|  |
| --- |
|  |

**数据库原理与实践期末项目**

**学号：**

**姓名：**



数据库原理与实践期末项目

20 24 —20 25 学年 一 学期

|  |  |
| --- | --- |
| 课程名称： | 数据库原理及应用 |
| 专业学院： | 现代技术学院 |
| 专业班级： | 计科B2215 |
| 学 号： |  |
| 姓 名： |  |
| 指导教师： | 李小蓉 |

目 录

[1、 项目背景及需求分析 1](#_Toc153742448)

[1.1项目背景 1](#_Toc153742449)

[1.2需求分析 1](#_Toc153742450)

[2、 概念结构设计 2](#_Toc153742451)

[2.1抽象出系统实体 2](#_Toc153742452)

[2.2局部E-R图 2](#_Toc153742453)

[2.3全局E-R图 2](#_Toc153742454)

[3、 逻辑结构设计 3](#_Toc153742455)

[3.1简单说明 3](#_Toc153742456)

[3.2关系模式 3](#_Toc153742457)

[3.3表结构 3](#_Toc153742458)

[4、 创建数据库及SQL相关操作 4](#_Toc153742459)

[4.1 创建数据库 4](#_Toc153742460)

[4.1.1 创建表 4](#_Toc153742461)

[4.1.2 添加记录 4](#_Toc153742462)

[4.1.3 修改记录 5](#_Toc153742463)

[4.2 删除记录 5](#_Toc153742464)

[4.3查询问题及查询结果 5](#_Toc153742465)

[4.3.1 选择列 5](#_Toc153742466)

[4.3.2函数使用 5](#_Toc153742467)

[4.3.3 LIKE子句 6](#_Toc153742468)

[4.3.4 条件查询 6](#_Toc153742469)

[4.4.5多表查询 6](#_Toc153742470)

[4.4.6子查询 6](#_Toc153742471)

[4.5建立视图 7](#_Toc153742472)

[4.5.1 定义视图 7](#_Toc153742473)

[4.5.2利用视图插入数据 7](#_Toc153742474)

[5、建立存储过程及函数 8](#_Toc153742475)

[5.1创建存储过程 8](#_Toc153742476)

[5.2存储过程应用 8](#_Toc153742477)

[6、建立触发器 9](#_Toc153742478)

[7、项目设计总结 10](#_Toc153742479)

# 1、 项目背景及需求分析

## 1.1项目背景

仓库管理系统是针对仓储中心的库房、职工和商品进行管理的信息系统。在现代物流管理中，仓储环节起着至关重要的作用，对于企业的运营效率和客户满意度有着直接影响。传统的手工管理方式存在着信息不准确、效率低下、管理困难等问题，因此需要引入仓库管理系统来提高管理效率和准确性。

仓库管理系统的应用背景包括以下几个方面：  
a. 库存管理：仓库管理系统能够实时跟踪商品的入库和出库情况，准确记录库存数量和位置，帮助企业合理安排库存，避免库存过多或过少的情况发生。  
b. 订单处理：系统能够根据订单需求自动分配库房和职工，提高订单处理的效率和准确性，确保订单能够及时发货。  
c. 进销存管理：系统可以记录商品的采购信息、销售信息和库存变动情况，帮助企业进行进销存的分析和预测，优化采购和销售策略。  
d. 职工管理：系统能够管理职工的个人信息和工作分配，提供职工考勤和绩效评估等功能，提高职工管理的效率和准确性

