大型超市数据库开发文档

21307071 詹迪　　21307077 凌国明

第一章 需求建模

1.1 系统调查

现如今，超大型超市营业模式盛行，如沃尔玛、山姆会员商店、麦德龙超市等。我们计划为大型超市开发一个数据库管理系统，经过信息收集与系统调查，对系统的功能性需求分析如下：

①商品信息：大型超市数据库管理系统，需要存储商品相关信息，诸如商品名称、商品类型、商品价格、上架数量、摆放位置、过期时间、在售数量等等。这些数据需要被及时更新、备份和维护，并且需要支持多用户同时访问。

②库存管理：大型超市数据库管理系统，需要提供库存管理服务，用于监控商品数量，及时更新商品的在售数量和库存数量，在必要时提醒员工对商品进行补货。

③顾客服务：大型超市数据库管理系统，需要提供完善的顾客服务，包括向顾客提供商品信息、促销活动和折扣优惠；告诉顾客商品的位置，以便导航顾客到相应的位置选购商品；为顾客生成购买订单用于自动结账等等。

④销售统计：大型超市数据库管理系统，还需要满足销售统计的需求，用于监控商品销售情况、顾客消费习惯等信息，生成对应的报表，以便管理者制定更好的销售策略。例如，经理可以查看某一段时间内，某种商品的订单数量、销售额和利润等信息，来决定是否继续进货。

⑤人事管理：大型超市数据库管理系统，也需要提供人事管理服务。经理可以规定员工的上班时间、对员工进行人事任免等。

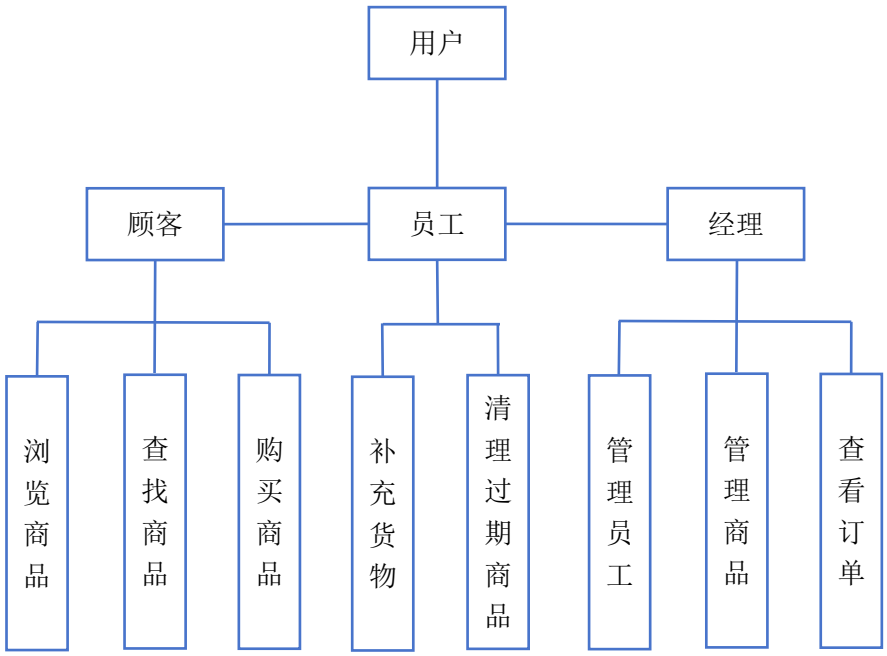
本系统主要面向三种用户角色：顾客、员工、经理。三种角色的具体功能需求各不相同：

顾客：浏览商品、查找商品、购买商品。

员工：补充商品、清理过期商品。

经理：管理员工、管理商品、查看订单。

用户组织架构图如下：



（图 1-1：用户组织架构图）

1.2 系统用例

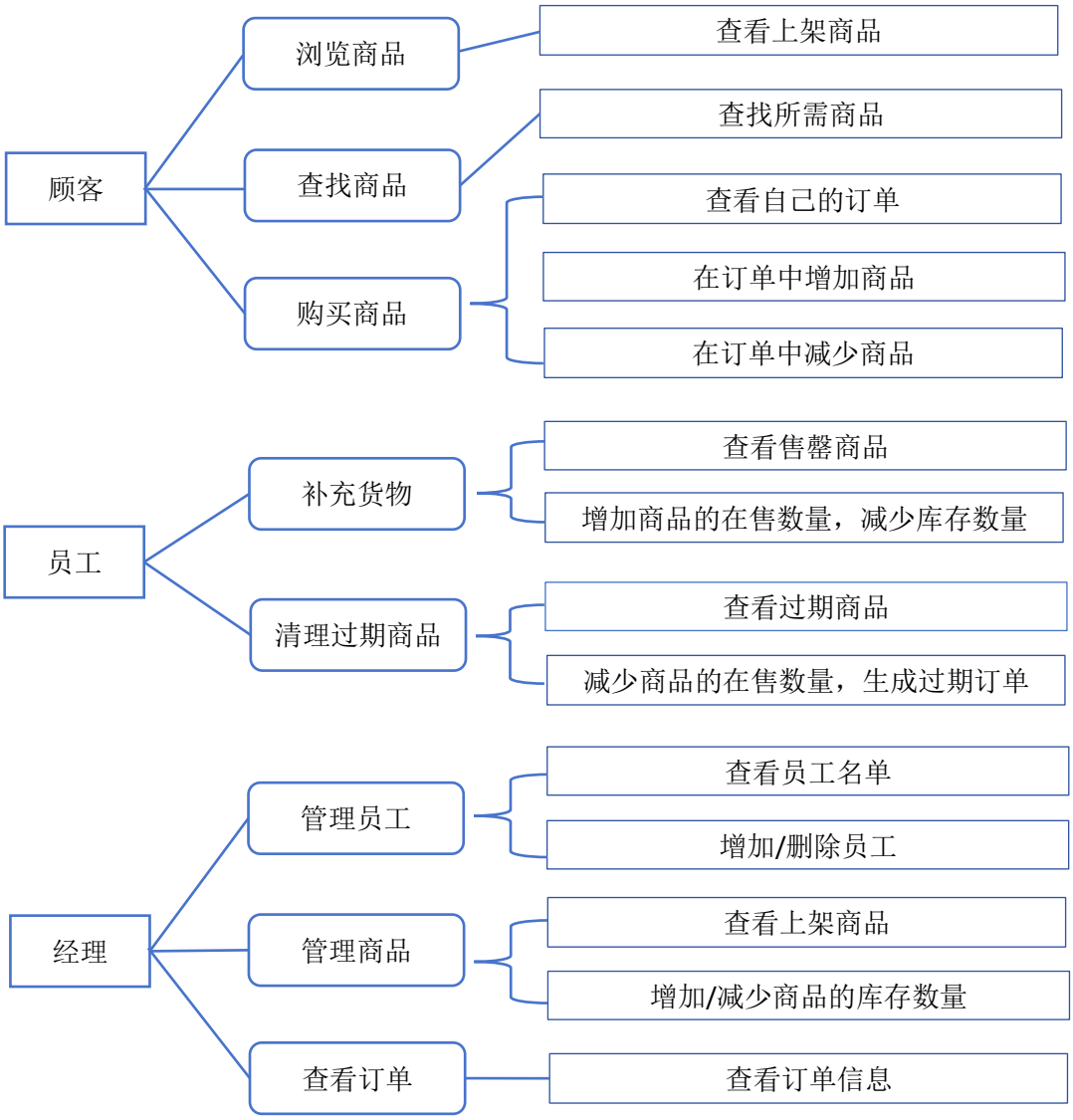
大型超市数据库系统的三种用户：顾客、员工、经理，根据他们各自的功能性需求不同，会产生不同的系统用例。

