

CHANGZHOU INSTITUTE OF TECHNOLOGY

**数据库设计及应用实验报告**

|  |  |
| --- | --- |
| 课题名称： 仓库货物管理系统 | |
| 二级学院（直属学部）： | |
| 专 业： | 班级： |
| 学生姓名： | 学号： |

仓库货物管理系统

目 录

[仓库货物管理系统 2](#_Toc186662236)

[一、课程设计目的 4](#_Toc186662237)

[二、业务场景的调研与需求分析 5](#_Toc186662238)

[2.1. 业务场景调研 5](#_Toc186662239)

[2.2 需求分析 6](#_Toc186662240)

[三、数据模型的规划与设计 8](#_Toc186662241)

[3.1系统功能分析 8](#_Toc186662242)

[3.1.1 基本信息管理 8](#_Toc186662243)

[3.1.2 商品操作管理 8](#_Toc186662244)

[3.1.3 数据统计与查询 9](#_Toc186662245)

[3.2概念设计 9](#_Toc186662246)

[3.3逻辑设计 9](#_Toc186662247)

[四、数据库和表的创建与管理 11](#_Toc186662248)

[4.1 数据库和表的创建 11](#_Toc186662249)

[4.2 数据操纵 16](#_Toc186662250)

[4.3数据查询 21](#_Toc186662251)

[4.3.1 选择列 21](#_Toc186662252)

[4.3.2函数使用 21](#_Toc186662253)

[4.3.3 LIKE子句 22](#_Toc186662254)

[4.3.4 条件查询 22](#_Toc186662255)

[4.4.5多表查询 23](#_Toc186662256)

[4.4.6子查询 24](#_Toc186662257)

[4.4 数据视图 25](#_Toc186662258)

[4.5.1 定义视图 25](#_Toc186662259)

[4.5.2利用视图插入数据 26](#_Toc186662260)

[4.5 约束的创建 27](#_Toc186662261)

[4.5.1 主键约束 27](#_Toc186662262)

[4.5.2 外键约束 27](#_Toc186662263)

[4.5.3 唯一约束 28](#_Toc186662264)

[4.5.4 检查约束 28](#_Toc186662265)

[4.5.5 非空约束 29](#_Toc186662266)

[4.6 索引创建 29](#_Toc186662267)

[4.6.1 普通索引 29](#_Toc186662268)

[4.6.2 唯一索引 29](#_Toc186662269)

[4.6.3 组合索引 30](#_Toc186662270)

[4.6.4 全文索引 30](#_Toc186662271)

[4.7 数据库编程 30](#_Toc186662272)

[4.7.1创建存储过程 30](#_Toc186662273)

[4.7.2存储过程应用 31](#_Toc186662274)

[4.7.3建立触发器 33](#_Toc186662275)

[五、数据库管理 33](#_Toc186662276)

[5.1 用户与权限管理 33](#_Toc186662277)

[5.2 性能监控与优化 34](#_Toc186662278)

[5.3. 数据库备份 35](#_Toc186662279)

[5.3.1 完全备份 35](#_Toc186662280)

[5.3.2. 数据库运行维护方法 36](#_Toc186662281)

[六、总结（问题+方法+体会） 37](#_Toc186662282)

一、课程设计目的

本次仓库管理系统的课程设计旨在通过数据库设计和实现，完成对仓库管理中商品采购（进货）、销售（出货）、库存管理等业务流程的全面支持。具体设计目的包括以下方面：

**1. 数据规范化与集中管理**

* **目的**：将商品类别、供应商、业务员、仓库等基础数据统一存储并规范化管理。
* **意义**：提高数据管理的效率，减少冗余和重复录入，提高数据的一致性和完整性。

**2. 商品全生命周期管理**

* **目的**：实现商品从采购（进货）到销售（出货）的全过程管理，确保商品的每一个环节都能被追踪和记录。
* **意义**：为企业提供清晰的业务流，减少商品流转过程中的漏洞和管理混乱，提高业务透明度。

**3. 实现动态库存管理**

* **目的**：支持商品的入库、出库、转仓等操作，实现库存数据的实时更新。
* **意义**：
  + 保证库存信息的准确性，为业务决策提供可靠的数据支持。
  + 避免因库存信息滞后导致的缺货或积压问题。

**4. 提供统计与决策支持**

* **目的**：通过查询和统计功能，实现关键业务数据（如商品销量、库存变化、采购量、销售额等）的可视化与分析。
* **意义**：为管理者提供数据决策支持，帮助其优化商品采购和销售策略，提高仓库管理效率。

**5. 实现多用户权限管理**

* **目的**：支持不同角色（如仓库管理员、业务员、管理者）对系统的分权限操作。
* **意义**：确保系统安全性，避免未经授权的操作或敏感数据泄露。

**6. 强化学生数据库设计与实现能力**

* **目的**：通过完整的系统设计过程，让学生掌握从需求分析、逻辑设计到物理实现的全流程数据库开发技能。
* **意义**：提升数据库应用能力，为后续的工作实践打下扎实基础。

**具体功能设计的实现方向**

1. **基础数据管理**：包括商品类别、供应商信息、业务员信息、仓库信息等的管理功能，支持增删改查操作。
2. **商品流转管理**：
   * **入库管理**：记录商品的采购和入库情况。
   * **出库管理**：支持商品的销售出库，记录销售信息。
   * **转仓管理**：支持商品从一个仓库转移到另一个仓库，确保库存数据同步更新。
3. **统计功能**：
   * 库存查询：实时查询当前库存状态。
   * 销售统计：统计某段时间内的商品销量。
   * 采购统计：统计某段时间内的商品采购量。
4. **用户权限管理**：为不同用户分配不同权限，限制敏感操作的权限范围。

二、业务场景的调研与需求分析

## 2.1.业务场景调研

仓库管理系统是企业日常运营中不可或缺的一部分，涵盖了商品采购（进货）、销售（出货）和库存管理等核心业务流程。基于调研，仓库管理的核心业务场景可以划分如下：

1. **商品管理**
   * 企业需要管理多种商品，商品根据类别分类存储，以便于库存清点和查询。
   * 每种商品有唯一的标识，包含名称、类别、价格、单位、库存量等属性。
2. **供应商管理**
   * 商品的采购需要与供应商合作，每个供应商的详细信息（如名称、联系方式、地址等）需要被记录。
   * 一个商品可能由多个供应商提供，供应商的选择需要根据价格、信誉等因素进行管理。
3. **业务员管理**
   * 商品采购和销售由业务员负责，每位业务员需要记录其基本信息及对应的操作记录。
4. **仓库管理**
   * 企业通常有多个仓库用于商品存储，每个仓库需要记录其编号、名称、地址和容量。
   * 商品可能需要在仓库之间进行调拨，以满足业务需求。
5. **商品流转管理**
   * **入库管理**：商品采购后，业务员需要将其登记并入库，系统需要记录商品采购的详细信息。
   * **出库管理**：销售商品时，业务员需要将商品出库并减少库存。
   * **转仓管理**：需要将商品从一个仓库转移到另一个仓库，同时确保库存数据实时更新。
6. **库存管理**
   * 实时掌握商品的库存情况，防止库存不足或库存过剩问题。
   * 对商品的过期或损耗进行及时处理，确保库存数据的准确性。
7. **数据统计与分析**
   * 提供采购统计、销售统计、库存状态统计等功能。
   * 支持管理者通过查询和分析优化决策。

## 2.2 需求分析

基于业务场景调研，系统需求可分为以下几个部分：

**2.1 基础数据管理需求**

1. 商品信息管理
   * 每个商品有唯一编号，记录名称、类别、单价、库存量等信息。
2. 商品类别管理
   * 商品按照类别进行分类，类别信息可灵活扩展。
3. 供应商信息管理
   * 每个供应商有唯一编号，记录名称、联系方式、地址等信息。
4. 业务员信息管理
   * 记录业务员编号、姓名、联系方式等基本信息。
5. 仓库信息管理
   * 每个仓库有唯一编号，记录仓库名称、地址、容量等信息。

**2.2 业务流转管理需求**

1. **入库管理**
   * 商品采购后，由业务员负责登记入库，记录采购时间、采购数量、采购价格、供应商等信息。
   * 库存数据需要同步更新。
2. **出库管理**
   * 商品销售后，记录销售时间、销售数量、销售价格、客户信息等。
   * 库存数据需要同步更新，防止库存不足。
3. **转仓管理**
   * 商品从一个仓库转移到另一个仓库时，记录转仓时间、数量和操作人。
   * 源仓库和目标仓库的库存数据需要同步更新。

