# 智能仓库管理系统数据库物理设计文档

**文档版本：** 1.0
**项目名称：** 智能仓库管理系统 (Warehouse Management System)
**编制日期：** 2025年11月
**技术栈：** MySQL 8.0 + JDK 8 + Tomcat 9.0 + JSP/Servlet

---

## 1. 物理设计概述

### 1.1 设计目标

基于逻辑设计模型，结合MySQL数据库特性和项目技术架构，设计高性能、高可用、易维护的物理数据库结构，确保系统在生产环境下的稳定运行和高效访问。

### 1.2 技术约束

1. **数据库系统**：MySQL 8.0
2. **连接方式**：JDBC连接池
3. **应用架构**：JSP/Servlet MVC模式
4. **部署环境**：Tomcat 9.0.82应用服务器
5. **开发环境**：JDK 8u202 + Eclipse 4.16

### 1.3 设计原则

1. **性能优先**：针对高频查询优化索引和表结构
2. **空间效率**：合理使用数据类型和存储引擎
3. **可维护性**：清晰的命名规范和注释
4. **安全性**：合理的数据访问权限控制
5. **可扩展性**：支持未来业务增长

---

## 2. 数据库环境配置

### 2.1 MySQL服务器配置

```
-- 基础配置 (my.cnf / my.ini)
[mysqld]
# 基础设置
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci
default-storage-engine = InnoDB
sql_mode =
ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIV
ISION_BY_ZERO,NO_ENGINE_SUBSTITUTION

# 连接配置
max_connections = 200
max_connect_errors = 10000
```

```
wait_timeout = 600
interactive_timeout = 600

# InnoDB配置
innodb_buffer_pool_size = 1G  -- 根据实际内存调整
innodb_log_file_size = 256M
innodb_flush_log_at_trx_commit = 2
innodb_lock_wait_timeout = 50
innodb_file_per_table = ON

# 查询缓存（MySQL 8.0已移除，此处使用性能架构）
performance_schema = ON
```

## 2.2 数据库创建脚本

```
-- 创建数据库（UTF8MB4支持完整Unicode）
CREATE DATABASE IF NOT EXISTS warehouse_ms
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;

-- 创建专用数据库用户（生产环境）
CREATE USER 'wms_user'@'localhost' IDENTIFIED BY 'StrongPass123!';
GRANT SELECT, INSERT, UPDATE, DELETE, EXECUTE ON warehouse_ms.* TO
'wms_user'@'localhost';
FLUSH PRIVILEGES;

-- 开发环境使用root用户（仅开发环境）
-- GRANT ALL PRIVILEGES ON warehouse_ms.* TO 'root'@'localhost';
```

---

# 3. 物理表结构设计

## 3.1 表空间规划

```
-- 表空间配置（InnoDB自动管理，无需手动指定）
-- 数据库文件路径：/var/lib/mysql/warehouse_ms/ (Linux) 或 C:\ProgramData\MySQL\...
(Windows)
```

## 3.2 详细物理表定义

### 3.2.1 warehouse（仓库表）

```
CREATE TABLE `warehouse` (
  `warehouse_id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '仓库ID，自增主键',
  `warehouse_code` VARCHAR(20) NOT NULL COMMENT '仓库编码，唯一标识',
  `warehouse_name` VARCHAR(50) NOT NULL COMMENT '仓库名称',
  `country` VARCHAR(50) NOT NULL COMMENT '所在国家',
  `address` VARCHAR(200) DEFAULT NULL COMMENT '详细地址',
  `is_active` TINYINT(1) NOT NULL DEFAULT '1' COMMENT '状态：1-有效，0-无效',
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT '更新时间',
  PRIMARY KEY (`warehouse_id`) USING BTREE,
```

```
  UNIQUE KEY `uk_warehouse_code` (`warehouse_code`) USING BTREE,
  KEY `idx_country` (`country`) USING BTREE,
  KEY `idx_active` (`is_active`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
COMMENT='仓库基本信息表'
ROW_FORMAT=DYNAMIC;
```