## 1.2需求分析

a. 商品信息管理：能够对商品信息进行增加、删除和修改操作，包括商品名、类别、规格、单价和计量单位等。

b. 库房信息管理：能够对库房信息进行增加、删除和更新操作，包括库房号、库名、地点和面积等。

c. 职工信息管理：能够对职工信息进行增加、删除和更新操作，包括工号、姓名、性别和电话等。

d. 出入库管理：能够记录商品的出入库操作类型、品名、数量、存放的库房、经手的职工和操作日期等信息。

e. 查询功能：提供灵活的查询功能，能够根据商品、库房、职工等条件进行查询，方便用户获取所需信息。

f. 系统维护功能：提供对系统基本信息的维护功能，包括用户管理、权限管理、备份和恢复等

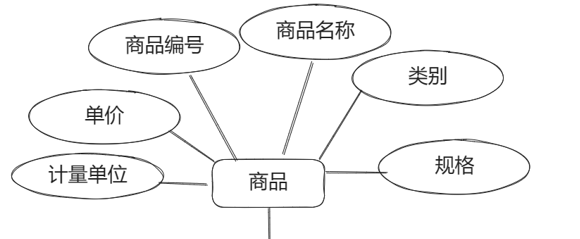
# 2、 概念结构设计

## 2.1抽象出系统实体

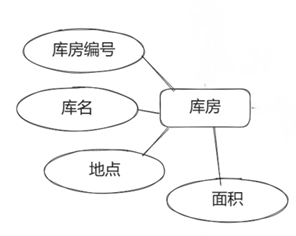
实体包括：商品、仓库、职工

## 2.2局部E-R图

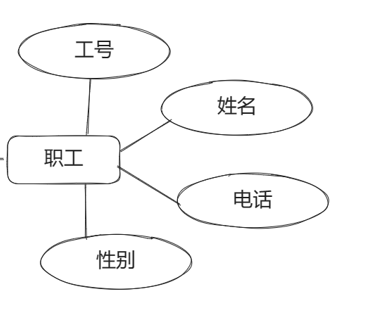
商品er图：



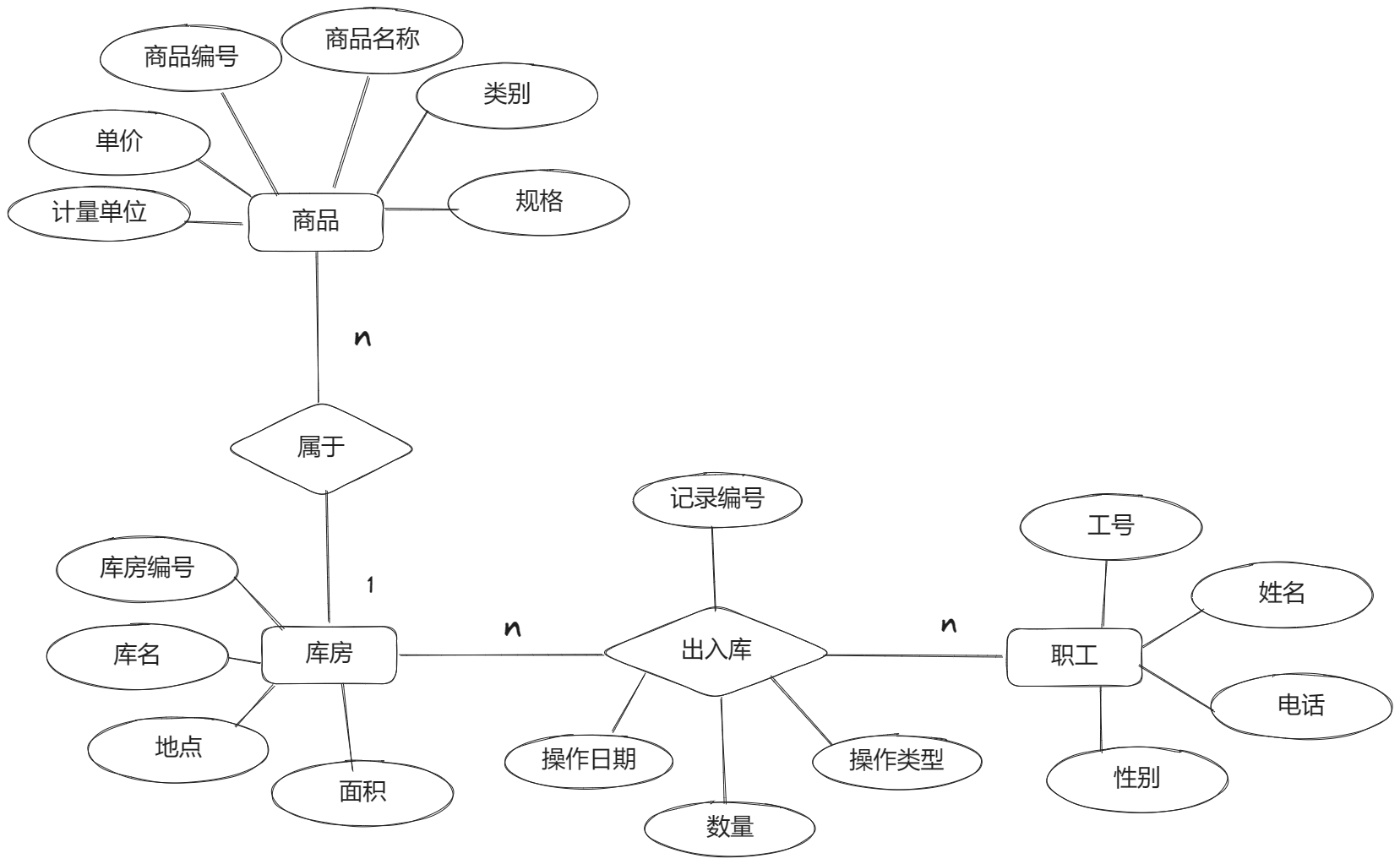
库房er图：



职工er图：



## 2.3全局E-R图



# 3、 逻辑结构设计

## 3.1简单说明

商品和库房是多对一的关系

职工和库房是多对一的关系

出入库信息包含商品进出库房和职工操作等信息

## 3.2关系模式

a. 商品信息表（商品编号，商品名，类别，规格，单价，计量单位）

b. 库房信息表（库房编号，库名，地点，面积）

c. 职工信息表（工号，姓名，性别，电话）

d. 出入库信息表（记录编号，操作类型，商品编号，数量，库房编号，职工工号，操作日期）

下划线是主键，波浪线是外键

## 3.3表结构

表5-1商品表信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 字段名 | 数据类型 | 是否可空约束 | 字段描述 |
| 1 | commodity\_id | INT | 是 | 商品ID，自增 |
| 2 | commodity\_name | VARCHAR(100) | 是 | 商品名称 |
| 3 | category | VARCHAR(100) | 是 | 商品类别 |
| 4 | specification | VARCHAR(100) | 是 | 商品规格 |
| 5 | unit\_price | DECIMAL(10,2) | 是 | 商品单价 |
| 6 | unit | VARCHAR(50) | 是 | 商品单位 |

表5-2仓库表信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 字段名 | 数据类型 | 是否可空约束 | 字段描述 |
| 1 | warehouse\_id | INT | 是 | 仓库ID，自增 |
| 2 | warehouse\_name | VARCHAR(100) | 是 | 仓库名称 |
| 3 | location | VARCHAR(100) | 是 | 仓库位置 |
| 4 | area | DECIMAL(10,2) | 是 | 仓库面积 |

表5-3员工表信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 字段名 | 数据类型 | 是否可空约束 | 字段描述 |
| 1 | employee\_id | INT | 是 | 员工ID，自增 |
| 2 | employee\_name | VARCHAR(100) | 是 | 员工姓名 |
| 3 | gender | VARCHAR(10) | 是 | 员工性别 |
| 4 | phone | VARCHAR(20) | 是 | 员工电话 |

表5-4出入库记录表信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 字段名 | 数据类型 | 是否可空约束 | 字段描述 |
| 1 | record\_id | INT | 是 | 记录ID，自增 |
| 2 | operation\_type | VARCHAR(20) | 是 | 操作类型 |
| 3 | commodity\_id | INT | 是 | 商品ID，非空 |
| 4 | quantity | INT | 是 | 数量 |
| 5 | warehouse\_id | INT | 是 | 仓库ID，非空 |
| 6 | employee\_id | INT | 是 | 员工ID，非空 |
| 7 | operation\_date | DATE | 是 | 操作日期 |

# 4、 创建数据库及SQL相关操作

（以下除特别说明，需要用代码完成）

## 4.1 创建数据库

建立数据库，数据库名称自取，设置字符集为UTF-8,排序规则为utf8\_general\_ci。

代码：

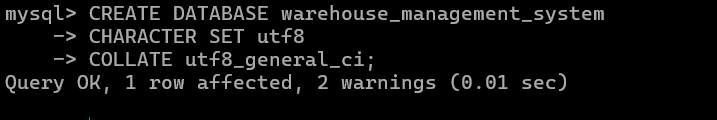
CREATE DATABASE warehouse\_management\_system