**2.3 库存管理需求**

1. **库存实时查询**
   * 支持按照商品名称、类别、仓库查询当前库存状态。
2. **库存预警**
   * 当库存量低于设定阈值时，系统发出警报，提醒及时补货。
3. **库存清单**
   * 按时间周期生成库存清单，帮助管理者进行盘点。

**2.4 数据统计需求**

1. **采购统计**
   * 按时间周期统计各类商品的采购量、采购金额。
   * 支持按照供应商统计采购数据。
2. **销售统计**
   * 按时间周期统计商品的销量、销售金额。
   * 支持按照业务员、商品类别统计销售数据。
3. **库存统计**
   * 提供实时库存统计，生成低库存商品清单。

**2.5 权限管理需求**

1. 支持多用户使用系统，不同用户角色（如管理员、仓库管理员、业务员）有不同权限。
2. 仓库管理员可以管理商品库存，业务员可以录入采购和销售信息，管理员有系统管理权限。

**3. 功能需求总结**

| **模块** | **功能描述** |
| --- | --- |
| 商品管理 | 商品信息增删改查、商品类别管理。 |
| 供应商管理 | 供应商信息增删改查、供应商与商品的关联管理。 |
| 业务员管理 | 业务员信息增删改查，记录业务员操作记录。 |
| 仓库管理 | 仓库信息增删改查、仓库容量管理、支持转仓操作。 |
| 入库管理 | 记录商品入库信息，更新库存数据。 |
| 出库管理 | 记录商品出库信息，更新库存数据。 |
| 转仓管理 | 记录商品转仓信息，更新两个仓库的库存数据。 |
| 数据统计与查询 | 提供采购、销售、库存统计及查询功能，支持多维度的筛选和统计。 |
| 权限管理 | 用户角色权限分配，不同角色操作不同模块（如管理员、仓库管理员、业务员）。 |

**4. 非功能需求**

1. **系统性能**
   * 支持多用户并发操作，查询响应时间控制在1秒以内。
2. **数据安全**
   * 提供权限控制，防止未授权用户访问敏感数据。
3. **数据备份**
   * 系统定期自动备份数据，防止意外数据丢失。
4. **扩展性**
   * 系统设计需支持未来功能扩展（如多仓库、多货币等）。

# 三、数据模型的规划与设计

## 3.1系统功能分析

### 3.1.1 基本信息管理

1. **商品类别管理**
   * 记录商品的类别信息，如家电、服装、食品等。
2. **供货商管理**
   * 管理供货商的基本信息，如名称、联系方式等。
3. **业务员管理**
   * 记录业务员的信息，如姓名、联系方式等。
4. **商品信息管理**
   * 管理商品的基本信息，如商品名称、单位（只、件、箱）、总库存等。
5. **仓库信息管理**
   * 管理仓库的信息，包括仓库名称和位置。
6. **仓库商品管理**
   * 记录商品在不同仓库的具体数量。

### 3.1.2 商品操作管理

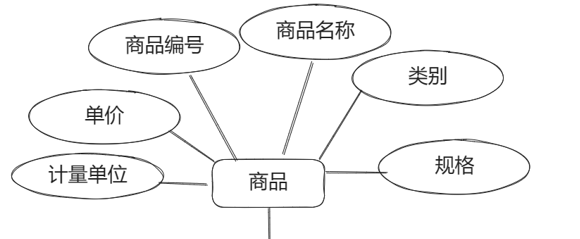
1. **商品入库管理**
   * 记录商品的进货信息，包括商品名称、供货商、入库数量、入库仓库、入库时间。
   * 入库时，自动增加商品的总库存和仓库对应商品的库存量（通过触发器实现）。
2. **商品销售出库管理**
   * 记录商品的销售出库信息，包括商品名称、业务员、出库数量、出库仓库、出库时间。
   * 出库时，自动减少商品的总库存和仓库对应商品的库存量（通过触发器实现）。
3. **商品转仓管理**
   * 实现商品在不同仓库之间的调拨操作，记录商品的转出仓库、转入仓库、商品数量及转仓日期。
   * 转仓时，自动更新转出仓库和转入仓库的商品数量（通过触发器实现）。

### 3.1.3 数据统计与查询

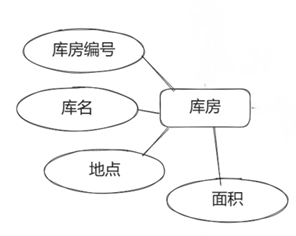
1. **商品库存查询**
   * 查询所有商品的总库存情况及每个仓库的具体库存量。
2. **商品进销统计**
   * 提供指定时间段内商品的进货数量和销售数量统计（通过存储过程实现）。
3. **报表查询**
   * 支持生成各类统计报表，如商品入库报表、销售出库报表、商品库存报表等。

## 3.2概念设计

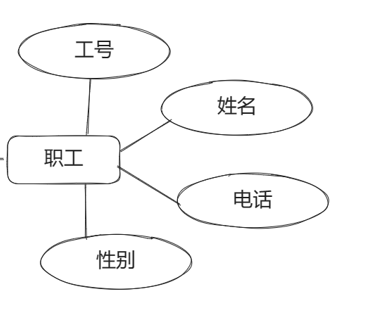
商品er图：



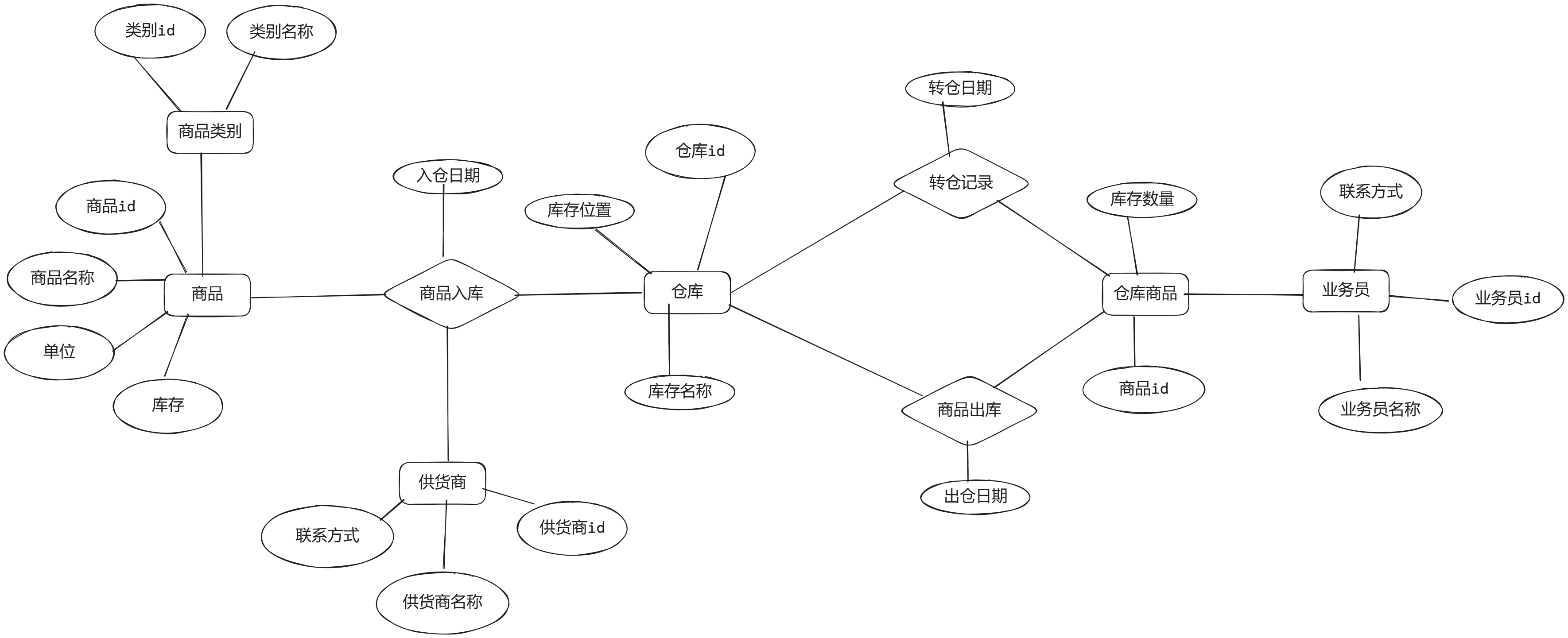
库房er图：



职工er图：



各实体及其联系的E-R图



## 3.3逻辑设计

**. 商品类别表 (ProductCategory)**

* **主键**: CategoryID
* **字段**:
  + CategoryID (类别ID, 自增, 主键)
  + CategoryName (类别名称, 非空)

