单，系统需要对其进行直接取消的操作。爱旅行项目中使用定时程序来完成该功能。

程序逻辑：每十分钟扫描一次，查询有没有两个小时以上未支付的订单，若存在则取消该订单。

1.定时器配置

|  |
| --- |
| //spring定时器 订单超时未支付刷单  @Scheduled(cron = "\*/600 \* \* \* \* ?")  public void flushCancelOrderStatus() {  try {  boolean flag = itripHotelOrderService.flushOrderStatus(1);  } catch (Exception e) {  e.printStackTrace();  } }  //spring定时器 订单支付成功刷单  @Scheduled(cron = "\*/6000 \* \* \* \* ?") public void flushOrderStatus() {  try {  logger.info("刷单程序开始执行.......");  boolean flag = itripHotelOrderService.flushOrderStatus(2);  logger.info("刷单程序执行完毕.......");  } catch (Exception e) {  e.printStackTrace();  } } |

2.Mapper文件

|  |
| --- |
| <update id="flushCancelOrderStatus">  update itrip\_hotel\_order set orderStatus=1 where orderStatus=0 and timestampdiff(SECOND, creationDate, NOW()) >= 2 \* 3600 </update>  <update id="flushSuccessOrderStatus">  update itrip\_hotel\_order set orderStatus=3 where orderStatus=2 and timestampdiff(SECOND, checkOutDate, NOW()) >= 0 </update> |

## 5.订单功能测试

### 5.1测试目的

测试爱旅行项目对订单库存的操作是否正确。

* 无房认定：所定的房间在入住日期和退房日期之间所有的日期内，房间的数量小于预订数量的，则认为房间数量不足，则给出相应的无房提示。
* 有房认定：所定的房间在入住日期和退房日期之间所有的日期内，房间的数量都大于预订数量的，则认为房间数量满足预订条件。

### 5.2测试用例

#### A测试用例

输入条件：

1. 临时库存表有入住日期到退房日期内的全部日期的房间库存信息
2. 库存数量充足

测试步骤：

1. 用户在未登陆状态下请求接口（输出提示用户未登录）
2. 用户请求房间数量验证接口(输出验证房间信息通过)
3. 用户下单订单（输出预订成功）

#### B.测试用例

输入条件：

1. 临时库存表缺少部分入住日期到退房日期内的库存信息
2. 库存数量充足

测试步骤：

1. 用户请求房间数量验证接口(输出验证房间信息通过，临时库存表新增相应记录)
2. 用户下单订单（输出预订成功，临时库存表新增相应记录）

#### C.测试用例

1. 临时库存表缺少全部入住日期到退房日期内的库存信息
2. 库存数量充足

测试步骤：

1. 用户请求房间数量验证接口(输出验证房间信息通过，临时库存表新增相应记录)
2. 用户下单订单（输出预订成功，临时库存表新增相应记录）

#### D.测试用例

1. 临时库存表中库存数量小于预订数量

测试步骤：

1. 用户请求房间数量验证接口(输出验证房间信息不通过)
2. 用户下单失败（输出预订失败）

#### E.测试用例

1. 临时库存表不存在对应的记录，且商品库存表中的库存不足

测试步骤：

1. 用户请求房间数量验证接口(输出验证房间信息不通过)
2. 用户下单失败（输出预订失败）

#### F.测试用例

1. 临时库存表库存数量充足
2. 临时库存数量减去待支付状态订单占用的房间数量，小于预订的房间数量

测试步骤：

1. 用户请求房间数量验证接口(输出验证房间信息不通过)
2. 用户下单失败（输出预订失败）
3. 两小时后，用户再次执行下单操作下单成功

### 5.3测试结果

#### A测试用例

输入1：

|  |
| --- |
| {  "checkInDate": "2017-01-01 12:00:00",  "checkOutDate": "2017-01-03 12:00:00",  "count": 20,  "hotelId": 1,  "roomId": 1  } |