CHARACTER SET utf8

COLLATE utf8\_general\_ci;

USE warehouse\_management\_system;

截图结果：



### 4.1.1 创建表

-- 创建商品信息表

CREATE TABLE Commodity (

commodity\_id INT PRIMARY KEY AUTO\_INCREMENT,

commodity\_name VARCHAR(100) NOT NULL,

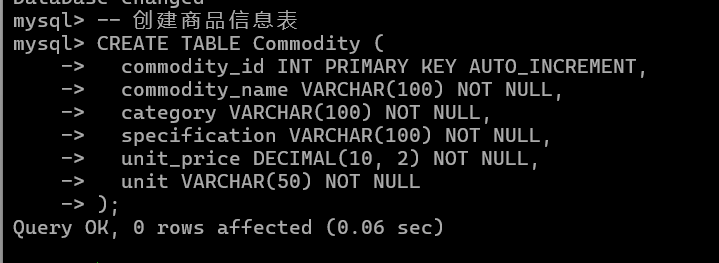
category VARCHAR(100) NOT NULL,

specification VARCHAR(100) NOT NULL,

unit\_price DECIMAL(10, 2) NOT NULL,

unit VARCHAR(50) NOT NULL

);



-- 创建库房信息表

CREATE TABLE Warehouse (

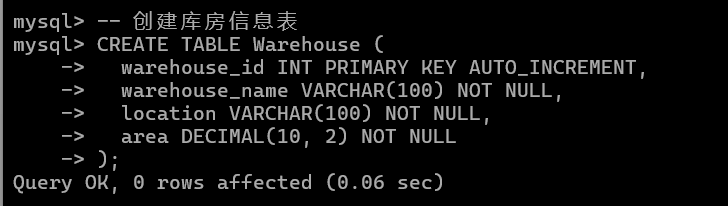
warehouse\_id INT PRIMARY KEY AUTO\_INCREMENT,

warehouse\_name VARCHAR(100) NOT NULL,

location VARCHAR(100) NOT NULL,

area DECIMAL(10, 2) NOT NULL

);



-- 创建职工信息表

CREATE TABLE Employee (

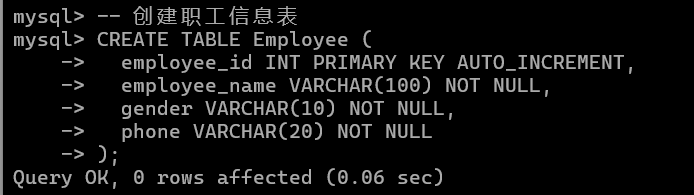
employee\_id INT PRIMARY KEY AUTO\_INCREMENT,

employee\_name VARCHAR(100) NOT NULL,

gender VARCHAR(10) NOT NULL,

phone VARCHAR(20) NOT NULL

);



-- 创建出入库信息表

CREATE TABLE InOutRecord (

record\_id INT PRIMARY KEY AUTO\_INCREMENT,

operation\_type VARCHAR(20) NOT NULL,

commodity\_id INT NOT NULL,

quantity INT NOT NULL,

warehouse\_id INT NOT NULL,

employee\_id INT NOT NULL,

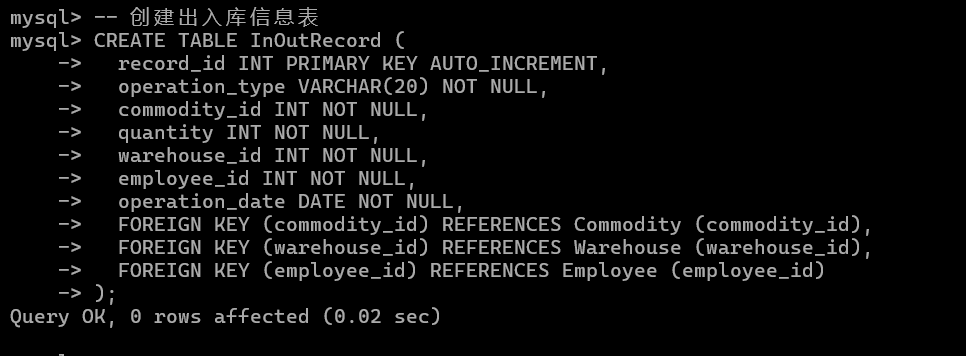
operation\_date DATE NOT NULL,

FOREIGN KEY (commodity\_id) REFERENCES Commodity (commodity\_id),

FOREIGN KEY (warehouse\_id) REFERENCES Warehouse (warehouse\_id),

FOREIGN KEY (employee\_id) REFERENCES Employee (employee\_id)

);



### 4.1.2 添加记录

-- 插入 Commodity 表的记录

INSERT INTO Commodity (commodity\_name, category, specification, unit\_price, unit)

VALUES

('笔记本电脑', '电子产品', '16GB RAM, 512GB SSD', 6999.99, '台'),

('手机', '电子产品', '128GB', 3999.99, '部'),

('打印机', '办公设备', '激光打印机', 1299.99, '台'),

('显示器', '电子产品', '27英寸', 1999.99, '台'),

('路由器', '网络设备', 'Wi-Fi 6', 499.99, '个'),

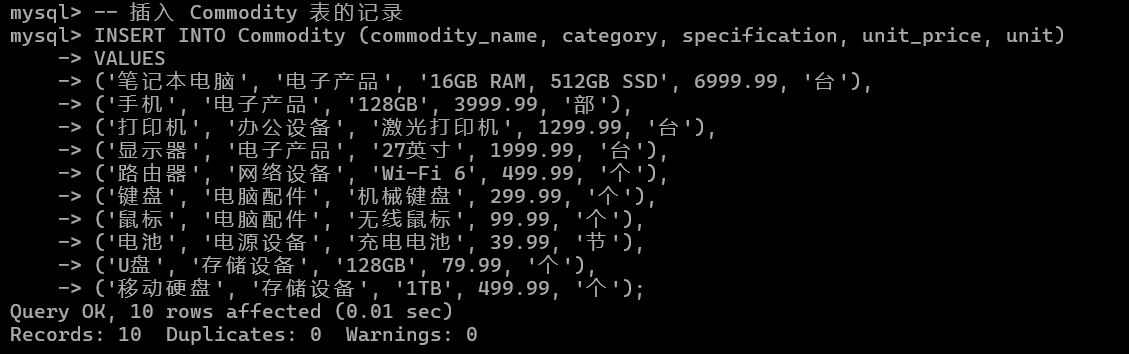
('键盘', '电脑配件', '机械键盘', 299.99, '个'),

('鼠标', '电脑配件', '无线鼠标', 99.99, '个'),

('电池', '电源设备', '充电电池', 39.99, '节'),

('U盘', '存储设备', '128GB', 79.99, '个'),

('移动硬盘', '存储设备', '1TB', 499.99, '个');