**2. 供货商表 (Supplier)**

* **主键**: SupplierID
* **字段**:
  + SupplierID (供货商ID, 自增, 主键)
  + SupplierName (供货商名称, 非空)
  + Contact (联系方式)

**3. 业务员信息表 (Salesperson)**

* **主键**: SalespersonID
* **字段**:
  + SalespersonID (业务员ID, 自增, 主键)
  + Name (业务员姓名, 非空)
  + Phone (联系电话)

**4. 商品信息表 (Product)**

* **主键**: ProductID
* **外键**: CategoryID → ProductCategory(CategoryID)
* **字段**:
  + ProductID (商品ID, 自增, 主键)
  + ProductName (商品名称, 非空)
  + CategoryID (类别ID, 外键)
  + Unit (单位, 非空，规则限制：'只'、'件'、'箱')
  + TotalStock (总库存, 默认值0, 非负约束)

**5. 仓库表 (Warehouse)**

* **主键**: WarehouseID
* **字段**:
  + WarehouseID (仓库ID, 自增, 主键)
  + WarehouseName (仓库名称, 非空)
  + Location (仓库位置)

**6. 仓库商品表 (WarehouseProduct)**

* **主键**: WarehouseID 和 ProductID (复合主键)
* **外键**:
  + WarehouseID → Warehouse(WarehouseID)
  + ProductID → Product(ProductID)
* **字段**:
  + WarehouseID (仓库ID, 外键)
  + ProductID (商品ID, 外键)
  + Quantity (库存数量, 默认0, 非空)

**7. 商品入库表 (ProductInbound)**

* **主键**: InboundID
* **外键**:
  + ProductID → Product(ProductID)
  + SupplierID → Supplier(SupplierID)
  + WarehouseID → Warehouse(WarehouseID)
* **字段**:
  + InboundID (入库ID, 自增, 主键)
  + ProductID (商品ID, 外键)
  + SupplierID (供货商ID, 外键)
  + Quantity (入库数量, 非空)
  + WarehouseID (仓库ID, 外键)
  + InboundDate (入库日期, 非空)

**8. 商品销售出库表 (ProductOutbound)**

* **主键**: OutboundID
* **外键**:
  + ProductID → Product(ProductID)
  + SalespersonID → Salesperson(SalespersonID)
  + WarehouseID → Warehouse(WarehouseID)
* **字段**:
  + OutboundID (出库ID, 自增, 主键)
  + ProductID (商品ID, 外键)
  + SalespersonID (业务员ID, 外键)
  + Quantity (出库数量, 非空)
  + WarehouseID (仓库ID, 外键)
  + OutboundDate (出库日期, 非空)

**9. 转仓记录表 (TransferLog)**

* **主键**: TransferID
* **外键**:
  + ProductID → Product(ProductID)
  + FromWarehouseID → Warehouse(WarehouseID)
  + ToWarehouseID → Warehouse(WarehouseID)
* **字段**:
  + TransferID (转仓记录ID, 自增, 主键)
  + ProductID (商品ID, 外键)
  + FromWarehouseID (转出仓库ID, 外键)
  + ToWarehouseID (转入仓库ID, 外键)
  + Quantity (转仓数量, 必须 > 0)
  + TransferDate (转仓日期, 非空)

四、数据库和表的创建与管理

## 4.1 数据库和表的创建

-- 创建数据库

CREATE DATABASE store\_management;

-- 使用数据库

USE store\_management;

-- 2. 创建基本信息表

-- 商品类别表

CREATE TABLE ProductCategory (

CategoryID INT AUTO\_INCREMENT PRIMARY KEY,

CategoryName VARCHAR(100) NOT NULL

);

-- 供货商表

CREATE TABLE Supplier (

SupplierID INT AUTO\_INCREMENT PRIMARY KEY,

SupplierName VARCHAR(100) NOT NULL,

Contact VARCHAR(50)

);

-- 业务员信息表

CREATE TABLE Salesperson (

SalespersonID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Phone VARCHAR(15)

);

-- 3. 商品信息与仓库管理

-- 商品信息表

CREATE TABLE Product (

ProductID INT AUTO\_INCREMENT PRIMARY KEY,

ProductName VARCHAR(100) NOT NULL,

CategoryID INT NOT NULL,

Unit ENUM('只', '件', '箱') NOT NULL,

TotalStock INT DEFAULT 0 CHECK (TotalStock >= 0),

FOREIGN KEY (CategoryID) REFERENCES ProductCategory(CategoryID)

);

-- 创建仓库表

CREATE TABLE Warehouse (

WarehouseID INT AUTO\_INCREMENT PRIMARY KEY,

WarehouseName VARCHAR(100) NOT NULL,

Location VARCHAR(100)

);

-- 仓库商品表

CREATE TABLE WarehouseProduct (

WarehouseID INT NOT NULL,

ProductID INT NOT NULL,

Quantity INT NOT NULL DEFAULT 0,

PRIMARY KEY (WarehouseID, ProductID),

FOREIGN KEY (WarehouseID) REFERENCES Warehouse(WarehouseID),

FOREIGN KEY (ProductID) REFERENCES Product(ProductID)

);

-- 4. 商品入库与出库管理

-- 商品入库表

CREATE TABLE ProductInbound (

InboundID INT AUTO\_INCREMENT PRIMARY KEY,

ProductID INT NOT NULL,

SupplierID INT NOT NULL,

Quantity INT NOT NULL,

WarehouseID INT NOT NULL,

InboundDate DATE NOT NULL,

FOREIGN KEY (ProductID) REFERENCES Product(ProductID),

FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID),

FOREIGN KEY (WarehouseID) REFERENCES Warehouse(WarehouseID)

);

-- 商品销售出库表

CREATE TABLE ProductOutbound (

OutboundID INT AUTO\_INCREMENT PRIMARY KEY,

ProductID INT NOT NULL,

SalespersonID INT NOT NULL,

Quantity INT NOT NULL,

WarehouseID INT NOT NULL,

OutboundDate DATE NOT NULL,

FOREIGN KEY (ProductID) REFERENCES Product(ProductID),

FOREIGN KEY (SalespersonID) REFERENCES Salesperson(SalespersonID),

FOREIGN KEY (WarehouseID) REFERENCES Warehouse(WarehouseID)

);

-- 转仓记录表

CREATE TABLE TransferLog (

TransferID INT AUTO\_INCREMENT PRIMARY KEY,

ProductID INT NOT NULL,

FromWarehouseID INT NOT NULL,

ToWarehouseID INT NOT NULL,

Quantity INT NOT NULL CHECK (Quantity > 0),

TransferDate DATE NOT NULL,

FOREIGN KEY (ProductID) REFERENCES Product(ProductID),

FOREIGN KEY (FromWarehouseID) REFERENCES Warehouse(WarehouseID),

FOREIGN KEY (ToWarehouseID) REFERENCES Warehouse(WarehouseID)

);

-- 5. 入库触发器：增加库存

DELIMITER //

CREATE TRIGGER trg\_ProductInbound

AFTER INSERT ON ProductInbound

FOR EACH ROW

BEGIN

-- 更新 Product 表的总库存

UPDATE Product

SET TotalStock = TotalStock + NEW.Quantity

WHERE ProductID = NEW.ProductID;

-- 更新 WarehouseProduct 表的库存数量

UPDATE WarehouseProduct

SET Quantity = Quantity + NEW.Quantity

WHERE WarehouseID = NEW.WarehouseID AND ProductID = NEW.ProductID;

-- 如果 WarehouseProduct 中不存在对应记录，则插入新记录

INSERT INTO WarehouseProduct (WarehouseID, ProductID, Quantity)

SELECT NEW.WarehouseID, NEW.ProductID, NEW.Quantity

WHERE NOT EXISTS (

SELECT 1

FROM WarehouseProduct

WHERE WarehouseID = NEW.WarehouseID AND ProductID = NEW.ProductID

);

END;

//

DELIMITER ;

-- 6. 出库触发器：减少库存

DELIMITER //

CREATE TRIGGER trg\_ProductOutbound

AFTER INSERT ON ProductOutbound

FOR EACH ROW

BEGIN

-- 更新 Product 表的总库存

UPDATE Product

SET TotalStock = TotalStock - NEW.Quantity

WHERE ProductID = NEW.ProductID;