顾客：查看上架商品、查找所需商品、查看自己的订单、在订单中增加商品（增加订单明细）、在订单中减少商品（减

少订单明细）

员工：查看售罄商品、增加商品在售数量（补货，即根据商品库存余量决定是否增加），查看过期商品、减少商品在售数量（清理过期的商品，同时生成过期订单）

经理：查看员工名单、增加员工、删除员工，修改员工的工作时间；查看商品信息、增加商品、删除商品，修改商品库存量；查看订单信息，进行统计分析。



（图 1-2：系统用例图）

1.3 系统核心用例

本系统涉及到的用例比较多，本节主要介绍以下三个核心用例，分别是：在订单中增加商品、清理过期商品、增加/删除员工。

1.3.1 在订单中增加商品

用例：在订单中增加商品

范围：大型超市数据库管理系统

级别：用户目标

主要参与者：顾客

涉众及其关注点：

1.顾客：希望准确快速地查找到所需的商品，并在订单中增加该商品。

2.商品：以订单明细的形式，快速地被添加到顾客订单中。同时，减少该商品的在售数量。

前置条件：顾客的帐号、相应的商品必须已经被添加到系统中。

成功保证：提示商品增加成功，新加入订单的商品会显示在相应顾客的订单上。

主成功场景：

1.顾客用帐号和密码成功登录系统。

2.顾客查找所需商品的位置等信息。

3.顾客向订单中增加商品。

4.顾客查看自己的订单中是否成功添加相应商品。

扩展：

1a. 顾客所输入的帐号或密码不正确：

系统提示用户帐号或密码错误，要求用户重新输入。

2a. 顾客查找的商品已售罄/不存在：

系统提示所查询的商品已售罄/不存在。

特殊需求：系统的设计保障系统的健壮性。

发生频率：经常发生。

1.3.2 清理过期商品

用例：清理过期商品

范围：大型超市数据库管理系统

级别：用户目标

主要参与者：员工

涉众及其关注点：

1.员工：希望准确快速地查找到已经过期的商品，清理该商品（减少该商品的在售数量），同时创建并提交过期商品订单。

2.商品：希望能够以订单明细的形式，快速地被添加到过期订单中。同时，减少该商品的在售数量。

前置条件：员工的帐号、过期的商品必须已经被记录在系统中。

成功保证：提示过期商品清理成功，加入订单的过期商品会显示在相应过期订单上。

主成功场景：

1.员工用帐号和密码成功登录系统。

2.员工查找过期商品的位置等信息。

3.员工向过期订单中增加过期商品，并减少该商品的在售数量。

4.顾客无法查询到过期的商品

5.经理可以查看过期订单中过期的商品。

扩展：

1. 员工所输入的帐号或密码不正确：

系统提示用户帐号或密码错误，要求用户重新输入。

特殊需求：系统的设计保障系统的健壮性。

发生频率：经常发生。

1.3.3 增加/删除员工

用例：增加/删除员工

范围：大型超市数据库管理系统

级别：用户目标

主要参与者：经理

涉众及其关注点：

1.经理：希望准确快速地增加或者删除员工。

2.员工：准确快速地从员工名单上添加（或者删除）。

前置条件：经理的帐号必须已经被添加到系统中。

成功保证：提示员工增加/删除成功，新加入的员工会显示在相应的员工名单上。

主成功场景：

1.经理用帐号和密码成功登录系统。

2.经理在员工名单中增加/删除员工。

3.经理查看员工订单中是否成功增加/删除相应员工。

扩展：

1. 经理所输入的帐号或密码不正确：

系统提示用户帐号或密码错误，要求用户重新输入。

2. 经理查找的员工不存在：

系统提示所查询的员工不存在。

特殊需求：系统的设计保障系统的健壮性。

发生频率：不经常发生。

1.4 领域模型

大型超市数据库管理系统共有 6 个实体：

顾客(Customer)：顾客 ID，姓名，联系方式，地址

员工(Employee)：员工 ID，姓名，联系方式，职位，工作时间

经理(Manager)：经理 ID，姓名，联系方式，职责

商品(Goods)：商品 ID，名称，进价，售价，过期日期，最大上架数量，货架上数量，摆放位置，库存数量

订单(Order)：订单号，日期，总售价，总利润，是否是过期清理

商品订单明细(OrderDetail)：数量

它们的关系如下：

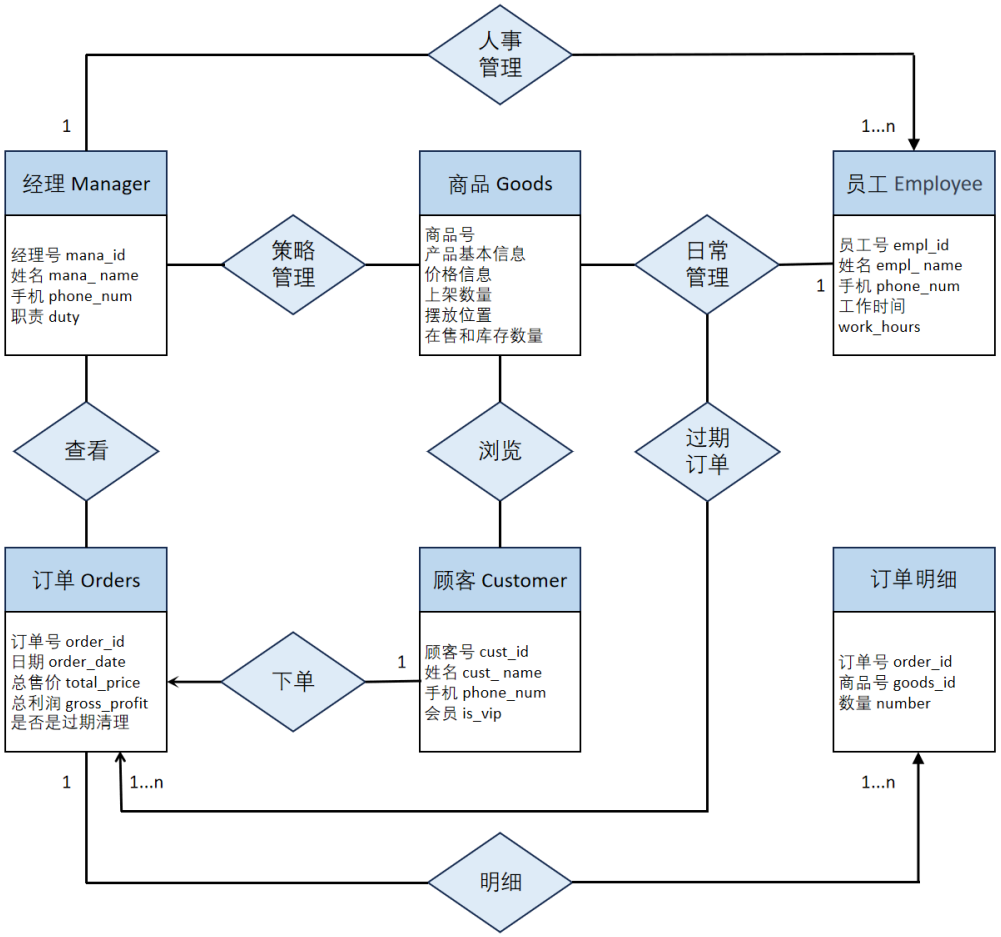
下单(Customer – Orders)：一对多联系，一个顾客可下单多个订单，一个订单对应一个顾客，关系为 1：0...n