-- 插入 Warehouse 表的记录

INSERT INTO Warehouse (warehouse\_name, location, area)

VALUES

('仓库A', '北京市朝阳区', 1000.00),

('仓库B', '上海市浦东新区', 800.00),

('仓库C', '广州市天河区', 1200.00),

('仓库D', '深圳市南山区', 900.00),

('仓库E', '杭州市西湖区', 750.00),

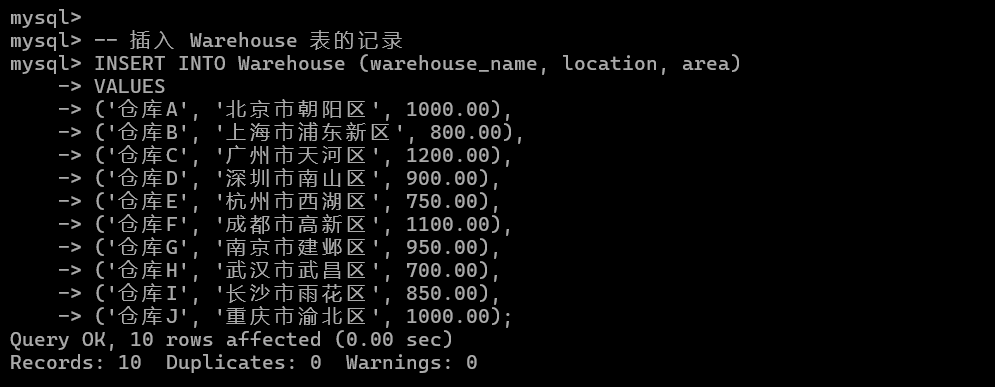
('仓库F', '成都市高新区', 1100.00),

('仓库G', '南京市建邺区', 950.00),

('仓库H', '武汉市武昌区', 700.00),

('仓库I', '长沙市雨花区', 850.00),

('仓库J', '重庆市渝北区', 1000.00);



-- 插入 Employee 表的记录

INSERT INTO Employee (employee\_name, gender, phone)

VALUES

('张三', '男', '13800000001'),

('李四', '女', '13800000002'),

('王五', '男', '13800000003'),

('赵六', '女', '13800000004'),

('陈七', '男', '13800000005'),

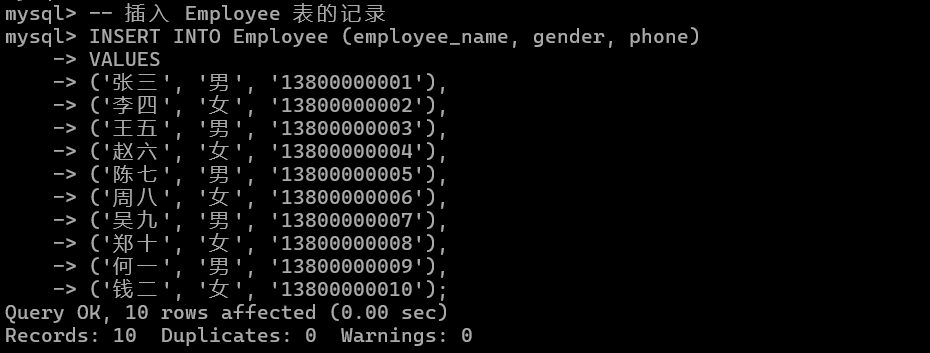
('周八', '女', '13800000006'),

('吴九', '男', '13800000007'),

('郑十', '女', '13800000008'),

('何一', '男', '13800000009'),

('钱二', '女', '13800000010');



-- 插入 InOutRecord 表的记录

INSERT INTO InOutRecord (operation\_type, commodity\_id, quantity, warehouse\_id, employee\_id, operation\_date)

VALUES

('入库', 1, 50, 1, 1, '2024-01-01'),

('入库', 2, 30, 2, 2, '2024-01-02'),

('入库', 3, 20, 3, 3, '2024-01-03'),

('入库', 4, 40, 4, 4, '2024-01-04'),

('入库', 5, 60, 5, 5, '2024-01-05'),

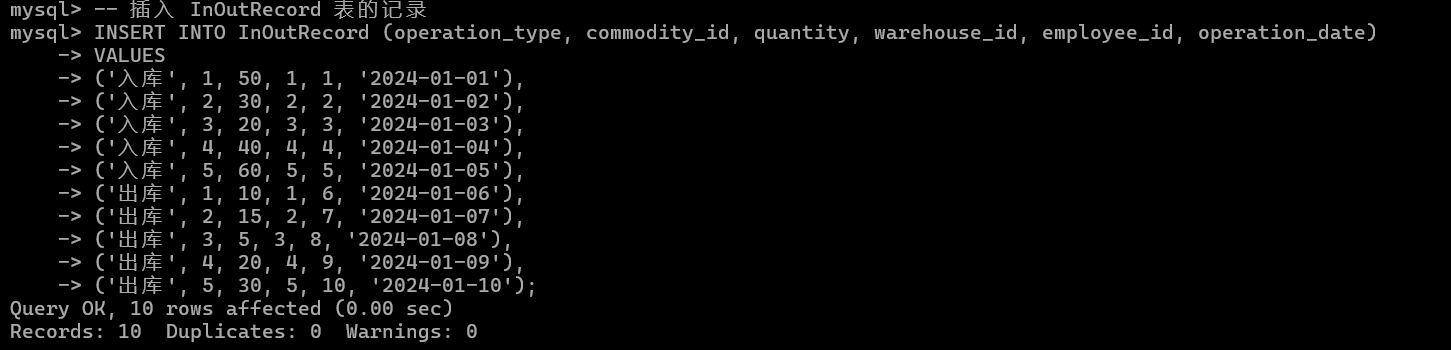
('出库', 1, 10, 1, 6, '2024-01-06'),

('出库', 2, 15, 2, 7, '2024-01-07'),

('出库', 3, 5, 3, 8, '2024-01-08'),

('出库', 4, 20, 4, 9, '2024-01-09'),

('出库', 5, 30, 5, 10, '2024-01-10');