-- 更新 WarehouseProduct 表的库存数量

UPDATE WarehouseProduct

SET Quantity = Quantity - NEW.Quantity

WHERE WarehouseID = NEW.WarehouseID AND ProductID = NEW.ProductID;

END;

//

DELIMITER ;

-- 7. 转仓触发器

DELIMITER //

CREATE TRIGGER trg\_TransferWarehouse

AFTER INSERT ON TransferLog

FOR EACH ROW

BEGIN

-- 更新转出仓库：减少商品数量

UPDATE WarehouseProduct

SET Quantity = Quantity - NEW.Quantity

WHERE WarehouseID = NEW.FromWarehouseID AND ProductID = NEW.ProductID;

-- 更新转入仓库：增加商品数量

UPDATE WarehouseProduct

SET Quantity = Quantity + NEW.Quantity

WHERE WarehouseID = NEW.ToWarehouseID AND ProductID = NEW.ProductID;

-- 如果转入仓库中不存在对应记录，则插入新记录

INSERT INTO WarehouseProduct (WarehouseID, ProductID, Quantity)

SELECT NEW.ToWarehouseID, NEW.ProductID, NEW.Quantity

WHERE NOT EXISTS (

SELECT 1

FROM WarehouseProduct

WHERE WarehouseID = NEW.ToWarehouseID AND ProductID = NEW.ProductID

);

END;

//

DELIMITER ;

-- 8. 统计存储过程：统计指定时间段内各种商品的进货和销售数量

DELIMITER //

CREATE PROCEDURE sp\_Statistics(IN StartDate DATE, IN EndDate DATE)

BEGIN

SELECT

p.ProductID AS 商品ID,

p.ProductName AS 商品名称,

IFNULL(SUM(CASE WHEN pi.InboundDate BETWEEN StartDate AND EndDate THEN pi.Quantity ELSE 0 END), 0) AS 进货数量,

IFNULL(SUM(CASE WHEN po.OutboundDate BETWEEN StartDate AND EndDate THEN po.Quantity ELSE 0 END), 0) AS 销售数量

FROM Product p

LEFT JOIN ProductInbound pi ON p.ProductID = pi.ProductID

LEFT JOIN ProductOutbound po ON p.ProductID = po.ProductID

GROUP BY p.ProductID, p.ProductName

ORDER BY p.ProductID;

END;

//

DELIMITER ;

## 4.2 数据操纵

-- 暂时禁用外键约束SET FOREIGN\_KEY\_CHECKS = 0;

INSERT INTO ProductCategory (CategoryName) VALUES

('电子产品'),

('家用电器'),

('服装'),

('食品'),

('玩具'),

('文具'),

('化妆品'),

('家具'),

('运动用品'),

('汽车配件'),

('书籍'),

('医药用品'),

('建筑材料'),

('五金工具'),

('饰品'),

('农产品'),

('电脑配件'),

('日用品'),

('手机配件'),

('清洁用品');

INSERT INTO Supplier (SupplierName, Contact) VALUES

('供应商A', '1234567890'),

('供应商B', '0987654321'),

('供应商C', '1231231234'),

('供应商D', '3213213214'),

('供应商E', '1112223334'),

('供应商F', '4445556667'),

('供应商G', '7778889990'),

('供应商H', '6665554443'),

('供应商I', '3332221111'),

('供应商J', '9998887770'),

('供应商K', '1011121314'),

('供应商L', '1516171819'),

('供应商M', '2021222324'),

('供应商N', '2526272829'),

('供应商O', '3031323334'),

('供应商P', '3536373839'),

('供应商Q', '4041424344'),

('供应商R', '4546474849'),

('供应商S', '5051525354'),

('供应商T', '5556575859');

INSERT INTO Salesperson (Name, Phone) VALUES

('业务员A', '1111111111'),

('业务员B', '2222222222'),

('业务员C', '3333333333'),

('业务员D', '4444444444'),

('业务员E', '5555555555'),

('业务员F', '6666666666'),

('业务员G', '7777777777'),

('业务员H', '8888888888'),

('业务员I', '9999999999'),

('业务员J', '1010101010'),

('业务员K', '1212121212'),

('业务员L', '1313131313'),

('业务员M', '1414141414'),

('业务员N', '1515151515'),

('业务员O', '1616161616'),

('业务员P', '1717171717'),

('业务员Q', '1818181818'),

('业务员R', '1919191919'),

('业务员S', '2020202020'),

('业务员T', '2121212121');

INSERT INTO Product (ProductName, CategoryID, Unit, TotalStock) VALUES

('手机', 1, '件', 50),

('电视', 2, '件', 30),

('衣服', 3, '件', 100),

('零食', 4, '箱', 200),

('玩具车', 5, '只', 150),

('钢笔', 6, '只', 500),

('口红', 7, '件', 80),

('沙发', 8, '件', 20),

('篮球', 9, '只', 60),

('刹车片', 10, '件', 40),

('小说', 11, '件', 300),

('维生素', 12, '件', 120),

('水泥', 13, '件', 50),

('螺丝刀', 14, '件', 100),

('耳环', 15, '件', 70),

('小麦', 16, '箱', 1000),

('键盘', 17, '件', 120),

('洗衣液', 18, '箱', 80),

('充电器', 19, '件', 200),

('清洁剂', 20, '箱', 60);

INSERT INTO Warehouse (WarehouseName, Location) VALUES

('仓库A', '北京'),

('仓库B', '上海'),

('仓库C', '广州'),

('仓库D', '深圳'),

('仓库E', '杭州'),

('仓库F', '南京'),

('仓库G', '成都'),

('仓库H', '武汉'),

('仓库I', '重庆'),

('仓库J', '西安'),

('仓库K', '天津'),

('仓库L', '郑州'),

('仓库M', '长沙'),

('仓库N', '苏州'),

('仓库O', '青岛'),

('仓库P', '济南'),

('仓库Q', '福州'),

('仓库R', '厦门'),

('仓库S', '南昌'),

('仓库T', '合肥');

INSERT INTO WarehouseProduct (WarehouseID, ProductID, Quantity) VALUES

(1, 1, 10),

(1, 2, 15),

(1, 3, 20),

(2, 4, 30),

(2, 5, 25),

(3, 6, 40),

(3, 7, 15),

(4, 8, 10),

(4, 9, 20),

(5, 10, 5),

(5, 11, 35),

(6, 12, 25),

(6, 13, 50),

(7, 14, 20),

(7, 15, 30),

(8, 16, 200),

(8, 17, 60),

(9, 18, 40),

(9, 19, 50),

(10, 20, 30);

INSERT INTO ProductInbound (ProductID, SupplierID, Quantity, WarehouseID, InboundDate) VALUES

(1, 1, 10, 1, '2025-01-01'),

(2, 2, 15, 1, '2025-01-02'),

(3, 3, 20, 1, '2025-01-03'),

(4, 4, 30, 2, '2025-01-04'),

(5, 5, 25, 2, '2025-01-05'),

(6, 6, 40, 3, '2025-01-06'),

(7, 7, 15, 3, '2025-01-07'),

(8, 8, 10, 4, '2025-01-08'),

(9, 9, 20, 4, '2025-01-09'),

(10, 10, 5, 5, '2025-01-10'),

(11, 11, 35, 5, '2025-01-11'),

(12, 12, 25, 6, '2025-01-12'),

(13, 13, 50, 6, '2025-01-13'),

(14, 14, 20, 7, '2025-01-14'),

(15, 15, 30, 7, '2025-01-15'),

(16, 16, 200, 8, '2025-01-16'),

(17, 17, 60, 8, '2025-01-17'),

(18, 18, 40, 9, '2025-01-18'),

(19, 19, 50, 9, '2025-01-19'),

(20, 20, 30, 10, '2025-01-20');

INSERT INTO ProductOutbound (ProductID, SalespersonID, Quantity, WarehouseID, OutboundDate) VALUES

(1, 1, 5, 1, '2025-01-21'),

(2, 2, 10, 1, '2025-01-22'),

(3, 3, 15, 1, '2025-01-23'),

(4, 4, 20, 2, '2025-01-24'),

(5, 5, 10, 2, '2025-01-25'),

(6, 6, 15, 3, '2025-01-26'),

(7, 7, 5, 3, '2025-01-27'),

(8, 8, 8, 4, '2025-01-28'),

(9, 9, 12, 4, '2025-01-29'),

(10, 10, 3, 5, '2025-01-30'),

(11, 11, 18, 5, '2025-01-31'),

(12, 12, 10, 6, '2025-02-01'),

(13, 13, 25, 6, '2025-02-02'),

(14, 14, 10, 7, '2025-02-03'),

(15, 15, 15, 7, '2025-02-04'),

(16, 16, 100, 8, '2025-02-05'),

(17, 17, 40, 8, '2025-02-06'),

(18, 18, 20, 9, '2025-02-07'),

(19, 19, 25, 9, '2025-02-08'),

(20, 20, 20, 10, '2025-02-09');