### 3.2.2 storage_location（货位表）

```
CREATE TABLE `storage_location` (
  `location_id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '货位ID，自增主键',
  `location_code` VARCHAR(20) NOT NULL COMMENT '货位编码，格式：仓库-区-排-层',
  `warehouse_id` INT(11) NOT NULL COMMENT '所属仓库ID',
  `status` TINYINT(1) NOT NULL DEFAULT '0' COMMENT '状态：0-空闲，1-占用',
  `capacity_length` DECIMAL(10,2) DEFAULT NULL COMMENT '货位长度容量(cm)',
  `capacity_width` DECIMAL(10,2) DEFAULT NULL COMMENT '货位宽度容量(cm)',
  `capacity_height` DECIMAL(10,2) DEFAULT NULL COMMENT '货位高度容量(cm)',
  `max_weight` DECIMAL(10,2) DEFAULT NULL COMMENT '最大承重(kg)',
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  PRIMARY KEY (`location_id`) USING BTREE,
  UNIQUE KEY `uk_location_code` (`location_code`) USING BTREE,
  KEY `idx_warehouse_id` (`warehouse_id`) USING BTREE,
  KEY `idx_status` (`status`) USING BTREE,
  KEY `idx_warehouse_status` (`warehouse_id`, `status`) USING BTREE,
  CONSTRAINT `fk_location_warehouse` FOREIGN KEY (`warehouse_id`) REFERENCES
`warehouse` (`warehouse_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
COMMENT='仓库货位信息表'
ROW_FORMAT=DYNAMIC;
```

### 3.2.3 product（产品表）

```
CREATE TABLE `product` (
  `product_id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '产品ID，自增主键',
  `sku` VARCHAR(50) NOT NULL COMMENT '产品SKU，唯一业务标识',
  `product_name` VARCHAR(100) NOT NULL COMMENT '产品名称',
  `category` VARCHAR(50) DEFAULT NULL COMMENT '产品分类',
  `specification` VARCHAR(200) DEFAULT NULL COMMENT '产品规格',
  `length` DECIMAL(10,2) DEFAULT NULL COMMENT '长度(cm)',
  `width` DECIMAL(10,2) DEFAULT NULL COMMENT '宽度(cm)',
  `height` DECIMAL(10,2) DEFAULT NULL COMMENT '高度(cm)',
  `weight` DECIMAL(10,2) DEFAULT NULL COMMENT '重量(kg)',
  `unit_price` DECIMAL(15,2) DEFAULT '0.00' COMMENT '单价（用于统计库存价值）',
  `safety_stock` INT(11) NOT NULL DEFAULT '0' COMMENT '安全库存',
  `is_active` TINYINT(1) NOT NULL DEFAULT '1' COMMENT '状态：1-有效，0-无效',
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT '更新时间',
  PRIMARY KEY (`product_id`) USING BTREE,
  UNIQUE KEY `uk_sku` (`sku`) USING BTREE,
  KEY `idx_category` (`category`) USING BTREE,
  KEY `idx_product_name` (`product_name`(20)) USING BTREE,
  KEY `idx_active_safety` (`is_active`, `safety_stock`) USING BTREE,
  KEY `idx_fulltext_search` (`product_name`, `specification`) USING BTREE
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
COMMENT='产品主数据表'
ROW_FORMAT=DYNAMIC;
```

### 3.2.4 inventory（库存表）

```
CREATE TABLE `inventory` (
  `inventory_id` BIGINT(20) NOT NULL AUTO_INCREMENT COMMENT '库存记录ID，自增主键',
  `product_id` INT(11) NOT NULL COMMENT '产品ID',
  `location_id` INT(11) NOT NULL COMMENT '货位ID',
  `batch_no` VARCHAR(50) DEFAULT NULL COMMENT '批次号',
  `quantity` INT(11) NOT NULL DEFAULT '0' COMMENT '当前数量',
  `inbound_time` DATETIME DEFAULT NULL COMMENT '入库时间',
  `expiry_date` DATE DEFAULT NULL COMMENT '有效期至（如有）',
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  `updated_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP COMMENT '更新时间',
  PRIMARY KEY (`inventory_id`) USING BTREE,
  UNIQUE KEY `uk_product_location` (`product_id`, `location_id`, `batch_no`)
USING BTREE,
  KEY `idx_product_id` (`product_id`) USING BTREE,
  KEY `idx_location_id` (`location_id`) USING BTREE,
  KEY `idx_inbound_time` (`inbound_time`) USING BTREE,
  KEY `idx_quantity` (`quantity`) USING BTREE,
  KEY `idx_product_location` (`product_id`, `location_id`) USING BTREE,
  CONSTRAINT `fk_inventory_product` FOREIGN KEY (`product_id`) REFERENCES
`product` (`product_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `fk_inventory_location` FOREIGN KEY (`location_id`) REFERENCES
`storage_location` (`location_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
COMMENT='库存明细表（核心表）'
ROW_FORMAT=DYNAMIC;
```