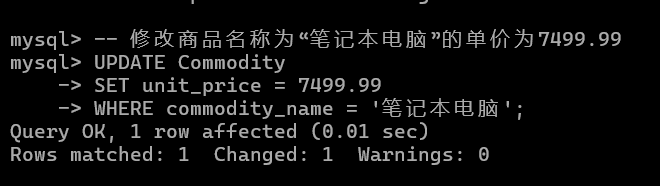
### 4.1.3 修改记录

-- 修改商品名称为“笔记本电脑”的单价为7499.99

UPDATE Commodity

SET unit\_price = 7499.99

WHERE commodity\_name = '笔记本电脑';

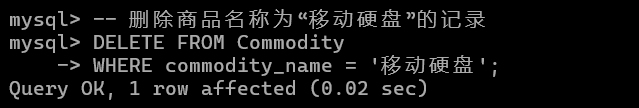


## 4.2 删除记录

-- 删除商品名称为“移动硬盘”的记录

DELETE FROM Commodity

WHERE commodity\_name = '移动硬盘';



## 4.3查询问题及查询结果

### 4.3.1 选择列

自行设计查询单表中的几列，并用别名代替。

代码：

-- 查询商品表的名称、分类和单价，给出别名

SELECT

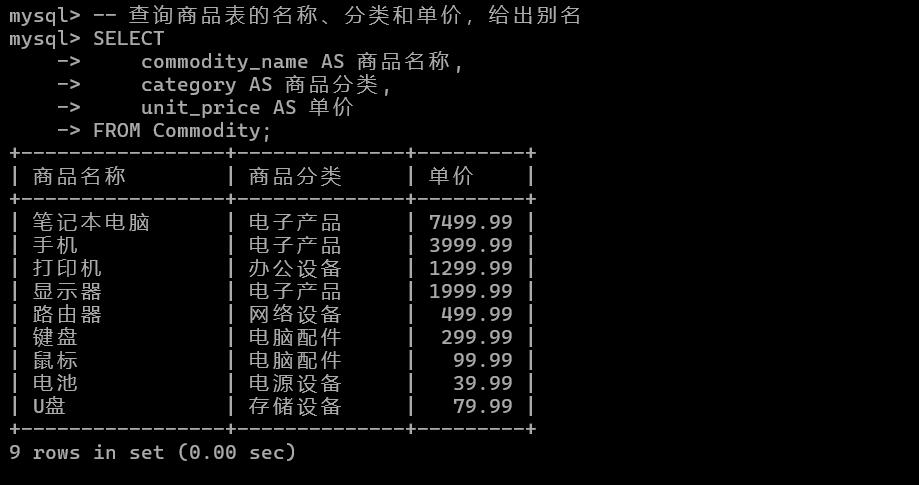
commodity\_name AS 商品名称,

category AS 商品分类,

unit\_price AS 单价

FROM Commodity;

结果截图：



### 4.3.2函数使用

计算满足条件的平均值，最大值，最小值。

代码：

-- 计算商品单价大于100的商品的平均值、最大值、最小值

SELECT

AVG(unit\_price) AS 平均单价,

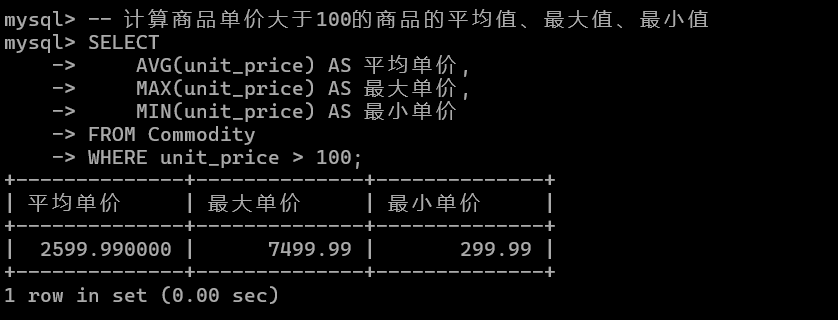
MAX(unit\_price) AS 最大单价,

MIN(unit\_price) AS 最小单价

FROM Commodity

WHERE unit\_price > 100;

结果截图：



### 4.3.3 LIKE子句

用LIKE子句实现模糊查询

说明实现的功能：

代码：

-- 查询商品名称中包含“电脑”的商品

SELECT \*

FROM Commodity

WHERE commodity\_name LIKE '%电脑%';

结果截图：



### 4.3.4 条件查询

用给定条件，查询结果

说明实现的功能：

代码：

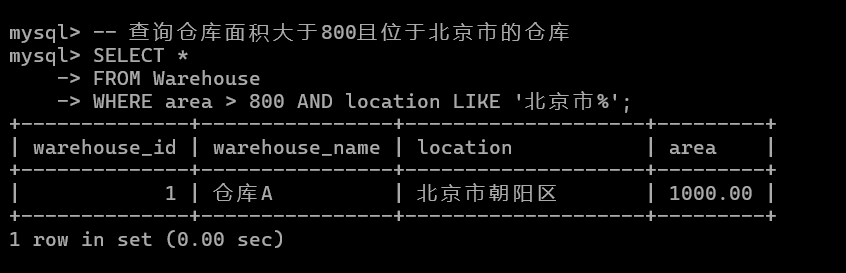
-- 查询仓库面积大于800且位于北京市的仓库

SELECT \*

FROM Warehouse

WHERE area > 800 AND location LIKE '北京市%';