INSERT INTO TransferLog (ProductID, FromWarehouseID, ToWarehouseID, Quantity, TransferDate) VALUES

(1, 1, 2, 5, '2025-02-01'),

(2, 2, 3, 10, '2025-02-02'),

(3, 3, 4, 15, '2025-02-03'),

(4, 4, 5, 20, '2025-02-04'),

(5, 5, 6, 25, '2025-02-05'),

(6, 6, 7, 30, '2025-02-06'),

(7, 7, 8, 10, '2025-02-07'),

(8, 8, 9, 12, '2025-02-08'),

(9, 9, 10, 15, '2025-02-09'),

(10, 10, 1, 20, '2025-02-10'),

(11, 1, 2, 8, '2025-02-11'),

(12, 2, 3, 12, '2025-02-12'),

(13, 3, 4, 18, '2025-02-13'),

(14, 4, 5, 22, '2025-02-14'),

(15, 5, 6, 25, '2025-02-15'),

(16, 6, 7, 30, '2025-02-16'),

(17, 7, 8, 35, '2025-02-17'),

(18, 8, 9, 40, '2025-02-18'),

(19, 9, 10, 45, '2025-02-19'),

(20, 10, 1, 50, '2025-02-20');

## 4.3 数据查询

### 4.3.1 选择列

代码：

-- 查询商品表的名称、分类和单价，给出别名

SELECT

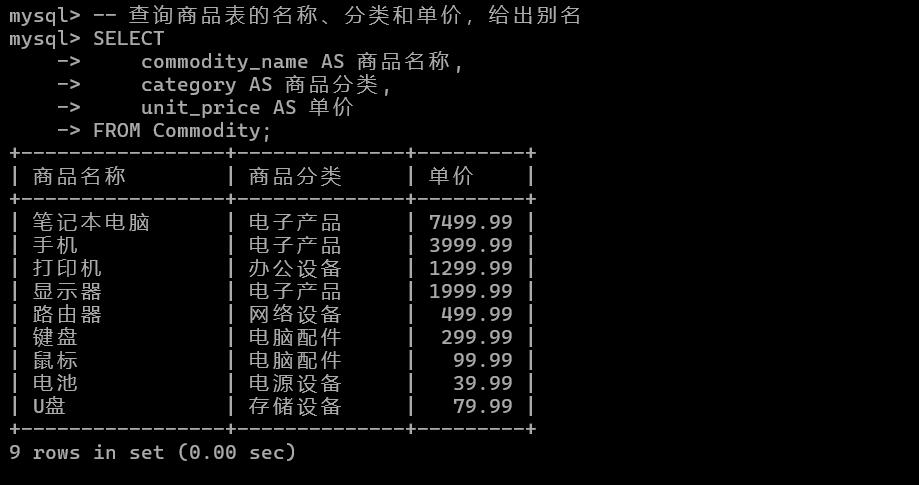
commodity\_name AS 商品名称,

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,

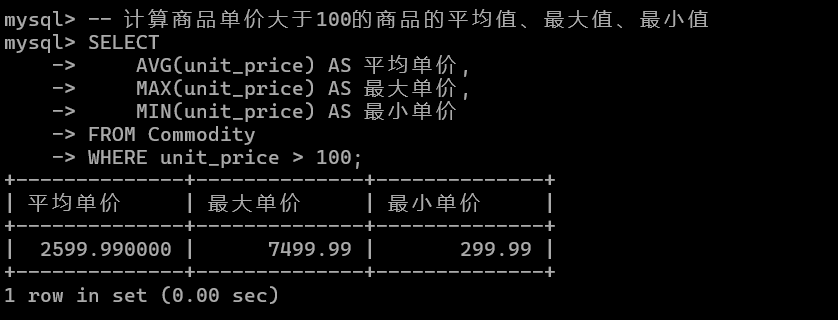
MAX(unit\_price) AS 最大单价,

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：

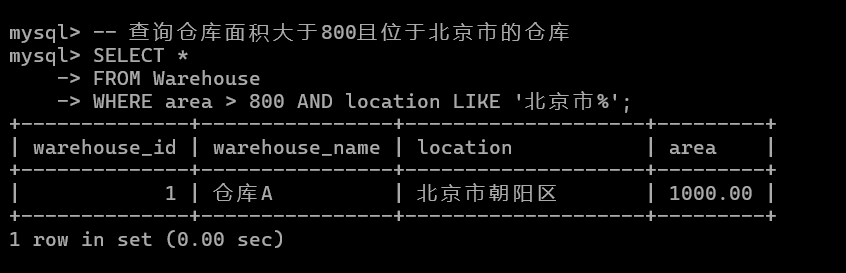
-- 查询仓库面积大于800且位于北京市的仓库

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,

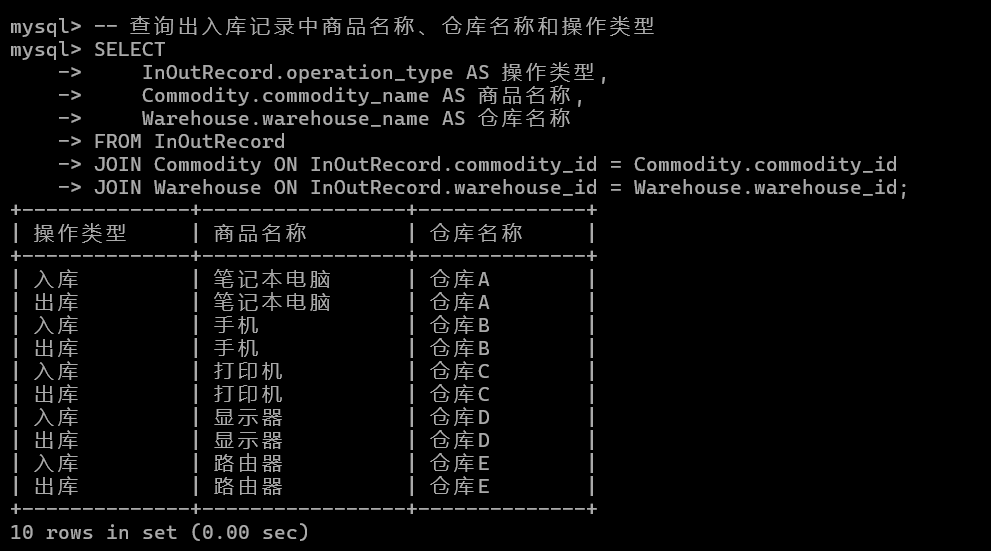
Warehouse.warehouse\_name AS 仓库名称

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：

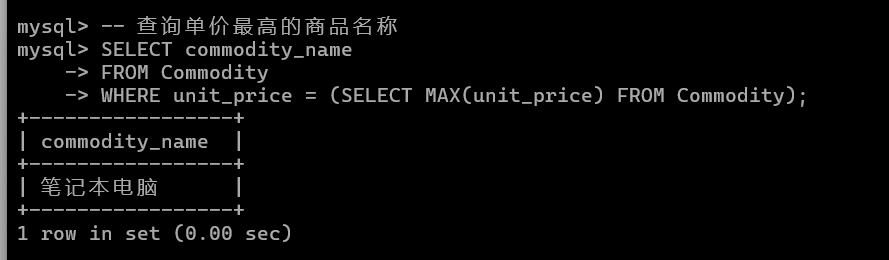
-- 查询单价最高的商品名称

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：

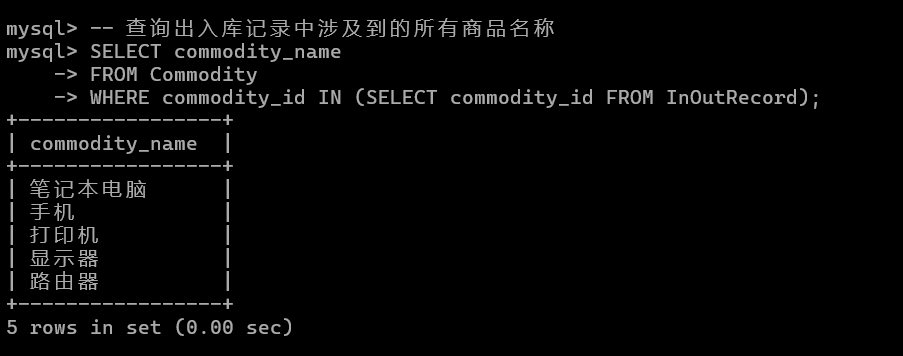
-- 查询出入库记录中涉及到的所有商品名称

SELECT commodity\_name

FROM Commodity

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

结果截图：



## 4.4 数据视图

建立一个用户user01，给user01账户授予建立视图的权限。

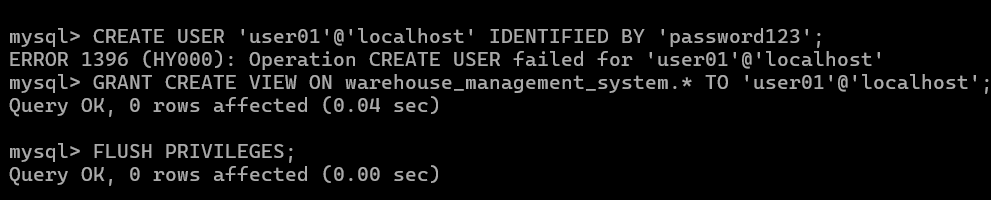
代码：

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

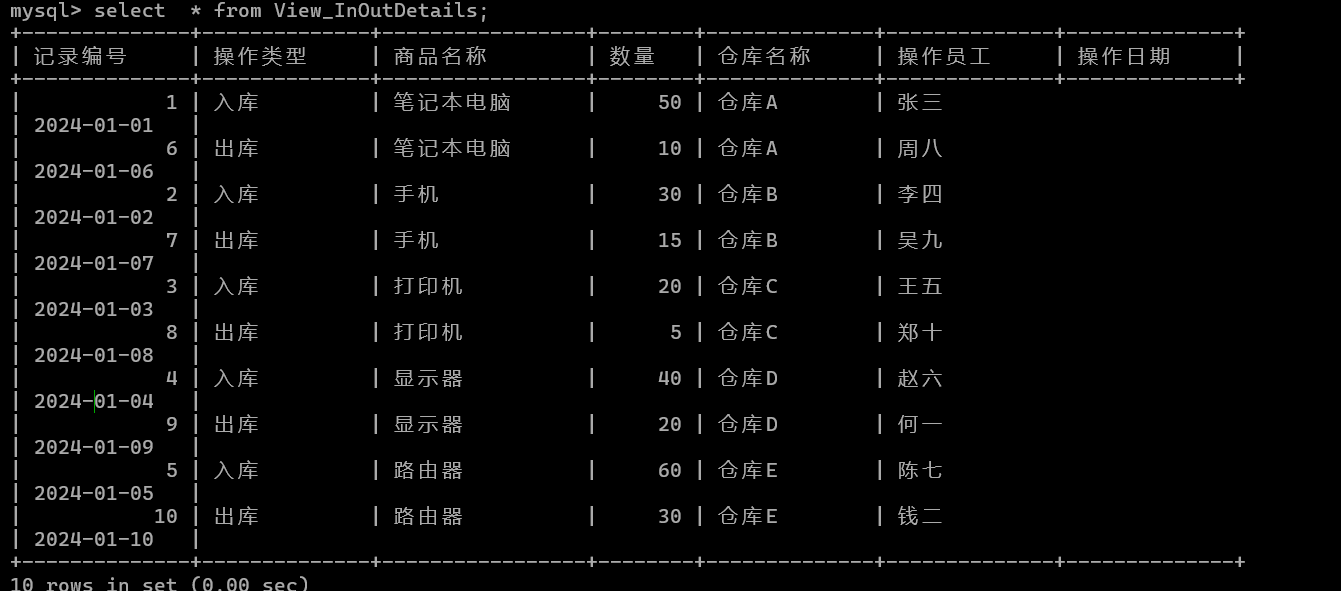
JOIN Warehouse ON InOutRecord.warehouse\_id = Warehouse.warehouse\_id

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：

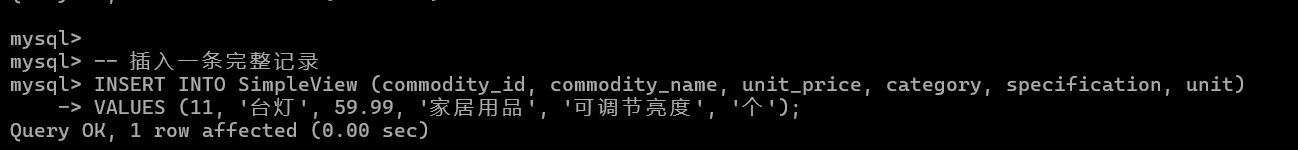
-- 查询刚插入的记录

SELECT \*

FROM Commodity

WHERE commodity\_name = '台灯';

查询结果截图：



