第三周课设报告

本周小结

本周我进一步完成了订单的业务逻辑设计，订单是本项目实现逻辑最为复杂的部分，因为它关联了商品、购物车、折扣中多项属性的改变，需要考虑到各种情况的组合。同时，本周配置好了开源组件库ElaWidgetTools的使用环境，ElaWidgetTools在Qt原版的控件基础上进行简单的继承封装，在不破坏原有控件灵活性的基础上提供了更加精致美观的控件以供开发者使用，有利于客户与前端页面进行更为流畅舒适的交互操作。

数据表设计

- 订单（t_order）

```
CREATE TABLE t_order (
  oid INT AUTO_INCREMENT COMMENT '订单id',
  uid INT NOT NULL COMMENT '用户id',
  receive_name VARCHAR(20) COMMENT '收货人姓名',
  receive_address VARCHAR(60) COMMENT '收货人详细地址',
```

```
  receive_phone_number VARCHAR(20) COMMENT '收货人手机号',
  total_price DECIMAL(10, 2) COMMENT '总价',
  state INT COMMENT '状态 0-未支付 1-已支付 2-已取消 3-已完成 4-已退款',
  order_time DATETIME COMMENT '下单时间',
  pay_time DATETIME COMMENT '支付时间',
  created_user VARCHAR(20) COMMENT '创建者',
  created_time DATETIME COMMENT '创建时间',
  modified_user VARCHAR(20) COMMENT '最后修改者',
  modified_time DATETIME COMMENT '最后修改时间',
  PRIMARY KEY(oid)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

创建订单

持久层

需要执行的SQL语句

根据uid获取用户的所有信息：已实现

根据aid获取地址的所有信息：已实现

根据uid获取购物车中对应商品的所有信息：已实现

根据pid获取对应商品的所有信息：已实现

根据pid获取对应商品的折扣信息：已实现

获取所有生效的按达到金额要求降序排列的满减策略：

```
SELECT * FROM t_strategy WHERE is_valid = 1 ORDER BY target DESC;
```

向t_order中插入一条订单数据：

```
INSERT INTO t_order (*) VALUES (*);
```

根据uid查询所有订单信息：

```
SELECT * FROM t_order WHERE uid = ? ORDER BY created_time DESC;
```

向t_order_item中插入一条订单物品数据：

```
INSERT INTO t_order_item (*) VALUES (*);
```

根据uid和pid删除对应购物车数据：已实现 根据pid更新num和priority：

```
UPDATE t_product SET num = num + ?, priority = priority + ? WHERE pid = ?;
```

接口和抽象方法

- StrategyMapper类设计
  - 新增接口
    - QList<Strategy> getValidInfo();
      - 获取有效的满减策略的所有信息
- OrderMapper类设计
  - 新增接口
    - int insert(const Order& order) ;
      - 插入一条订单数据的所有信息
    - QList<Order> findByUid(int uid);
      - 根据uid获取对应订单的所有信息
      - 结果按创建时间降序排序
- OrderItemMapper类设计
  - 新增接口
    - int insert(const OrderItem& order_item);
      - 插入一条订单物品数据的所有信息
- ProductMapper类设计
  - 新增接口
    - int updateNum(int pid, int num);
      - 根据pid和num更新相应商品数目和优先级（num可为负）

业务层

异常规划

- UserNotFoundError：根据uid找不到用户
- AddressNotFoundError：根据aid找不到地址
- AccessDeniedError：aid和uid不匹配
- NoProductInCartError：购物车内没有商品
- ProductNotEnoughError：下单的商品超出库存
- InsertError：插入订单/订单物品数据过程发生未知异常
- RemoveError：删除购物车数据的过程发生未知异常
- UpdateError：修改商品数据的过程发生未知异常

接口和抽象方法

- OrderService类
  - 新增接口
    - int createOrder(int uid, int aid);
      - 根据uid和aid创建订单
      - 返回创建好的订单的oid

完成订单支付

持久层

需要执行的SQL语句

根据oid获取对应订单的所有信息：

```
SELECT * FROM t_order WHERE oid = ?;
```

根据oid修改state属性：

```
UPDATE t_order SET state = ?,
modified_user = ?, modified_time = ? WHERE
oid = ?;
```

根据oid修改pay_time属性：

```
UPDATE t_order SET pay_time = ? WHERE oid
= ?;
```

接口和抽象方法

- OrderMapper类设计
  - 新增接口
    - Order findByOid(int oid);
      - 根据oid获取对应订单的所有信息
    - int updateState(int oid, int state, const QString& modified_user, const QDateTime& modified_time);
      - 根据oid修改state属性
      - 返回影响的行数

        - int updatePayTime(int oid, const QDateTime& pay_time);
            - 根据oid修改pay_time属性
            - 返回影响的行数

业务层

异常规划

- OrderNotFoundError：找不到可操作的订单
- AccessDeniedError：uid和oid不匹配
- UpdateError：更新订单数据过程发生未知异常

接口和抽象方法

- OrderService类
    - 新增接口
        - void payOrder(int uid, int oid, const QString& modified_user);
            - 根据uid和oid完成相应订单支付
            - modified_user为实际操作者

取消订单

持久层

需要执行的SQL语句

根据oid获取对应订单的所有信息：已实现 根据oid修改state属性：已实现 根据oid获取所有相应订单商品的信息：

```
SELECT * FROM t_order_item WHERE oid = ?;
```

根据pid更新num和priority：已实现

接口和抽象方法

- OrderItemMapper类设计
  - 新增接口
    - QList<OrderItem> findByOid(int oid);
      - 根据oid查询对应的订单物品的所有信息

业务层

异常规划

- OrderNotFoundError：找不到可操作的订单
- AccessDeniedError：uid和oid不匹配
- UpdateError：更新订单数据/商品数据过程发生未知异常

接口和抽象方法

- OrderService类
  - 新增接口

- void cancelOrder(int uid, int oid, const QString& modified_user);
  - 根据uid和oid取消相应的订单
  - modified_user为实际操作者

订单退款

持久层

需要执行的SQL语句

根据oid获取对应订单的所有信息：已实现 根据oid修改state属性：已实现 根据oid获取所有相应订单商品的信息：已实现 根据pid更新num和priority：已实现

接口和抽象方法

均已实现，无新增。

业务层

异常规划

- OrderNotFoundError：找不到可操作的订单
- AccessDeniedError：uid和oid不匹配
- UpdateError：更新订单数据/商品数据过程发生未知异常

接口和抽象方法

- OrderService类
  - 新增接口
    - void RefundOrder(int uid, int oid, const QString& modified_user);
      - 根据uid和oid对订单进行退款
      - modified_user为实际操作者

获取订单信息

持久层

需要执行的SQL语句

根据uid获取对应订单的所有信息：已实现

接口和抽象方法

均已实现，无新增。

业务层

异常规划

无异常

接口和抽象方法

- OrderService类

- 新增接口
  - QList<Order> getOrderInfo(int uid);
    - 根据uid获取所有订单的详细信息

获取订单商品详细信息

需要执行的SQL语句

根据oid获取对应订单商品的所有信息：已实现

接口和抽象方法

均已实现，无新增。

业务层

异常规划

无异常

接口和抽象方法

- OrderService类
  - 新增接口
    - QList<OrderItem> getOrderMoreInfo(int oid);
      - 根据oid获取所有订单商品的详细信息