结果截图：



### 4.4.5多表查询

至少牵扯两个表，根据给定条件，查询结果

说明实现的功能：

代码：

-- 查询出入库记录中商品名称、仓库名称和操作类型

SELECT

InOutRecord.operation\_type AS 操作类型,

Commodity.commodity\_name AS 商品名称,

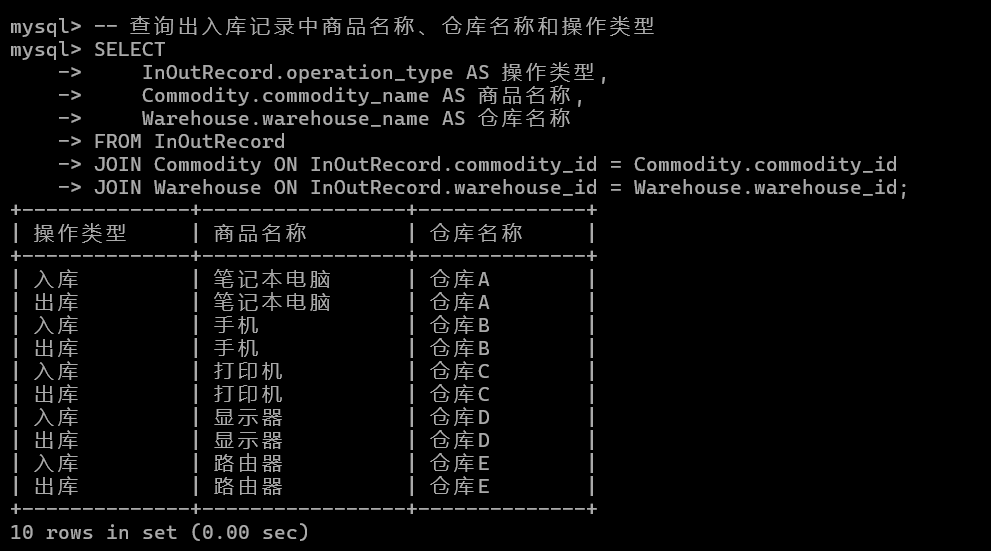
Warehouse.warehouse\_name AS 仓库名称

FROM InOutRecord

JOIN Commodity ON InOutRecord.commodity\_id = Commodity.commodity\_id

JOIN Warehouse ON InOutRecord.warehouse\_id = Warehouse.warehouse\_id;

结果截图：



### 4.4.6子查询

单行子查询举例

说明实现的功能：

代码：

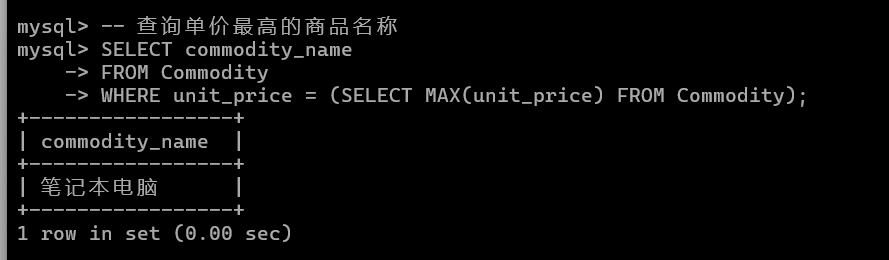
-- 查询单价最高的商品名称

SELECT commodity\_name

FROM Commodity

WHERE unit\_price = (SELECT MAX(unit\_price) FROM Commodity);

结果截图：



多行子查询举例（用in，any,all任意一个均可）

说明实现的功能：

代码：

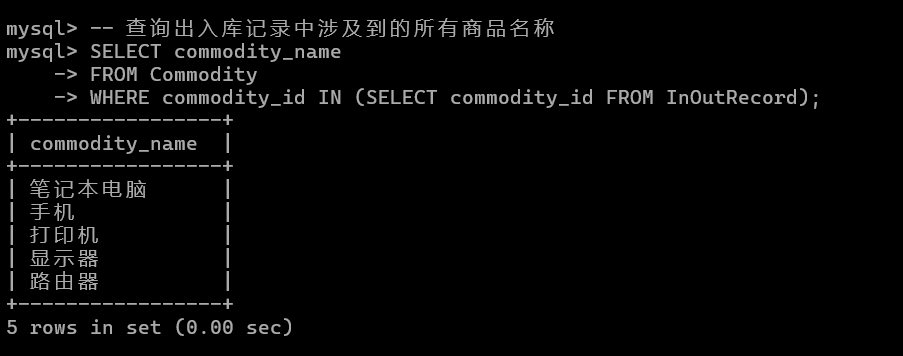
-- 查询出入库记录中涉及到的所有商品名称

SELECT commodity\_name

FROM Commodity

WHERE commodity\_id IN (SELECT commodity\_id FROM InOutRecord);

结果截图：



## 4.5建立视图

建立一个用户user01，给user01账户授予建立视图的权限。

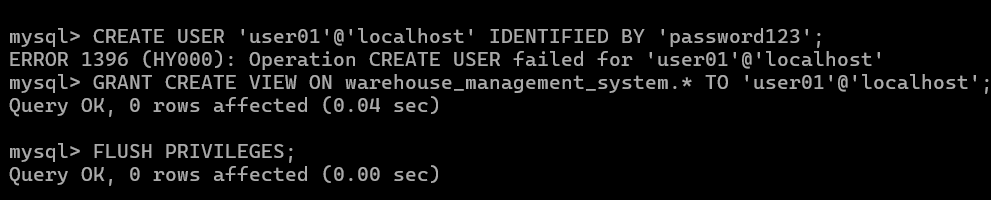
代码：

CREATE USER 'user01'@'localhost' IDENTIFIED BY 'password123';

GRANT CREATE VIEW ON warehouse\_management\_system.\* TO 'user01'@'localhost';

FLUSH PRIVILEGES;