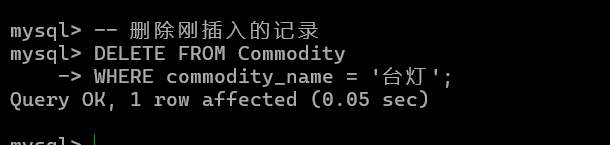
删除插入的结果代码：

-- 删除刚插入的记录

DELETE FROM Commodity

WHERE commodity\_name = '台灯';

删除结果截图：



## 4.5 约束的创建

### 4.5.1 主键约束

主键约束用于确保表中的记录具有唯一标识，主键字段不能包含 NULL 值。

-- 创建表时定义主键约束

CREATE TABLE ProductCategory (

CategoryID INT AUTO\_INCREMENT,

CategoryName VARCHAR(100) NOT NULL,

PRIMARY KEY (CategoryID)

);

-- 或在表创建后添加主键约束

ALTER TABLE ProductCategory

ADD PRIMARY KEY (CategoryID);

### 4.5.2 外键约束

外键约束用于确保引用字段的值必须存在于另一张表的主键或唯一键中。

-- 创建表时定义外键约束

CREATE TABLE Product (

ProductID INT AUTO\_INCREMENT,

ProductName VARCHAR(100) NOT NULL,

CategoryID INT NOT NULL,

PRIMARY KEY (ProductID),

FOREIGN KEY (CategoryID) REFERENCES ProductCategory(CategoryID)

);

-- 或在表创建后添加外键约束

ALTER TABLE Product

ADD CONSTRAINT fk\_Product\_Category

FOREIGN KEY (CategoryID) REFERENCES ProductCategory(CategoryID);

### 4.5.3 唯一约束

唯一约束用于确保某字段的值在表中是唯一的。

-- 创建表时定义唯一约束

CREATE TABLE Supplier (

SupplierID INT AUTO\_INCREMENT,

SupplierName VARCHAR(100) NOT NULL,

Contact VARCHAR(50),

UNIQUE (SupplierName)

);

-- 或在表创建后添加唯一约束

ALTER TABLE Supplier

ADD CONSTRAINT uq\_Supplier\_SupplierName

UNIQUE (SupplierName);

### 4.5.4 检查约束

检查约束用于确保字段的值满足特定条件。

-- 创建表时定义检查约束

CREATE TABLE Product (

ProductID INT AUTO\_INCREMENT,

ProductName VARCHAR(100) NOT NULL,

TotalStock INT NOT NULL DEFAULT 0,

PRIMARY KEY (ProductID),

CHECK (TotalStock >= 0) -- 确保库存不能为负

);

-- 或在表创建后添加检查约束

ALTER TABLE Product

ADD CONSTRAINT chk\_Product\_TotalStock

CHECK (TotalStock >= 0);

### 4.5.5 非空约束

非空约束用于确保字段的值不能为空。

-- 创建表时定义非空约束

CREATE TABLE Salesperson (

SalespersonID INT AUTO\_INCREMENT,

Name VARCHAR(100) NOT NULL, -- 非空约束

Phone VARCHAR(15),

PRIMARY KEY (SalespersonID)

);

## 4.6 索引创建

索引用于加速查询性能。可以在一个或多个字段上创建索引。

### 4.6.1 普通索引

普通索引可以加速查询，但不强制唯一性。

-- 在创建表时定义索引

CREATE TABLE Product (

ProductID INT AUTO\_INCREMENT,

ProductName VARCHAR(100) NOT NULL,

CategoryID INT,

PRIMARY KEY (ProductID),

INDEX idx\_ProductName (ProductName) -- 在ProductName列上创建索引

);