明细(Orders – OrderDetail)：一对多联系，一个订单对应多个明细，一个明细对应一个订单，关系为 1：1...n

包含(OrderDetail – Goods)：多对一联系，一个明细对应一个商品，一个商品对应多个明细，关系为 1...n：1

管理(Employee – Orders)：一对多联系，一个订单对应一个员工，一个员工可对应多个订单,关系为 1：1...n

管理(Manager – Empolyee)：一对多关系，一个经理对应多个员工，一个员工对应一个经理，关系为 1：1...n

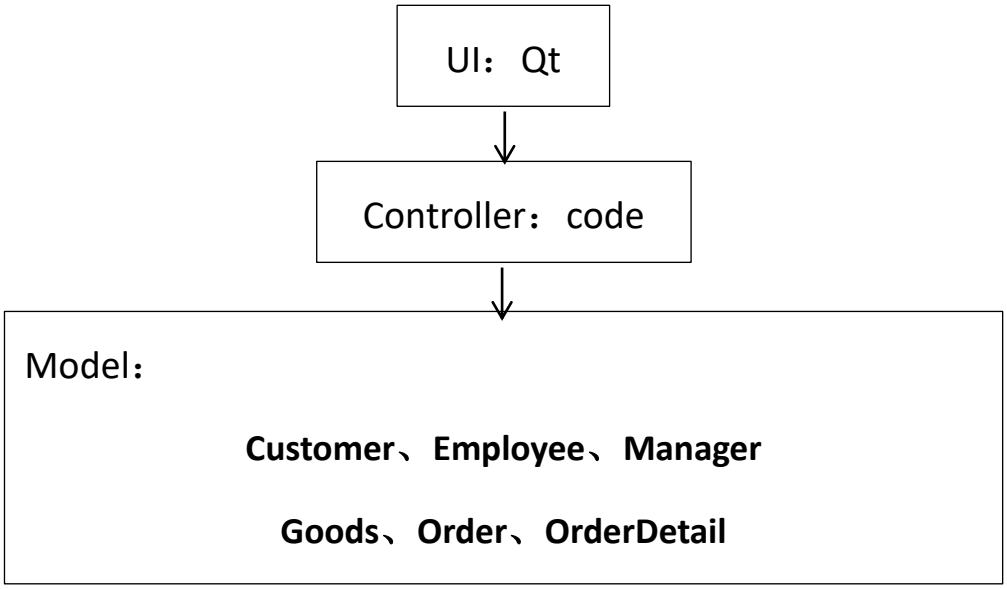


（图 1-3：系统领域模型图）

第二章 架构设计

2.1 系统架构及原理

大型超市数据库管理系统采用经典的 MVC 设计模式，将应用程序分成三个主要部分：模型（Model）、视图（View）和控制器（Controller），以实现应用程序的分层、松耦合的架构。



（图 2-1：系统架构设计图）

UI 层

视图（View）是应用程序中用户界面的显示部分，它们呈现模型中的数据给用户，并提供适当的界面元素和操作。视图通常是被动的，它们只响应用户的输入事件，展示模型数据，并将任何用户交互传递给控制器。

控制层

控制器（Controller）作为模型和视图之间的协调者，处理用户的所有输入事件，调用模型来更新数据并通知视图进行更新。控制器负责管理应用程序的整个运行时逻辑，并协调模型和视图之间的交互。

模型层

模型（Model）是应用程序中数据和业务逻辑的中心。它们表示应用程序中的实体，例如用户、订单、商品等。在 MVC 模式中，模型被设计为完全独立于视图和控制器。模型通过提供 API 接口来与控制器交互，控制器则使用这些接口来管理和更新模型。

2.2 数据库设计

2.2.1 数据库概念结构设计

先识别出系统中的实体。然后根据实际语义，分析实体之间的联系，确定实体之间一对一，一对多和多对多联系。据此，绘制实体-联系图(E-R 图)。

设计一个大型超市管理系统，根据需求调查信息，设计实体如下：

（图 2-2：系统实体图）

顾客(Customer)：顾客 ID，姓名，联系方式，地址

员工(Employee)：员工 ID，姓名，联系方式，职位，工作时间

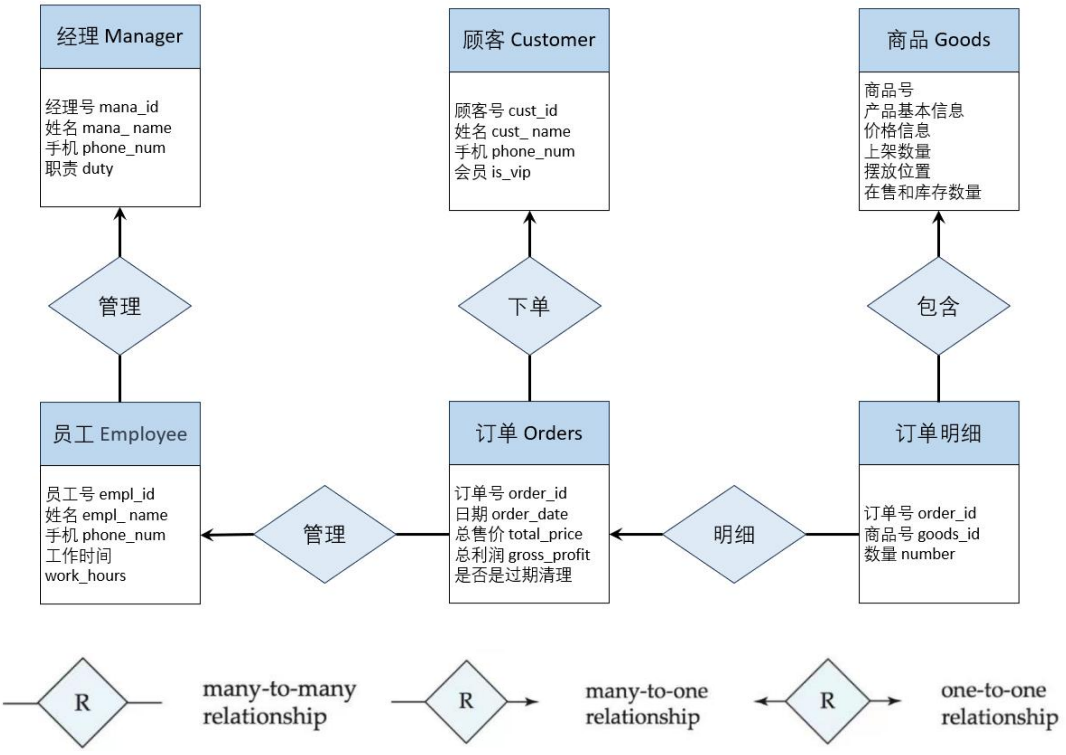
经理(Manager)：经理 ID，姓名，联系方式，职责

商品(Goods)：商品 ID，名称，进价，售价，过期日期，最大上架数量，货架上数量，摆放位置，库存数量

订单(Order)：订单号，日期，总售价，总利润，是否是过期清理

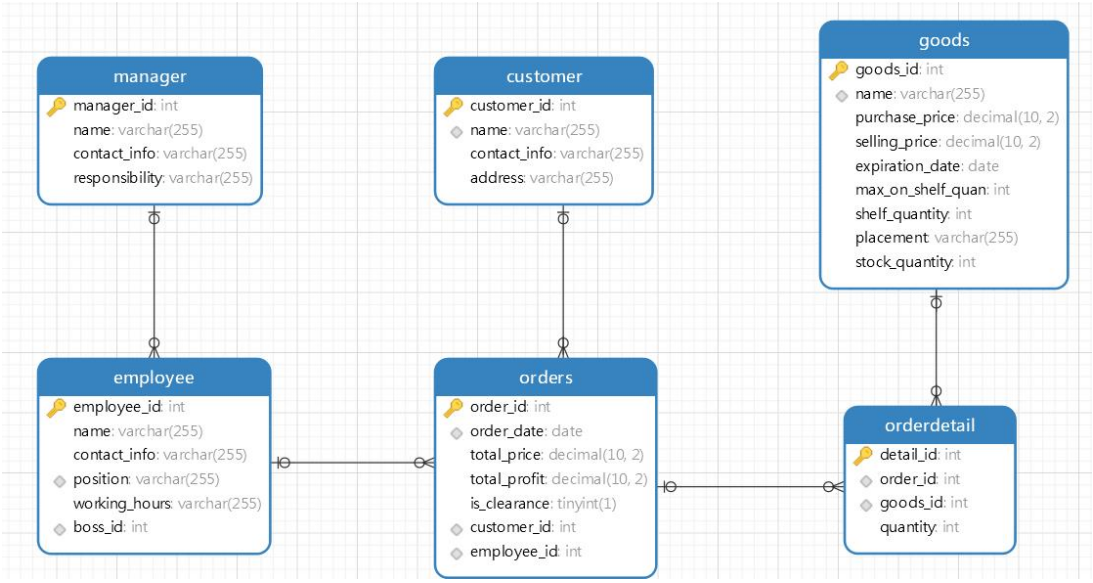
商品订单明细(OrderDetail)：数量

大型超市数据库系统的 E-R 图绘制如下：



（图 2-3：系统 E-R 图）

下单(Customer – Orders)：一对多联系，一个顾客可对应多个订单，一个订单对应一个顾客
明细(Orders – OrderDetail)：一对多联系，一个订单对应多个明细，一个明细对应一个订单
包含(OrderDetail – Goods)：多对一联系，一个明细对应一个商品，一个商品对应多个明细
管理(Employee – Orders)：一对多联系，一个订单对应一个员工，一个员工可对应多个订单
管理(Manager – Empolyee)：一对多关系，一个经理对应多个员工，一个员工对应一个经理
这里为了方便向逻辑结构转换，统一采用二元关系去表达实体之间的联系。