输入2

|  |
| --- |
| {  "checkInDate": "2017-01-01",  "checkOutDate": "2017-01-03",  "count": 1,  "hotelId": 1,  "hotelName": "北京首都大酒店",  "invoiceHead": "发票抬头",  "invoiceType": 1,  "isNeedInvoice": 1,  "linkUserIds": "1,2",  "linkUserName": "张三，李四",  "noticeEmail": "sdodh366@163.com",  "noticePhone": "18012345555",  "orderType": 0,  "roomId": 1,  "specialRequirement": "no"  } |

结果1

|  |
| --- |
| {  "data": null,  "errorCode": "100507",  "msg": "库存不足",  "success": "false"  } |

结论1：程序正常

结果2：

|  |
| --- |
| {  "data": null,  "errorCode": "0",  "msg": "生成订单成功",  "success": "true"  } |

结论2：程序正常

#### B.测试用例

输入1：

|  |
| --- |
| {  "checkInDate": "2000-01-01 12:00:00",  "checkOutDate": "2000-01-03 12:00:00",  "count": 20,  "hotelId": 1,  "roomId": 1  } |

输入2：

|  |
| --- |
| {  "checkInDate": "2000-01-06 12:00:00",  "checkOutDate": "2000-01-08 12:00:00",  "count": 20,  "hotelId": 1,  "roomId": 1  } |

输入3：

|  |
| --- |
| {  "checkInDate": "2000-01-03 12:00:00",  "checkOutDate": "2000-01-06 12:00:00",  "count": 20,  "hotelId": 1,  "roomId": 1  } |

1.先用前两种数据进行测试房间数量验证接口，生成两个时间段的临时库存表的数据

2.使用第三种数据进行测试房间数量验证接口，看一下能否在这两个时间段中间的日期段添加数据

结果均为：

|  |
| --- |
| {  "data": {  "flag": true  },  "errorCode": "0",  "msg": "操作成功",  "success": "true"  } |

结论：正确

输入4：

|  |
| --- |
| {  "checkInDate": "2000-02-01",  "checkOutDate": "2000-02-03",  "count": 16,  "hotelId": 1,  "hotelName": "北京首都大酒店",  "invoiceHead": "发票抬头",  "invoiceType": 1,  "isNeedInvoice": 1,  "linkUserIds": "1,2",  "linkUserName": "张三，李四",  "noticeEmail": "sdodh366@163.com",  "noticePhone": "18012345555",  "orderType": 0,  "roomId": 1,  "specialRequirement": "no"  } |

输入5：

|  |
| --- |
| {  "checkInDate": "2000-02-06",  "checkOutDate": "2000-02-08",  "count": 16,  "hotelId": 1,  "hotelName": "北京首都大酒店",  "invoiceHead": "发票抬头",  "invoiceType": 1,  "isNeedInvoice": 1,  "linkUserIds": "1,2",  "linkUserName": "张三，李四",  "noticeEmail": "sdodh366@163.com",  "noticePhone": "18012345555",  "orderType": 0,  "roomId": 1,  "specialRequirement": "no"  } |

输入6:

|  |
| --- |
| {  "checkInDate": "2000-02-03",  "checkOutDate": "2000-02-06",  "count": 16,  "hotelId": 1,  "hotelName": "北京首都大酒店",  "invoiceHead": "发票抬头",  "invoiceType": 1,  "isNeedInvoice": 1,  "linkUserIds": "1,2",  "linkUserName": "张三，李四",  "noticeEmail": "sdodh366@163.com",  "noticePhone": "18012345555",  "orderType": 0,  "roomId": 1,  "specialRequirement": "no"  } |