-- 或在表创建后添加索引

CREATE INDEX idx\_ProductName ON Product (ProductName);

### 4.6.2 唯一索引

唯一索引确保字段值在表中是唯一的。

-- 在创建表时定义唯一索引

CREATE TABLE Supplier (

SupplierID INT AUTO\_INCREMENT,

SupplierName VARCHAR(100) NOT NULL,

PRIMARY KEY (SupplierID),

UNIQUE INDEX uq\_SupplierName (SupplierName) -- 唯一索引

);

-- 或在表创建后添加唯一索引

CREATE UNIQUE INDEX uq\_SupplierName ON Supplier (SupplierName);

### 4.6.3 组合索引

组合索引用于多个字段的联合查询。

-- 在创建表时定义组合索引

CREATE TABLE ProductOutbound (

OutboundID INT AUTO\_INCREMENT,

ProductID INT NOT NULL,

SalespersonID INT NOT NULL,

OutboundDate DATE NOT NULL,

PRIMARY KEY (OutboundID),

INDEX idx\_ProductID\_OutboundDate (ProductID, OutboundDate) -- 在ProductID和OutboundDate上创建组合索引

);

-- 或在表创建后添加组合索引

CREATE INDEX idx\_ProductID\_OutboundDate ON ProductOutbound (ProductID, OutboundDate);

### 4.6.4 全文索引

用于对大文本字段（如 TEXT 或 VARCHAR）进行全文搜索。

-- 创建表时定义全文索引

CREATE TABLE Documents (

DocumentID INT AUTO\_INCREMENT,

Title VARCHAR(255),

Content TEXT,

PRIMARY KEY (DocumentID),

FULLTEXT INDEX ft\_Content (Content) -- 全文索引

);

-- 或在表创建后添加全文索引

CREATE FULLTEXT INDEX ft\_Content ON Documents (Content);

## 4.7 数据库编程

### 4.7.1 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 $$

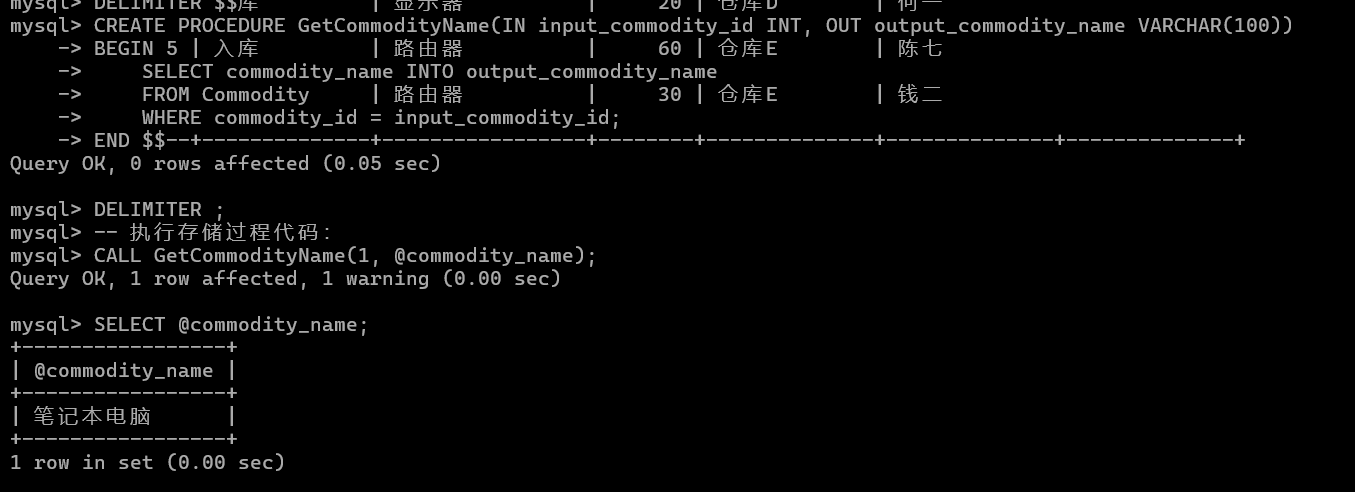
DELIMITER ;

-- 执行存储过程代码：

CALL GetCommodityName(1, @commodity\_name);

SELECT @commodity\_name;

结果截图：



### 4.7.2 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;

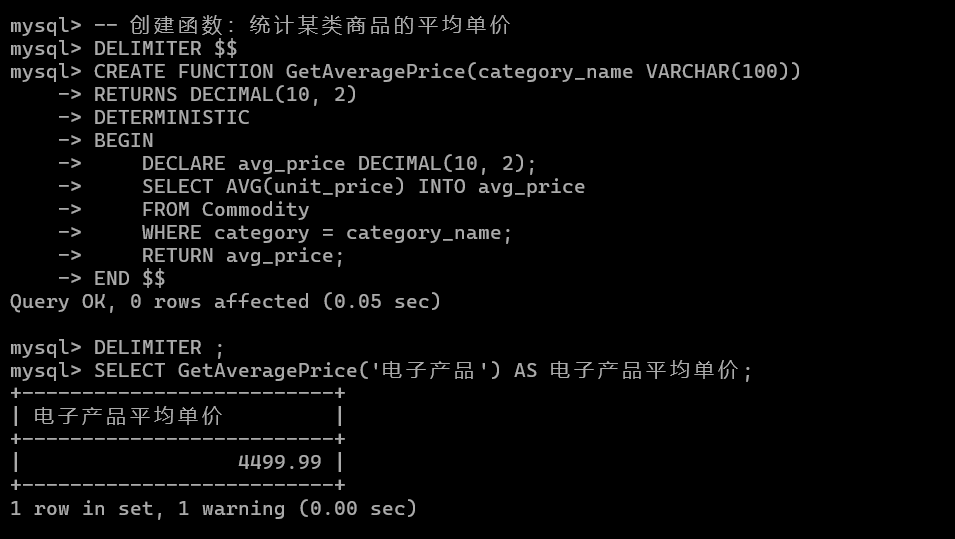
END $$

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：



统计指定商品的进货总量和销售总量

该存储过程接收一个商品 ID，统计该商品的进货总量和销售总量，并返回结果。

DELIMITER //

CREATE PROCEDURE GetProductStatistics(

IN ProductID INT

)

BEGIN

SELECT

p.ProductName AS 商品名称,

IFNULL(SUM(pi.Quantity), 0) AS 总进货量,

IFNULL(SUM(po.Quantity), 0) AS 总销售量

FROM Product p

LEFT JOIN ProductInbound pi ON p.ProductID = pi.ProductID

LEFT JOIN ProductOutbound po ON p.ProductID = po.ProductID

WHERE p.ProductID = ProductID

GROUP BY p.ProductName;

END;

//

DELIMITER ;

CALL GetProductStatistics(1);



### 4.7.3 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

);



# 五、数据库管理

## 5.1 用户与权限管理

* 定义不同的用户角色，赋予对应的权限，确保数据安全。
* 使用 GRANT 和 REVOKE 语句管理用户权限。

**示例**

-- 创建用户

CREATE USER 'db\_user'@'localhost' IDENTIFIED BY 'password';

-- 授予权限

GRANT SELECT, INSERT, UPDATE, DELETE ON store\_management.\* TO 'db\_user'@'localhost';

-- 撤销权限

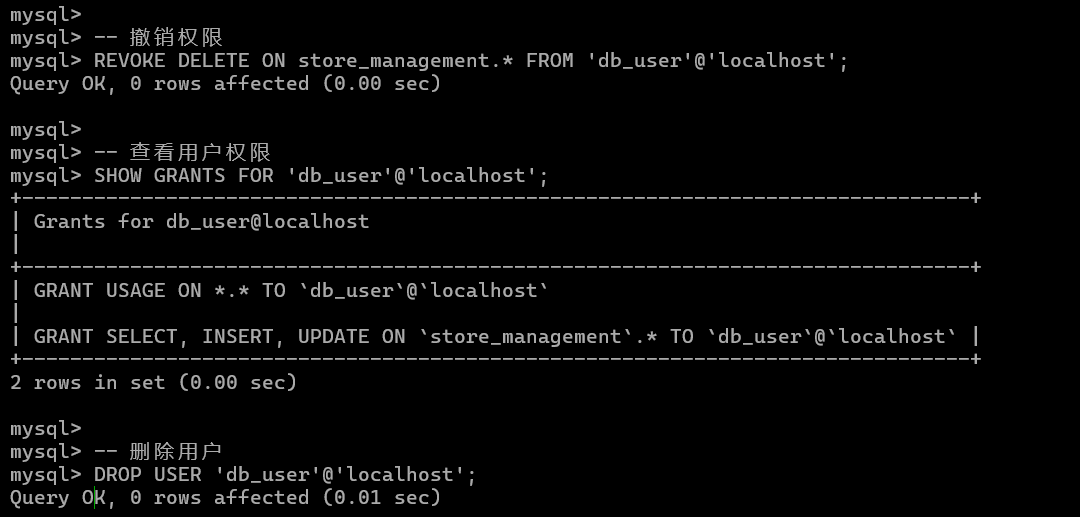
REVOKE DELETE ON store\_management.\* FROM 'db\_user'@'localhost';