2.2.2 数据库逻辑结构设计

顾客表(Customer)： 顾客 ID(主键) 姓名 联系方式 地址	员工表(Employee)： 员工 ID(主键) 姓名 联系方式 职位 工作时间 领导 ID(外键)	经理表(Manager)： 经理 ID(主键) 姓名 联系方式 职责
商品表(Goods)： 商品 ID(主键)，名称 进价，售价 过期日期 最大上架数量 货架上数量 摆放位置 库存数量	订单表(Order)： 订单号(主键) 日期 总售价 总利润 是否是过期清理 顾客 ID(外键) 员工 ID(外键)	商品订单明细(OrderDetail)： 明细 ID(主键) 订单号(外键) 商品 ID(外键) 数量

顾客表(Customer)设计如下

字段	数据类型	主键、外键	允许空	注释
customer_id	Integer	主键	否	顾客号
name	String		是	姓名
contact_info	String		是	联系方式
address	String		是	地址

经理表(Manager)设计如下

字段	数据类型	主键、外键	允许空	注释
manager_id	Integer	主键	否	经理号
name	String		否	姓名
contact_info	String		是	联系方式
duty	String		是	职责

员工表(Employee)设计如下

字段	数据类型	主键、外键	允许空	注释
employee_id	Integer	主键	否	员工号
name	String		否	姓名
contact_info	String		是	联系方式
position	String		是	职位
working_hours	String		是	工作时间
boss_id	Integer	外键引用经理	是	领导的编号

订单表(Orders)设计如下

字段	数据类型	主键、外键	允许空	注释
order_id	Integer	主键	否	订单号
order_date	Date		否	日期
total_price	Float		是	总售价
total_profit	Float		是	总利润
is_clearance	Boolean		是	是否过期清理
customer_id	Integer	外键引用顾客	是	顾客号
employee_id	Integer	外键引用员工	是	员工号

订单明细表(OrderDetail)设计如下

字段	数据类型	主键、外键	允许空	注释
employee_id	Integer	主键	否	明细号
order_id	Integer	外键引用订单	是	订单号
goods_id	Integer	外键引用商品	是	商品号
quantity	Integer		是	数量

商品表(Goods)设计如下

字段	数据类型	主键、外键	允许空	注释
goods_id	Integer	主键	否	商品号
name	String		否	名称
purch_price	Float		是	进价
selling_price	Float		是	售价
expire_date	Date		是	过期日期
max_quan	Integer		是	最大上架数量
shelf_quan	Integer		是	货架上数量
placement	String		是	摆放位置
stock_quan	Integer		是	库存数量

2.2.3 数据库物理结构设计

以下是索引结构的设计：全部使用 B+树创建索引

顾客表(Customer)	员工表(Employee)	经理表(Manager)
主键索引：顾客 ID	主键索引：员工 ID	主键索引：经理 ID
二级索引：姓名	二级索引：职位	二级索引：职责

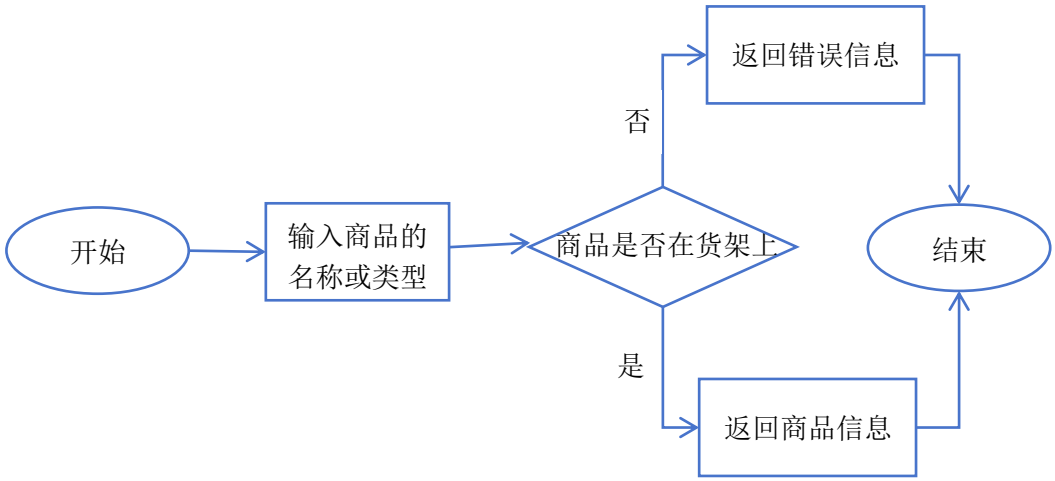
商品表(Goods) 主键索引：商品 ID 二级索引：名称	订单表(Order) 主键索引：订单号 二级索引：日期，顾客 ID	明细表(OrderDetail) 主键索引：明细 ID 二级索引：订单号、商品 ID

第三章 模块设计

大型超市数据库系统按照用户群体划分为三个模块：顾客模块、员工模块和经理模块。每个模块下面又分为多个子模块，比如顾客模块可以划分为：查看商品信息模块、查看订单模块、向订单增加商品模块、在订单中删除商品模块。本章主要选取顾客模块的查询商品信息子模块、员工模块的清除过期商品子模块和经理模块的删除员工子模块进行详细的介绍。

3.1 查询商品信息子模块

查询商品商品信息子模块的功能是实现客户能够根据商品的名称或者商品的种类，查询到相应商品的信息（包括位置信息、价格信息、商品上架数量等）。模块功能的具体流程图如下：