1.先用前两种数据进行请求生成订单接口，生成两个时间段的临时库存表的数据

2.使用第三种数据进行请求生成订单接口，看一下能否在这两个时间段中间的日期段添加数据

输出结果：

|  |
| --- |
| {  "data": {  "orderNo": "D1000001201706011054246a81e1",  "id": "83"  },  "errorCode": "0",  "msg": "生成订单成功",  "success": "true"  } |

结论：正确

#### c测试用例

输入1：

|  |
| --- |
| {  "checkInDate": "2000-03-01 12:00:00",  "checkOutDate": "2000-03-03 12:00:00",  "count": 20,  "hotelId": 1,  "roomId": 1  } |

输入2：

|  |
| --- |
| {  "checkInDate": "2000-03-01",  "checkOutDate": "2000-03-03",  "count": 1,  "hotelId": 1,  "hotelName": "北京首都大酒店",  "invoiceHead": "发票抬头",  "invoiceType": 1,  "isNeedInvoice": 1,  "linkUserIds": "1,2",  "linkUserName": "张三，李四",  "noticeEmail": "sdodh366@163.com",  "noticePhone": "18012345555",  "orderType": 0,  "roomId": 1,  "specialRequirement": "no"  } |

1.先确保数据库的临时库存表中没有对应第一种数据的数据库记录，

用第一种数据请求试房间数量验证接口，看一下数据库中能否生成所需的数据

请求结果为：

|  |
| --- |
| {  "data": {  "flag": true  },  "errorCode": "0",  "msg": "操作成功",  "success": "true"  } |

结论：正确

2.先确保数据库的临时库存表中没有对应第二种数据的数据库记录

用第二种数据请求订单生成接口，看一下数据库中能否生成所需的数据

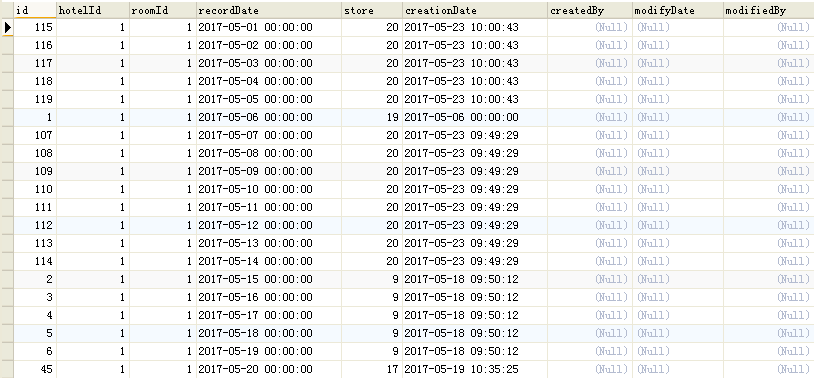
结果：

|  |
| --- |
| {  "data": {  "orderNo": "D100000120170601105902470967",  "id": "84"  },  "errorCode": "0",  "msg": "生成订单成功",  "success": "true"  } |

结论：正确

#### D 测试用例测试

数据库初始态:



输入1:请求验证房间库存接口

|  |
| --- |
| {  "checkInDate": "2017-05-01 00:00:00",  "checkOutDate": "2017-05-20 00:00:00",  "count": 10,  "hotelId": 1,  "roomId": 1} |

结果1:

|  |
| --- |
| {  "data": {  "flag": false  },  "errorCode": "0",  "msg": "操作成功",  "success": "true"  } |

结论：程序正常

输入2：请求验证房间库存接口

|  |
| --- |
| {  "checkInDate": "2017-05-01 00:00:00",  "checkOutDate": "2017-05-20 00:00:00",  "count": 9  "hotelId": 1,  "roomId": 1} |

结果2：

|  |
| --- |
| {  "data": {  "flag": true  },  "errorCode": "0",  "msg": "操作成功",  "success": "true"  } |