### 3.2.5 inbound_order（入库单表）

```
CREATE TABLE `inbound_order` (
  `order_id` BIGINT(20) NOT NULL AUTO_INCREMENT COMMENT '入库单ID，自增主键',
  `order_code` VARCHAR(50) NOT NULL COMMENT '入库单号，格式：IN-YYYYMMDD-001',
  `product_id` INT(11) NOT NULL COMMENT '产品ID',
  `location_id` INT(11) NOT NULL COMMENT '货位ID',
  `quantity` INT(11) NOT NULL COMMENT '入库数量',
  `batch_no` VARCHAR(50) DEFAULT NULL COMMENT '批次号',
  `supplier` VARCHAR(100) DEFAULT NULL COMMENT '供应商',
  `operator` VARCHAR(50) DEFAULT NULL COMMENT '操作员',
  `order_time` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '入库时间',
  `remark` VARCHAR(500) DEFAULT NULL COMMENT '备注',
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  PRIMARY KEY (`order_id`) USING BTREE,
  UNIQUE KEY `uk_order_code` (`order_code`) USING BTREE,
  KEY `idx_product_id` (`product_id`) USING BTREE,
  KEY `idx_location_id` (`location_id`) USING BTREE,
  KEY `idx_order_time` (`order_time`) USING BTREE,
  KEY `idx_operator` (`operator`) USING BTREE,
  KEY `idx_product_time` (`product_id`, `order_time`) USING BTREE,
```

```
  CONSTRAINT `fk_inbound_product` FOREIGN KEY (`product_id`) REFERENCES `product`
(`product_id`) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT `fk_inbound_location` FOREIGN KEY (`location_id`) REFERENCES
`storage_location` (`location_id`) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
COMMENT='入库单表'
ROW_FORMAT=DYNAMIC;
```

### 3.2.6 outbound_order（出库单表）

```
CREATE TABLE `outbound_order` (
  `order_id` BIGINT(20) NOT NULL AUTO_INCREMENT COMMENT '出库单ID，自增主键',
  `order_code` VARCHAR(50) NOT NULL COMMENT '出库单号，格式：OUT-YYYYMMDD-001',
  `product_id` INT(11) NOT NULL COMMENT '产品ID',
  `location_id` INT(11) NOT NULL COMMENT '货位ID',
  `quantity` INT(11) NOT NULL COMMENT '出库数量',
  `batch_no` VARCHAR(50) DEFAULT NULL COMMENT '批次号',
  `customer` VARCHAR(100) DEFAULT NULL COMMENT '客户',
  `operator` VARCHAR(50) DEFAULT NULL COMMENT '操作员',
  `order_time` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '出库时间',
  `remark` VARCHAR(500) DEFAULT NULL COMMENT '备注',
  `created_at` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间',
  PRIMARY KEY (`order_id`) USING BTREE,
  UNIQUE KEY `uk_order_code` (`order_code`) USING BTREE,
  KEY `idx_product_id` (`product_id`) USING BTREE,
  KEY `idx_location_id` (`location_id`) USING BTREE,
  KEY `idx_order_time` (`order_time`) USING BTREE,
  KEY `idx_operator` (`operator`) USING BTREE,
  KEY `idx_customer` (`customer`(20)) USING BTREE,
  KEY `idx_product_time` (`product_id`, `order_time`) USING BTREE,
  CONSTRAINT `fk_outbound_product` FOREIGN KEY (`product_id`) REFERENCES
`product` (`product_id`) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT `fk_outbound_location` FOREIGN KEY (`location_id`) REFERENCES
`storage_location` (`location_id`) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci
COMMENT='出库单表'
ROW_FORMAT=DYNAMIC;
```

---

# 4. 索引策略优化

## 4.1 B-Tree索引设计原则

1. **前缀索引**：对长字符串字段使用前缀索引（如 product_name(20)）
2. **覆盖索引**：高频查询尽量使用覆盖索引
3. **复合索引**：按查询条件顺序创建复合索引
4. **选择性原则**：优先为高选择性字段创建索引