（图 3-1：查询商品信息子模块流程图）

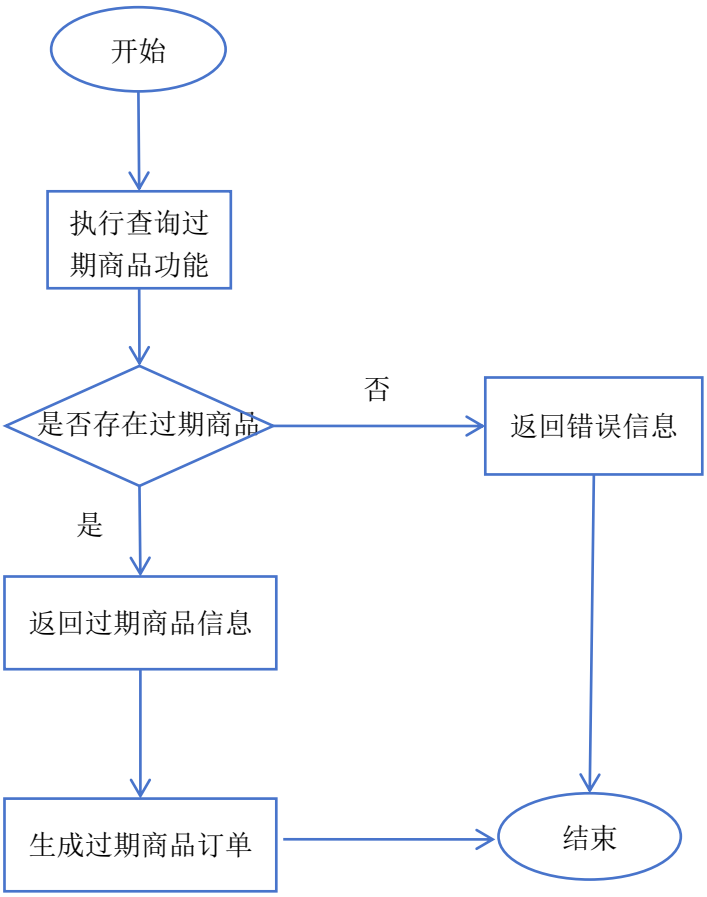
图 3-2 描述了查询商品信息子模块的详细代码实现，该模块的具体实现逻辑如下：定义了一个名为 GetGoodsInfo 的存储过程，用于获取商品信息。该存储过程有三个参数：orderType、searchValue 和 searchType。其中，orderType 用于指定排序方式（升序或降序），searchValue 表示搜索关键词（商品名称或商品种类），searchType 则表示搜索类型（按照商品名称或商品种类搜索）。存储过程内部根据输入的参数执行相应的查询语句，返回符合条件的商品信息。如果输入的排序方式不正确，则默认按照升序排序。在基于商品名称或商品种类搜索时，使用了类似于模糊搜索的方式，可以匹配包含输入关键词的商品。最后，如果没有指定搜索类型，则会返回所有商品信息，并按照商品编号排序。整个存储过程使用了 MySQL 的语法格式。

```
-- 顾客查看商品
-- 模式一：按照商品名称查看
-- 模式二：按照商品种类查看
-- 模式三：查看所有商品信息
CREATE PROCEDURE GetGoodsInfo(
    IN orderType VARCHAR(4),  -- 'ASC' 或 'DESC'，用于指定排序方式
    IN searchValue VARCHAR(255), -- 商品名称或商品种类
    IN searchType VARCHAR(9)  -- 'name' 或 'category'，指定搜索类型
)
BEGIN
    IF orderType NOT IN ('ASC', 'DESC') THEN
        SET orderType = 'ASC';  -- 如果输入的排序方式不正确，默认为升序
    END IF;

    IF searchType = 'name' THEN
        -- 基于商品名称搜索
        SELECT name, category, selling_price, expiration_date, shelf_quantity, goods_id
        FROM Goods
        WHERE name LIKE CONCAT('%', searchValue, '%')
        ORDER BY selling_price COLLATE utf8mb4_unicode_ci;
    ELSEIF searchType = 'category' THEN
        -- 基于商品种类搜索
        SELECT name, category, selling_price, expiration_date, shelf_quantity, goods_id
        FROM Goods
        WHERE category LIKE CONCAT('%', searchValue, '%')
        ORDER BY selling_price COLLATE utf8mb4_unicode_ci;
        ELSE
        -- 查看所有商品信息，按照orderType排序
        SELECT name, category, selling_price, expiration_date, shelf_quantity, goods_id
        FROM Goods
        ORDER BY goods_id COLLATE utf8mb4_unicode_ci;

    END IF;
END$$
```

（图 3-2：查询商品信息子模块的代码实现）

3.2 清除过期商品子模块

清除过期商品是员工用户的功能。员工首先使用查询过期商品功能，查看当前日期下的过期商品列表，如果没有过期商品，员工任务完成；如果有，员工可以得知过期商品的位置信息，从而到相应的位置下架过期商品，同时，员工在下架过期商品后，系统还需要自动生成过期订单，以便管理者查看亏损情况。其流程图如图 3-3 所示。



（图 3-3：清除过期商品子模块流程图）

图 3-4 则描述了清除过期商品信息子模块的详细代码实现，该模块的具体实现逻辑如下：首先，定义了一个名为 ClearExpiredGoods 的存储过程，用于清理过期的商品并进行相应的操作。该存储过程有

一个参数 input_goods_id，表示输入的过期商品 ID。

在存储过程内部，首先声明了几个局部变量：_stock_quantity（商品在仓库中的数量）、_shelf_quan（商品在货架上的数量）和_purchase_price（商品的进价）。接下来，通过查询语句从 Goods 表中获取指定商品 ID 的货架数量、进价和库存数量，并将结果赋值给相应的局部变量。然后，通过条件判断，如果货架上的数量大于 0，则执行清理操作：首先，向订单表插入一行数据，其中包括当前日期、总售价为 0、利润为数量乘以进价的负值（表示亏损），is_clearance 为 TRUE（表示清理过期商品导致的订单），customer_id 为 NULL（表示没有指定客户），employee_id 为 1（表示员工 ID）。

然后，获取新插入的订单 ID，使用 @last_order_id 变量保存。接着，向订单明细表插入一行数据，其中包括订单 ID（即刚刚插入的订单 ID）、商品 ID（即输入的过期商品 ID）和数量（即货架上的数量）。最后，更新 Goods 表，将货架上的数量和库存数量均设置为 0，表示清理了过期商品。

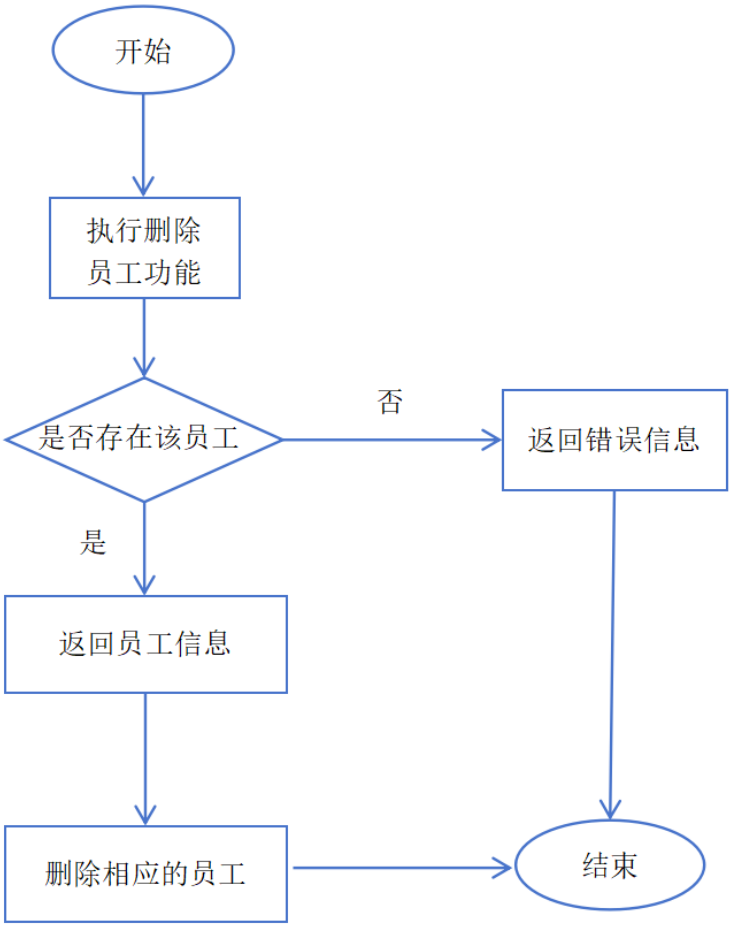
```
62  DELIMITER $$
63  -- 第四个存储过程{用于清理过期的商品，输入过期商品的goods_id；为订单明细增加一行：数量为商品在货架上的数量；
64  -- 为订单表增加一行，售价设置为0，利润设置为商品的数量x商品的进价，表明因商品过期导致亏损}
65  CREATE PROCEDURE ClearExpiredGoods(
66      IN input_goods_id INT  -- 输入的过期商品ID
67  )
68  BEGIN
69          DECLARE _stock_quantity INT;      -- 商品在仓库中的数量
70      DECLARE _shelf_quan INT;          -- 商品在货架上的数量
71      DECLARE _purchase_price DECIMAL(10, 2); -- 商品的进价
72
73      -- 获取商品在货架上的数量和进价
74      SELECT shelf_quantity, purchase_price, stock_quantity INTO _shelf_quan, _purchase_price, _stock_quantity
75      FROM Goods
76      WHERE goods_id = input_goods_id;
77
78      -- 如果货架上数量大于0，则进行清理操作
79      IF _shelf_quan > 0 THEN
80          -- 为订单表增加一行，售价为0，利润为数量乘以进价的负值
81          INSERT INTO Orders(order_date, total_price, total_profit, is_clearance, customer_id, employee_id)
82          VALUES(CURDATE(), 0, -(_shelf_quan+_stock_quantity) * _purchase_price, TRUE, NULL, 1);
```

```
84          -- 获取新插入的订单ID
85          SET @last_order_id = LAST_INSERT_ID();
86
87          -- 为订单明细表增加一行
88          INSERT INTO OrderDetail(order_id, goods_id, quantity)
89          VALUES(@last_order_id, input_goods_id, _shelf_quan);
90
91                  -- 更新Goods表，将货架上的数量和库存数量均设置为0
92          UPDATE Goods
93          SET shelf_quantity = 0, stock_quantity = 0
94          WHERE goods_id = input_goods_id;
95      END IF;
96  END$$
97
98  DELIMITER ;
```