结论：正常

输入3:请求下单接口:

|  |
| --- |
| {  "checkInDate": "2017-05-01",  "checkOutDate": "2017-05-20",  "count": 10,  "hotelId": 1,  "hotelName": "北京首都大酒店",  "invoiceHead": "北大青鸟阿博泰克",  "invoiceType": 1,  "isNeedInvoice": 1,  "linkUserName": "周东成",  "noticeEmail": "1044755845@qq.com",  "noticePhone": "13366985584",  "orderType": 1,  "roomId": 1,  "specialRequirement": "暂无"  } |

结果3：

|  |
| --- |
| {  "data": null,  "errorCode": "100507",  "msg": "库存不足",  "success": "false"  } |

输入4:请求下单接口

|  |
| --- |
| {  "checkInDate": "2017-05-01",  "checkOutDate": "2017-05-20",  "count": 9  "hotelId": 1,  "hotelName": "北京首都大酒店",  "invoiceHead": "北大青鸟阿博泰克",  "invoiceType": 1,  "isNeedInvoice": 1,  "linkUserName": "周东成",  "noticeEmail": "1044755845@qq.com",  "noticePhone": "13366985584",  "orderType": 1,  "roomId": 1,  "specialRequirement": "暂无"  } |

结果4:

|  |
| --- |
| {  "data": null,  "errorCode": "0,  "msg": "下单成功  "success": "false"  } |

结论：正常

#### E.测试用例

目前针对酒店id为1的房间id为1的房间，临时库存表不存在相关库存数据，商品库存表中库存为20.

输入1:请求验证库存接口

|  |
| --- |
| {  "checkInDate": "2017-06-01 00:00:00",  "checkOutDate": "2017-06-10 00:00:00",  "count": 30,  "hotelId": 1,  "roomId": 1  } |

结果1: 请求验证库存接口

|  |
| --- |
| {  "data": {  "flag": false  },  "errorCode": "0",  "msg": "操作成功",  "success": "true"  } |

结论：正常

输入2: 请求验证库存接口

|  |
| --- |
| {  "checkInDate": "2017-06-01 00:00:00",  "checkOutDate": "2017-06-10 00:00:00",  "count": 5  "hotelId": 1,  "roomId": 1  } |

结果2：

|  |
| --- |
| {  "data": {  "flag": true  },  "errorCode": "0",  "msg": "操作成功",  "success": "true"  } |

输入3:请求下单接口

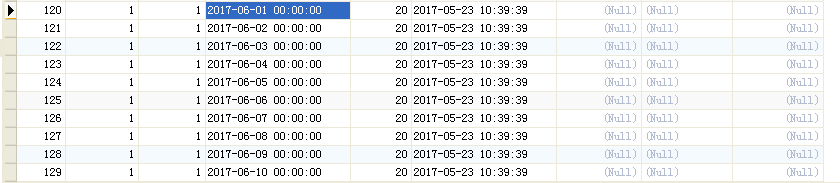
|  |
| --- |
| {  "checkInDate": "2017-06-01",  "checkOutDate": "2017-06-10",  "count": 10,  "hotelId": 1,  "hotelName": "北京首都大酒店",  "invoiceHead": "北大青鸟",  "invoiceType": 1,  "isNeedInvoice": 1,  "linkUserIds":1,  "linkUserName": "周翔",  "noticeEmail": "10447@qq.com",  "noticePhone": "13555845878",  "orderType": 1,  "roomId": 1,  "specialRequirement": "无"  } |

结果3：

|  |
| --- |
| {  "data": null,  "errorCode": "0",  "msg": "生成订单成功",  "success": "true"  } |

结论：结果正常

#### F.测试用例



输入1:

|  |
| --- |
| {  "checkInDate": "2017-06-01 00:00:00",  "checkOutDate": "2017-06-10 00:00:00",  "count": 15,  "hotelId": 1,  "roomId": 1  } |

结果1:

|  |
| --- |
| {  "data": {  "flag": false  },  "errorCode": "0",  "msg": "操作成功",  "success": "true"  } |

结论：正常