结果截图：



### 4.5.1 定义视图

自定义一个视图，要求实现多表查询。

代码：

-- 建立视图，显示出入库记录的详细信息

CREATE VIEW View\_InOutDetails AS

SELECT

InOutRecord.record\_id AS 记录编号,

InOutRecord.operation\_type AS 操作类型,

Commodity.commodity\_name AS 商品名称,

InOutRecord.quantity AS 数量,

Warehouse.warehouse\_name AS 仓库名称,

Employee.employee\_name AS 操作员工,

InOutRecord.operation\_date AS 操作日期

FROM InOutRecord

JOIN Commodity ON InOutRecord.commodity\_id = Commodity.commodity\_id

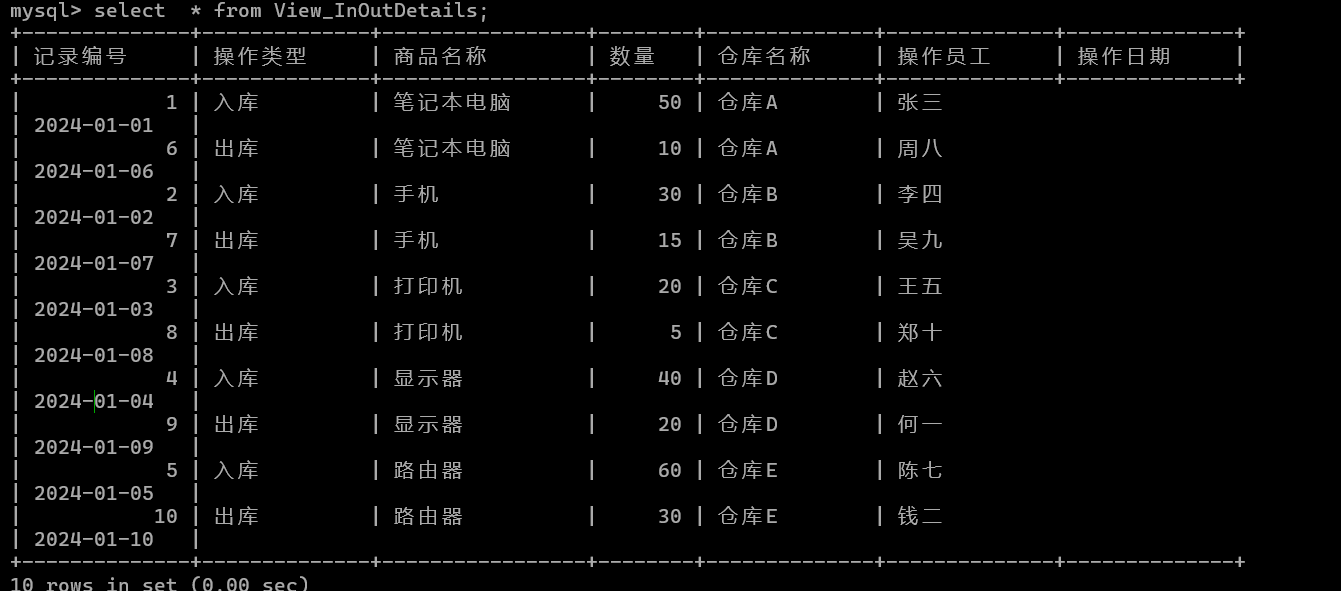
JOIN Warehouse ON InOutRecord.warehouse\_id = Warehouse.warehouse\_id

JOIN Employee ON InOutRecord.employee\_id = Employee.employee\_id;

查询自定义的视图代码：

select \* from View\_InOutDetails;

结果截图：



### 4.5.2利用视图插入数据

创建简单视图代码：

-- 创建一个简单视图，显示商品的ID、名称和单价

CREATE VIEW SimpleView AS

SELECT commodity\_id, commodity\_name, unit\_price, category, specification, unit

FROM Commodity;

利用视图向表中插入一条记录代码：

-- 使用视图向 Commodity 表中插入一条新记录

-- 插入一条完整记录 INSERT INTO SimpleView (commodity\_id, commodity\_name, unit\_price, category, specification, unit) VALUES (11, '台灯', 59.99, '家居用品', '可调节亮度', '个');

查询插入的记录代码：

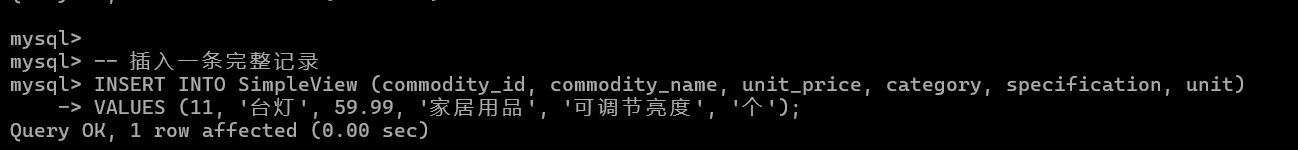
-- 查询刚插入的记录

SELECT \*

FROM Commodity

WHERE commodity\_name = '台灯';

查询结果截图：



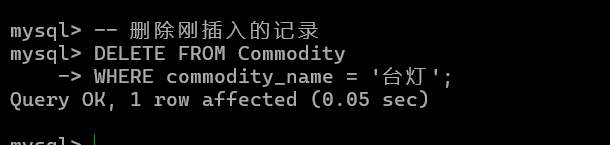
删除插入的结果代码：

-- 删除刚插入的记录

DELETE FROM Commodity

WHERE commodity\_name = '台灯';

删除结果截图：



# 5、建立存储过程及函数

## 5.1创建存储过程

创建一个存储过程，该存储过程实现输入某一个参数可以返回对应行上另外一个参数的值（如给定学号，返回选修课程信息等）。

代码：

-- 创建存储过程：输入商品ID，返回商品名称

DELIMITER $$

CREATE PROCEDURE GetCommodityName(IN input\_commodity\_id INT, OUT output\_commodity\_name VARCHAR(100))

BEGIN

SELECT commodity\_name INTO output\_commodity\_name

FROM Commodity

WHERE commodity\_id = input\_commodity\_id;