（图 3-4：清除过期商品子模块的代码实现）

3.3 删除员工子模块

删除员工是经理用户的功能。经理可以删除现有的员工：如果输入的员工用户不在系统中，则返回错误信息；如果存在该员工，则在员工用户表中删除该员工。其流程图如图 3-5 所示。



（图 3-5：删除员工子模块的流程图）

图 3-6 则描述了删除员工子模块的详细代码实现，该模块的具体实现逻辑如下：首先，定义了一个名为 FireEmployee 的存储过程，用于解雇员工并从数据库中删除其账户。该存储过程有一个参数 employeeUsername，表示员工的用户名。

在存储过程内部，首先声明了一个异常处理程序（exit handler），用于在出现 SQL 异常时进行错误处理。当出现异常时，会执行 ROLLBACK 操作，回滚之前的事务。然后，开始一个事务（START TRANSACTION）。接下来，通过拼接字符串的方式生成一个 SQL 语句，用于删除指定员工账户。使用 CONCAT 函数将 DROP USER 命令与传入的员工用户名拼接起来，并保存在变量 @s 中。之后，使用 PREPARE 语句将 SQL 语句准备好，通过 EXECUTE 执行该语句，最后使用 DEALLOCATE PREPARE 释放准备语句。

最后，提交事务（COMMIT），完成解雇员工和账户删除的操作。

```
33  CREATE PROCEDURE FireEmployee(
34      IN employeeUsername VARCHAR(255)
35  )
36  BEGIN
37      DECLARE exit handler for sqlexception
38      BEGIN
39          -- 错误处理
40          ROLLBACK;
41      END;
42
43      START TRANSACTION;
44
45      -- 从数据库中删除员工账户
46      SET @s = CONCAT('DROP USER ', employeeUsername, '@\'localhost\'');
47      PREPARE stmt FROM @s;
48      EXECUTE stmt;
49      DEALLOCATE PREPARE stmt;
50
51      COMMIT;
52  END$$
```

（图 3-6：删除员工子模块的代码实现）

3.4 cpp 连接

我们使用 CPP，通过 ODBC 进行数据库的连接，以下是登录账户的逻辑。

```cpp
int main() {

    SQLRETURN ret; // 调用结果
    /* 定义句柄和变量 */
    SQLHENV src_env;  // 环境句柄
    SQLHDBC src_dbc;  // 连接句柄
    SQLHSTMT src_stmt; // 语句句柄

    int roleChoice;
    bool running = true;
    std::string username, password;

    while (running) {
        std::cout << "==========欢迎登录超市管理系统==========" << std::endl;
        std::cout << "请选择角色: 1. 顾客 2. 员工 3. 经理 4. 退出" << std::endl;
        std::cin >> roleChoice;
        // 现在连接是基于用户输入的用户名和密码
        std::string username, password;
        std::cout << "请输入账户: ";
        std::cin >> username;
        std::cout << "请输入密码: ";
        std::cin >> password;


        /* 初始化环境 */
        SQLAllocEnv(&src_env);
        // 设置管理环境的属性
        SQLSetEnvAttr(src_env, SQL_ATTR_ODBC_VERSION, (void *)SQL_OV_ODBC3, 0);

        /* 建立连接 */
        // 分配连接句柄
        ret = SQLAllocConnect(src_env, &src_dbc);

        ret = SQLConnect(src_dbc, (SQLCHAR *)"supermarket_in_mysql80", SQL_NTS,
                         (SQLCHAR *)username.c_str(), SQL_NTS,
                         (SQLCHAR *)password.c_str(), SQL_NTS);

        if (!SQL_SUCCEEDED(ret)) {
            std::cout << "账户或密码错误\n" << std::endl;
```

通过 switch 语句来进行用户的交互，操作的选择

```cpp
void customerFunctions(SQLHDBC src_dbc) {
    int functionChoice;
    std::string sortOrder, item, type;

    bool inCustomerMenu = true;

    while (inCustomerMenu) {
        std::cout << "\n=== 顾客功能 ===\n"
                  << "1. 查看商品\n"
                  << "2. 查找商品\n"
                  << "3. 购买商品\n"
                  << "4. 退出账号\n"
                  << "请选择操作: ";
        std::cin >> functionChoice;

        switch (functionChoice) {
            case 1:
                // 查询商品的实现
                callGetGoodsInfo(src_dbc, "ASC", "", "");
                break;
```

CPP 的函数只是一个外壳，通过变量连接 Sql Query 的输入和输出来实现。

```cpp
// 函数用于调用 员工的 CheckExpiredGoods 存储过程并输出结果
int callCheckExpiredGoodsProcedure(SQLHDBC src_dbc) {
    SQLHSTMT hStmt;
    SQLRETURN ret;

    // 分配语句句柄
    ret = SQLAllocHandle(SQL_HANDLE_STMT, src_dbc, &hStmt);
    if (!SQL_SUCCEEDED(ret)) {
        std::cerr << "Error allocating statement handle." << std::endl;
        return -1;
    }

    // 准备调用存储过程的 SQL 命令
    std::string procCall = "CALL CheckExpiredGoods()";

    // 执行存储过程
    ret = SQLExecDirect(hStmt, (SQLCHAR*)procCall.c_str(), SQL_NTS);
    if (!SQL_SUCCEEDED(ret)) {
        std::cerr << "Error executing the stored procedure." << std::endl;
        SQLFreeHandle(SQL_HANDLE_STMT, hStmt);
        return -1;
    }

    // 处理返回的数据
    SQLINTEGER goods_id;
    SQLCHAR name[256];
    SQLCHAR expiration_date[256];
    SQLINTEGER shelf_quantity;
    SQLINTEGER stock_quantity;

    while (SQLFetch(hStmt) == SQL_SUCCESS) {
        SQLGetData(hStmt, 1, SQL_C_SLONG, &goods_id, 0, NULL);
        SQLGetData(hStmt, 2, SQL_C_CHAR, name, sizeof(name), NULL);
        SQLGetData(hStmt, 3, SQL_C_CHAR, expiration_date, sizeof(expiration_date), NULL);
        SQLGetData(hStmt, 4, SQL_C_SLONG, &shelf_quantity, 0, NULL);
        SQLGetData(hStmt, 5, SQL_C_SLONG, &stock_quantity, 0, NULL);

        // 设置每个字段的宽度
        const int width_goods_id = 3;
        const int width_name = 16;
        const int width_expiration_date = 14; // 过期日期的宽度
        const int width_shelf_quantity = 5;
        const int width_stock_quantity = 5;

        std::cout << "ID: " << std::left << std::setw(width_goods_id) << goods_id
                  << "名称: " << std::left << std::setw(width_name) << name
                  << "过期日期: " << std::left << std::setw(width_expiration_date) << expiration_date
                  << "在架数量: " << std::left << std::setw(width_shelf_quantity) << shelf_quantity
                  << "库存数量: " << std::left << std::setw(width_stock_quantity) << stock_quantity
                  << std::endl;
    }
```

```cpp
// 函数用于 经理 增加商品的库存数量
int updateStockQuantity(SQLHDBC src_dbc, int goods_id, int increase_amount) {
    SQLHSTMT hStmt;
    SQLRETURN ret;

    // 分配语句句柄
    ret = SQLAllocHandle(SQL_HANDLE_STMT, src_dbc, &hStmt);
    if (!SQL_SUCCEEDED(ret)) {
        std::cerr << "Error allocating statement handle." << std::endl;
        return -1;
    }

    // 准备SQL更新命令
    char sqlQuery[256];
    sprintf(sqlQuery, "UPDATE Goods SET stock_quantity = stock_quantity + %d WHERE goods_id = %d", increase_amount, goods_id);

    // 执行SQL命令
    ret = SQLExecDirect(hStmt, (SQLCHAR*)sqlQuery, SQL_NTS);
    if (!SQL_SUCCEEDED(ret)) {
        std::cerr << "Error executing the update statement." << std::endl;
        SQLFreeHandle(SQL_HANDLE_STMT, hStmt);
        return -1;
    } else {
        std::cout << "Updated stock quantity successfully for goods ID: " << goods_id << std::endl;
    }

    // 释放语句句柄
    SQLFreeHandle(SQL_HANDLE_STMT, hStmt);
    return 0;
}
```