-- 查看用户权限

SHOW GRANTS FOR 'db\_user'@'localhost';

-- 删除用户

DROP USER 'db\_user'@'localhost';



## 5.2 性能监控与优化

* 定期检查慢查询日志，优化索引和查询语句。
* 使用 EXPLAIN 分析查询执行计划。

**慢查询日志配置**

在 my.cnf 配置文件中启用慢查询日志：

[mysqld]

slow\_query\_log = 1

slow\_query\_log\_file = /var/log/mysql/slow.log

long\_query\_time = 2

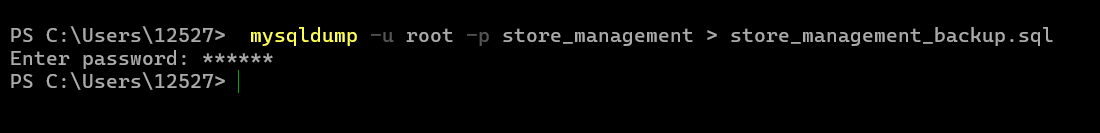
## 5.3. 数据库备份

### 5.3.1 完全备份

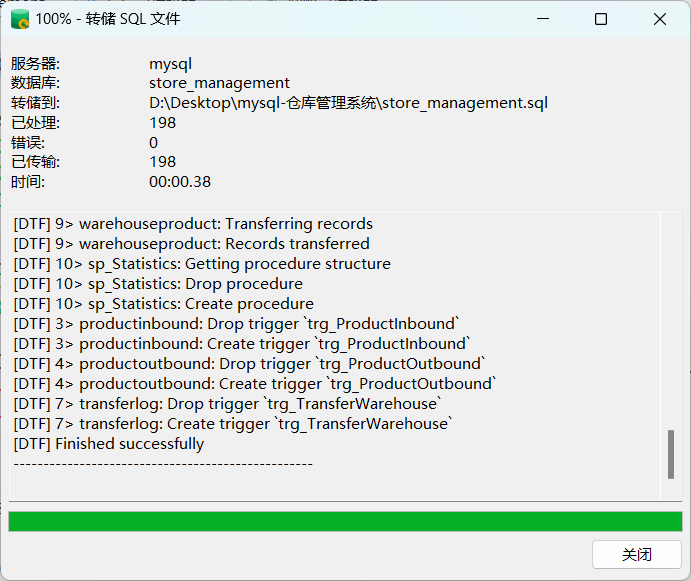
完全备份包含所有表数据和结构，适合定期备份整个数据库。

1、使用 mysqldump 工具

mysqldump -u root -p store\_management > store\_management\_backup.sql



2、使用navicat工具导出



3. 数据库恢复

3.1 从完全备份恢复

将备份的 .sql 文件导入数据库：

mysql -u root -p store\_management < store\_management\_backup.sql

### 5.3.2. 数据库运行维护方法

**4.1 数据库更新**

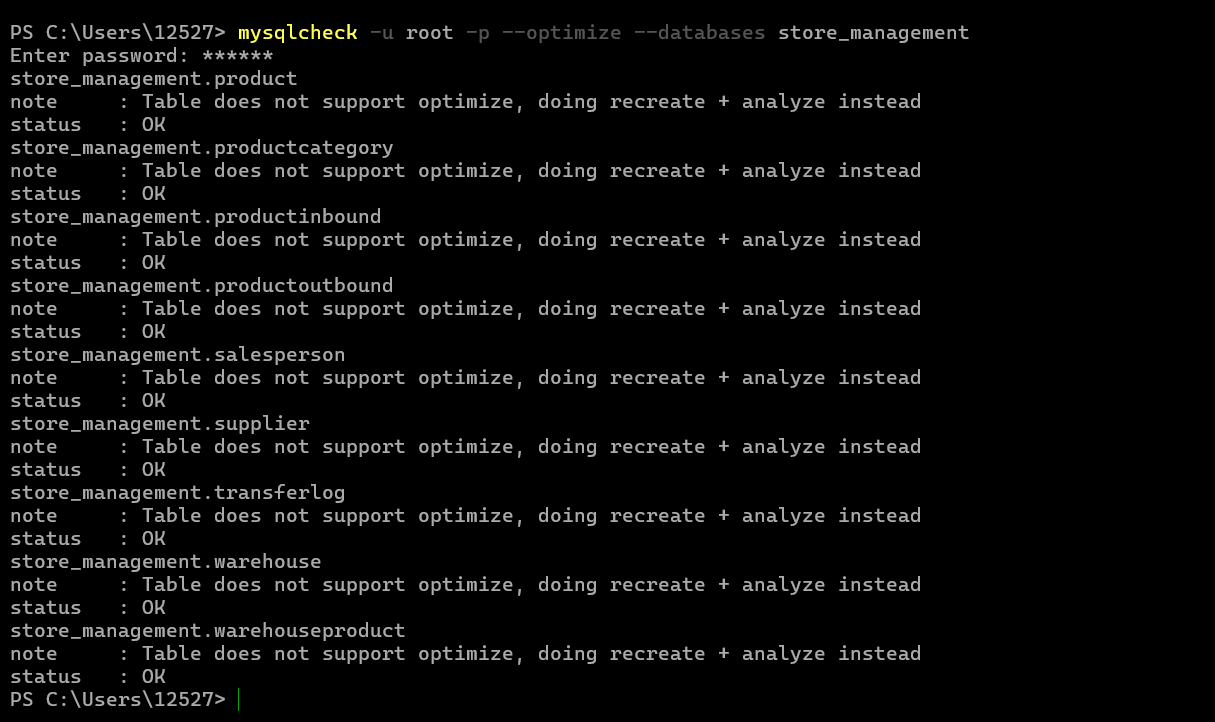
* 在更新表结构时，使用事务避免影响在线业务。
* 示例：添加字段时，首先检查是否会影响现有应用程序。

ALTER TABLE Product ADD COLUMN Description VARCHAR(255);

**4.2 数据库健康检查**

* 定期运行检查：

mysqlcheck -u root -p --optimize --databases store\_management



**4.3 容量规划**

* 使用 information\_schema 查询数据库大小：

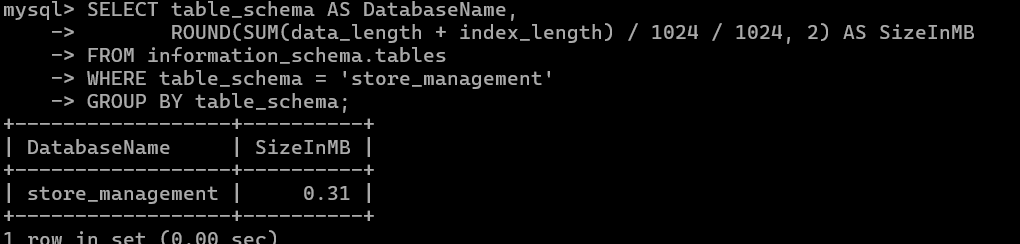
SELECT table\_schema AS DatabaseName,

ROUND(SUM(data\_length + index\_length) / 1024 / 1024, 2) AS SizeInMB

FROM information\_schema.tables

WHERE table\_schema = 'store\_management'

GROUP BY table\_schema;



# 六、总结（问题+方法+体会）

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。

期间遇到许多问题，好在成功解决：

外键约束冲突：在插入数据时，因外键依赖关系未满足，导致无法插入数据（如 ProductID 未在主表中定义）。

枚举值数据截断：商品单位字段使用 ENUM 类型时，插入不在允许范围内的值，触发 Data truncated 错误。

存储过程与触发器语法问题：实现触发器和存储过程时，因 MySQL 的DELIMITER 和 NEW不熟悉，导致多次错误。

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。