END $$

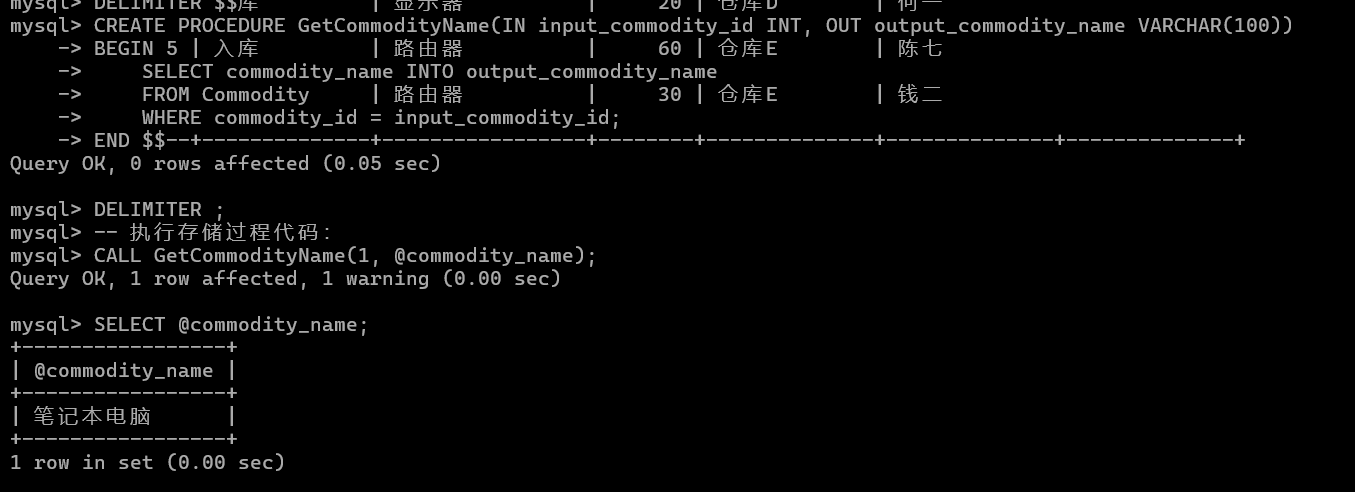
DELIMITER ;

-- 执行存储过程代码：

CALL GetCommodityName(1, @commodity\_name);

SELECT @commodity\_name;

结果截图：



## 5.2存储过程应用

创建一个函数过程，实现统计功能（统计个数或平均值等）。

代码：

-- 创建函数：统计某类商品的平均单价

DELIMITER $$

CREATE FUNCTION GetAveragePrice(category\_name VARCHAR(100))

RETURNS DECIMAL(10, 2)

DETERMINISTIC

BEGIN

DECLARE avg\_price DECIMAL(10, 2);

SELECT AVG(unit\_price) INTO avg\_price

FROM Commodity

WHERE category = category\_name;

RETURN avg\_price;

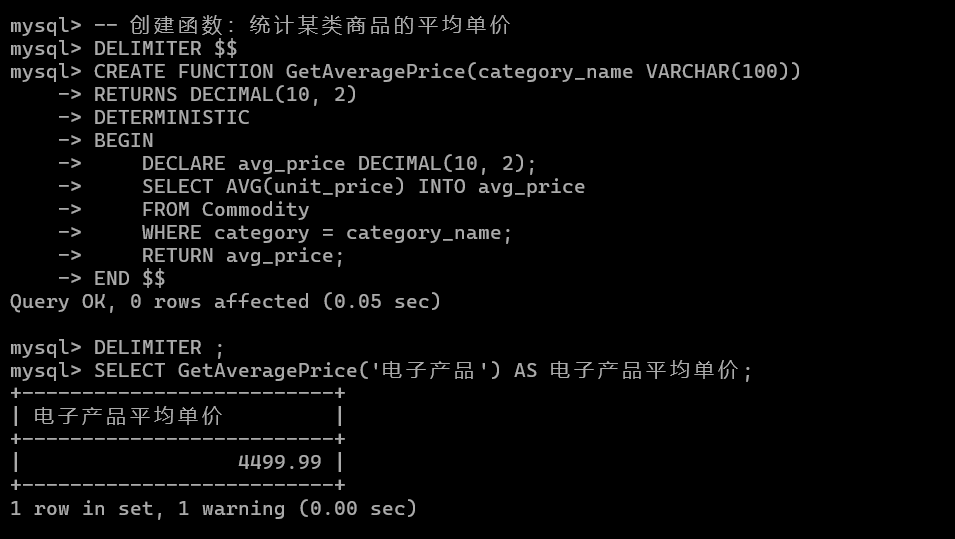
END $$

DELIMITER ;

执行函数过程代码：

SELECT GetAveragePrice('电子产品') AS 电子产品平均单价;

结果截图：



# 6、建立触发器

-- 创建触发器：在插入新的出入库记录时，记录日志

DELIMITER $$

CREATE TRIGGER AfterInsertInOutRecord

AFTER INSERT ON InOutRecord

FOR EACH ROW

BEGIN

INSERT INTO LogTable (log\_message, log\_date)

VALUES (CONCAT('新增出入库记录，ID: ', NEW.record\_id), NOW());

END $$

DELIMITER ;

-- 创建日志表

CREATE TABLE LogTable (

log\_id INT PRIMARY KEY AUTO\_INCREMENT,

log\_message TEXT NOT NULL,

log\_date DATETIME NOT NULL

);



# 7、项目设计总结

首先，仓库管理系统极大地提高了仓库管理的效率和准确性。通过系统的自动化记录和实时更新，我们能够准确地了解商品的入库和出库情况，及时掌握库存数量和位置，避免了传统手工管理中容易出现的信息不准确和漏记的问题。系统的查询功能也使得我们能够快速地获取所需信息，提高了工作效率。

其次，仓库管理系统帮助我们优化了库存管理和订单处理。系统能够根据订单需求自动分配库房和职工，提高了订单处理的效率和准确性，确保订单能够及时发货。同时，系统记录了商品的采购信息、销售信息和库存变动情况，我们可以通过系统进行进销存的分析和预测，优化采购和销售策略，减少了库存积压和缺货的情况。

最后，仓库管理系统的维护功能也很重要。系统提供了用户管理、权限管理、备份和恢复等功能，确保系统的安全性和稳定性。我们可以根据实际需求对系统进行定制和调整，满足企业的特定管理需求。

总的来说，仓库管理系统的使用极大地提高了仓库管理的效率和准确性，帮助我们优化了库存管理和订单处理，提高了工作效率和客户满意度。同时，系统的维护功能也确保了系统的安全性和稳定性。我对仓库管理系统的应用效果非常满意。