第四章 运行展示

4.1 顾客模块运行展示

顾客登录系统：

```
===========欢迎登录超市管理系统===========
请选择角色：1. 顾客 2. 员工 3. 经理 4. 退出
1
请输入账户：customer_0
请输入密码：123456
```

顾客查看所有商品信息：

```
=== 顾客功能 ===
1. 查看商品
2. 查找商品
3. 购买商品
4. 退出账号
请选择操作：1
ID: 1     名称：Apple          种类：Fruit          过期日期：2024-06-01     在架数量：94      位置：A1
ID: 2     名称：Milk           种类：Dairy          过期日期：2023-12-12     在架数量：0       位置：A2
ID: 3     名称：Bread          种类：Bakery         过期日期：2023-09-10     在架数量：0       位置：B1
ID: 4     名称：Chocolate      种类：Snack          过期日期：2024-02-28     在架数量：44      位置：B2
ID: 5     名称：Shampoo        种类：Personal Care  过期日期：2024-06-30     在架数量：70      位置：C1
ID: 7     名称：Orange Juice   种类：Beverage       过期日期：2023-11-20     在架数量：30      位置：D1
ID: 8     名称：Chicken        种类：Meat           过期日期：2023-12-31     在架数量：50      位置：E3
ID: 9     名称：Toothpaste     种类：Personal Care  过期日期：2025-01-01     在架数量：24      位置：C1
```

顾客按名称查看商品信息：

这里，顾客查找 Apple 商品，展示出该商品的数量和位置。

```
=== 顾客功能 ===
1. 查看商品
2. 查找商品
3. 购买商品
4. 退出账号
请选择操作：2
请输入查询类型 (name/category): name
请输入商品名称或类别：Apple
ID: 1     名称：Apple          种类：Fruit          过期日期：2024-06-01     在架数量：94      位置：A1
```

顾客按商品种类查看商品信息：顾客查找 Snack 类别的商品信息。

```
=== 顾客功能 ===
1. 查看商品
2. 查找商品
3. 购买商品
4. 退出账号
请选择操作：2
请输入查询类型 (name/category): category
请输入商品名称或类别：Snack
ID: 4     名称：Chocolate      种类：Snack          过期日期：2024-02-28     在架数量：44      位置：B2
```

顾客购买商品：

```
=== 顾客功能 ===
1. 查看商品
2. 查找商品
3. 购买商品
4. 退出账号
请选择操作：3
请输入商品ID：4
请输入商品数量：10
```

顾客购买前：购买商品 ID 为 4，name 为 chocolate 初始数量为 44

goods_id	name	category	purchase_price	selling_price	expiration_date	placement	max_on_shelf_quan	shelf_quantity	stock_quantity
1	Apple	Fruit	1.50	2.50	2024-06-01	A1	78	94	204
2	Milk	Dairy	0.90	1.40	2023-12-12	A2	80	30	0
3	Bread	Bakery	1.00	1.50	2023-09-10	B1	100	0	0
4	Chocolate	Snack	2.00	3.00	2024-02-28	B2	90	44	180
5	Shampoo	Personal Care	3.50	5.00	2024-06-30	C1	70	25	140
7	Orange Juic	Beverage	2.50	3.50	2023-11-20	D1	60	30	120
8	Chicken	Meat	4.00	6.00	2023-12-31	E3	100	50	200
9	Toothpaste	Personal Care	1.20	2.00	2025-01-01	C1	50	24	100

顾客购买后：chocolate 的数量变为 34

goods_id	name	category	purchase_price	selling_price	expiration_date	placement	max_on_shelf_quan	shelf_quantity	stock_quantity
1	Apple	Fruit	1.50	2.50	2024-06-01	A1	78	94	204
2	Milk	Dairy	0.90	1.40	2023-12-12	A2	80	0	0
3	Bread	Bakery	1.00	1.50	2023-09-10	B1	100	0	0
4	Chocolate	Snack	2.00	3.00	2024-02-28	B2	90	34	180
5	Shampoo	Personal Care	3.50	5.00	2024-06-30	C1	70	70	95
7	Orange Juic	Beverage	2.50	3.50	2023-11-20	D1	60	30	120
8	Chicken	Meat	4.00	6.00	2023-12-31	E3	100	50	200
9	Toothpaste	Personal Care	1.20	2.00	2025-01-01	C1	50	24	100

4.2 员工模块运行展示

员工登录系统：

```
============欢迎登录超市管理系统============
请选择角色：1. 顾客 2. 员工 3. 经理 4. 退出
2
请输入账户：employee_1
请输入密码：123456
```

员工查看缺货商品，根据缺货程度降序排序：

```
=== 员工功能 ===
1. 查看缺货商品
2. 补充货架商品
3. 查看过期商品
4. 清理过期商品
5. 查看所有商品
6. 退出账户
请选择操作：1
ID: 2  名称：Milk         种类：Dairy         位置：A2   在架数量：0    库存数量：0    最大上架数量：80   缺货比：1
ID: 3  名称：Bread        种类：Bakery        位置：B1   在架数量：0    库存数量：0    最大上架数量：100  缺货比：1
ID: 4  名称：Chocolate    种类：Snack         位置：B2   在架数量：34   库存数量：180  最大上架数量：90   缺货比：0.6222
ID: 9  名称：Toothpaste   种类：Personal Care 位置：C1   在架数量：24   库存数量：100  最大上架数量：50   缺货比：0.52
ID: 7  名称：Orange Juice 种类：Beverage      位置：D1   在架数量：30   库存数量：120  最大上架数量：60   缺货比：0.5
ID: 8  名称：Chicken      种类：Meat          位置：E3   在架数量：50   库存数量：200  最大上架数量：100  缺货比：0.5
```

员工补充缺货商品，库存不足则无法补货：

```
=== 员工功能 ===
1. 查看缺货商品
2. 补充货架商品
3. 查看过期商品
4. 清理过期商品
5. 查看所有商品
6. 退出账户
请选择操作：2
请输入需要补货的商品ID：4
Restocking procedure executed successfully for goods ID: 4

=== 员工功能 ===
1. 查看缺货商品
2. 补充货架商品
3. 查看过期商品
4. 清理过期商品
5. 查看所有商品
6. 退出账户
请选择操作：1
ID: 2  名称：Milk         种类：Dairy         位置：A2   在架数量：0    库存数量：0    最大上架数量：80   缺货比：1
ID: 3  名称：Bread        种类：Bakery        位置：B1   在架数量：0    库存数量：0    最大上架数量：100  缺货比：1
ID: 9  名称：Toothpaste   种类：Personal Care 位置：C1   在架数量：24   库存数量：100  最大上架数量：50   缺货比：0.52
ID: 7  名称：Orange Juice 种类：Beverage      位置：D1   在架数量：30   库存数量：120  最大上架数量：60   缺货比：0.5
ID: 8  名称：Chicken      种类：Meat          位置：E3   在架数量：50   库存数量：200  最大上架数量：100  缺货比：0.5
```

员工查看过期商品：

```
=== 员工功能 ===
1. 查看缺货商品
2. 补充货架商品
3. 查看过期商品
4. 清理过期商品
5. 查看所有商品
6. 退出账户
请选择操作: 3
ID: 7  名称: Orange Juice    过期日期: 2023-11-20    在架数量: 30   库存数量: 120
```

员工清理过期商品 7 后，再次查看，发现已经没有过期商品：

```
=== 员工功能 ===
1. 查看缺货商品
2. 补充货架商品
3. 查看过期商品
4. 清理过期商品
5. 查看所有商品
6. 退出账户
请选择操作: 4
请输入需要清理的过期商品ID: 7
ClearExpiredGoods procedure executed successfully for goods ID: 7

=== 员工功能 ===
1. 查看缺货商品
2. 补充货架商品
3. 查看过期商品
4. 清理过期商品
5. 查看所有商品
6. 退出账户
请选择操作: 3
```

员工可以查看所有商品的更多信息：

```
=== 员工功能 ===
1. 查看缺货商品
2. 补充货架商品
3. 查看过期商品
4. 清理过期商品
5. 查看所有商品
6. 退出账户
请选择操作: 5
ID: 1   名称: Apple          位置: A1    过期日期: 2024-06-01   在架数量: 94   最大上架数量: 78   库存数量: 204
ID: 2   名称: Milk           位置: A2    过期日期: 2023-12-12   在架数量: 0    最大上架数量: 80   库存数量: 0
ID: 3   名称: Bread          位置: B1    过期日期: 2023-09-10   在架数量: 0    最大上架数量: 100  库存数量: 0
ID: 4   名称: Chocolate      位置: B2    过期日期: 2024-02-28   在架数量: 90   最大上架数量: 90   库存数量: 124
ID: 5   名称: Shampoo        位置: C1    过期日期: 2024-06-30   在架数量: 70   最大上架数量: 70   库存数量: 95
ID: 7   名称: Orange Juice   位置: D1    过期日期: 2023-11-20   在架数量: 0    最大上架数量: 60   库存数量: 0
ID: 8   名称: Chicken        位置: E3    过期日期: 2023-12-31   在架数量: 50   最大上架数量: 100  库存数量: 200
ID: 9   名称: Toothpaste     位置: C1    过期日期: 2025-01-01   在架数量: 24   最大上架数量: 50   库存数量: 100
```

4.3 经理模块运行展示

经理查看所有商品：

```
=== 经理功能 ===
1. 查看所有商品
2. 修改上架数量
3. 补充库存数量
4. 查看销售报告
5. 开除员工
6. 招聘员工
7. 退出账号
请选择操作: 1
ID: 1  名称: Apple          种类: Fruit          进价: 1.5    售价: 2.5    最大上架数量: 78     库存数量: 204
ID: 2  名称: Milk           种类: Dairy          进价: 0.9    售价: 1.4    最大上架数量: 80     库存数量: 0
ID: 3  名称: Bread          种类: Bakery         进价: 1      售价: 1.5    最大上架数量: 100    库存数量: 0
ID: 4  名称: Chocolate      种类: Snack          进价: 2      售价: 3      最大上架数量: 90     库存数量: 124
ID: 5  名称: Shampoo        种类: Personal Care  进价: 3.5    售价: 5      最大上架数量: 70     库存数量: 95
ID: 7  名称: Orange Juice   种类: Beverage       进价: 2.5    售价: 3.5    最大上架数量: 60     库存数量: 0
ID: 8  名称: Chicken        种类: Meat           进价: 4      售价: 6      最大上架数量: 100    库存数量: 200
ID: 9  名称: Toothpaste     种类: Personal Care  进价: 1.2    售价: 2      最大上架数量: 50     库存数量: 100
```

经理查看从开始日期到结束日期的每种商品的销售情况：

```
=== 经理功能 ===
1. 查看所有商品
2. 修改上架数量
3. 补充库存数量
4. 查看销售报告
5. 开除员工
6. 招聘员工
7. 退出账号
请选择操作: 4
请输入销售报告的开始日期 (YYYY-MM-DD): 2023-11-01
请输入销售报告的结束日期 (YYYY-MM-DD): 2023-12-31
商品ID: 1  名称: Apple        总数量: 10   总销售额: 25      总利润: 10      最大上架数量: 78     库存数量: 204
商品ID: 2  名称: Milk         总数量: 214  总销售额: 299.6   总利润: 107     最大上架数量: 80     库存数量: 0
商品ID: 3  名称: Bread        总数量: 177  总销售额: 265.5   总利润: 88.5    最大上架数量: 100    库存数量: 0
商品ID: 4  名称: Chocolate    总数量: 11   总销售额: 33      总利润: 11      最大上架数量: 90     库存数量: 124
商品ID: 5  名称: Shampoo      总数量: 10   总销售额: 50      总利润: 15      最大上架数量: 70     库存数量: 95
商品ID: 7  名称: Orange Juice 总数量: 30   总销售额: 105     总利润: 30      最大上架数量: 60     库存数量: 0
商品ID: 9  名称: Toothpaste   总数量: 1    总销售额: 2       总利润: 0.8     最大上架数量: 50     库存数量: 100
```

经理查看到牛奶 milk 的销售情况比较好，于是决定在货架上摆更多的牛奶，可见将最大上架数量改成 140 后，确实变化了。

```
=== 经理功能 ===
1. 查看所有商品
2. 修改上架数量
3. 补充库存数量
4. 查看销售报告
5. 开除员工
6. 招聘员工
7. 退出账号
请选择操作: 2
请输入商品ID: 2
请输入最大上架数量: 140
Updated max shelf quantity successfully for goods ID: 2

=== 经理功能 ===
1. 查看所有商品
2. 修改上架数量
3. 补充库存数量
4. 查看销售报告
5. 开除员工
6. 招聘员工
7. 退出账号
请选择操作: 1
ID: 1  名称: Apple          种类: Fruit          进价: 1.5    售价: 2.5    最大上架数量: 78     库存数量: 204
ID: 2  名称: Milk           种类: Dairy          进价: 0.9    售价: 1.4    最大上架数量: 140    库存数量: 0
```

牛奶销售情况好，缺货了，经理决定补货 100 件：

```
=== 经理功能 ===
1. 查看所有商品
2. 修改上架数量
3. 补充库存数量
4. 查看销售报告
5. 开除员工
6. 招聘员工
7. 退出账号
请选择操作: 3
请输入商品ID: 2
请输入要增加的库存数量: 100
Updated stock quantity successfully for goods ID: 2

=== 经理功能 ===
1. 查看所有商品
2. 修改上架数量
3. 补充库存数量
4. 查看销售报告
5. 开除员工
6. 招聘员工
7. 退出账号
请选择操作: 1
ID: 1  名称: Apple          种类: Fruit          进价: 1.5    售价: 2.5    最大上架数量: 78     库存数量: 204
ID: 2  名称: Milk           种类: Dairy          进价: 0.9    售价: 1.4    最大上架数量: 140    库存数量: 100
```

员工 employee_1 上班偷懒，经理决定开除他：

```
=== 经理功能 ===
1. 查看所有商品
2. 修改上架数量
3. 补充库存数量
4. 查看销售报告
5. 开除员工
6. 招聘员工
7. 退出账号
请选择操作: 5
请输入要删除的员工: employee_1
Employee 'employee_1' has been successfully fired.
```

员工 employee_1 被开除后，无法登录系统：

```
===========欢迎登录超市管理系统===========
请选择角色: 1. 顾客 2. 员工 3. 经理 4. 退出
2
请输入账户: employee_1
请输入密码: 123456
账户或密码错误
```

开除了一个员工后，经理决定新找一个员工 employee_2：

```
=== 经理功能 ===
1. 查看所有商品
2. 修改上架数量
3. 补充库存数量
4. 查看销售报告
5. 开除员工
6. 招聘员工
7. 退出账号
请选择操作: 6
请输入要增加的员工: employee_2
New employee 'employee_2' has been successfully hired.
```

新员工密码默认是 123456，可见登录成功：