## 4.2 关键索引说明

```sql
-- 1. 仪表盘查询优化索引
CREATE INDEX idx_dashboard_summary ON inventory(product_id, location_id,
quantity);

-- 2. 低库存预警查询优化
CREATE INDEX idx_low_stock_check ON product(is_active, safety_stock, product_id);

-- 3. 智能货位推荐查询优化
CREATE INDEX idx_location_recommend ON storage_location(warehouse_id, status,
capacity_length, capacity_width);

-- 4. FIFO出库策略优化
CREATE INDEX idx_fifo_outbound ON inbound_order(product_id, location_id,
order_time, batch_no);
```

## 4.3 索引使用监控

```sql
-- 查看索引使用情况
SELECT
    OBJECT_SCHEMA,
    OBJECT_NAME,
    INDEX_NAME,
    COUNT_READ,
    COUNT_FETCH,
    COUNT_INSERT,
    COUNT_UPDATE,
    COUNT_DELETE
FROM performance_schema.table_io_waits_summary_by_index_usage
WHERE OBJECT_SCHEMA = 'warehouse_ms'
ORDER BY COUNT_READ DESC;

-- 查看未使用的索引
SELECT
    OBJECT_SCHEMA,
    OBJECT_NAME,
    INDEX_NAME
FROM performance_schema.table_io_waits_summary_by_index_usage
WHERE OBJECT_SCHEMA = 'warehouse_ms'
  AND INDEX_NAME IS NOT NULL
  AND COUNT_STAR = 0
  AND OBJECT_SCHEMA NOT IN ('mysql', 'performance_schema', 'information_schema')
ORDER BY OBJECT_SCHEMA, OBJECT_NAME;
```

# 5. 分区与分表策略

## 5.1 时间分区策略

```sql
-- inbound_order 按月分区
ALTER TABLE inbound_order
PARTITION BY RANGE (YEAR(order_time) * 100 + MONTH(order_time)) (
    PARTITION p202401 VALUES LESS THAN (202402),
    PARTITION p202402 VALUES LESS THAN (202403),
    PARTITION p202403 VALUES LESS THAN (202404),
    PARTITION p_future VALUES LESS THAN MAXVALUE
);

-- outbound_order 类似分区
```

## 5.2 归档策略

```sql
-- 创建归档表（存储2年以上数据）
CREATE TABLE inbound_order_archive LIKE inbound_order;
CREATE TABLE outbound_order_archive LIKE outbound_order;

-- 月度归档存储过程
DELIMITER //
CREATE PROCEDURE archive_old_data()
BEGIN
    DECLARE cutoff_date DATE;
    SET cutoff_date = DATE_SUB(CURDATE(), INTERVAL 2 YEAR);

    -- 归档入库单
    INSERT INTO inbound_order_archive
    SELECT * FROM inbound_order
    WHERE order_time < cutoff_date;

    DELETE FROM inbound_order
    WHERE order_time < cutoff_date;

    -- 归档出库单
    INSERT INTO outbound_order_archive
    SELECT * FROM outbound_order
    WHERE order_time < cutoff_date;

    DELETE FROM outbound_order
    WHERE order_time < cutoff_date;

    -- 优化表空间
    OPTIMIZE TABLE inbound_order;
    OPTIMIZE TABLE outbound_order;
END //
DELIMITER ;

-- 创建定时任务（每月1日执行）
CREATE EVENT archive_monthly
ON SCHEDULE EVERY 1 MONTH
STARTS '2025-02-01 02:00:00'
```

```
DO CALL archive_old_data();
```

# 6. 存储过程与触发器设计

## 6.1 业务逻辑存储过程

### 6.1.1 智能入库过程

```
DELIMITER //
CREATE PROCEDURE sp_smart_inbound(
    IN p_sku VARCHAR(50),
    IN p_quantity INT,
    IN p_operator VARCHAR(50),
    IN p_supplier VARCHAR(100),
    OUT p_order_code VARCHAR(50),
    OUT p_location_code VARCHAR(20)
)
BEGIN
    DECLARE v_product_id INT;
    DECLARE v_location_id INT;
    DECLARE v_order_code VARCHAR(50);
    DECLARE v_batch_no VARCHAR(50);

    -- 获取产品ID
    SELECT product_id INTO v_product_id
    FROM product
    WHERE sku = p_sku AND is_active = 1;

    IF v_product_id IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '产品不存在或已停用';
    END IF;

    -- 智能货位推荐（简版：找第一个空闲货位）
    SELECT location_id, location_code INTO v_location_id, p_location_code
    FROM storage_location
    WHERE status = 0
    AND warehouse_id = 1  -- 默认第一个仓库
    LIMIT 1;

    IF v_location_id IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '没有可用货位';
    END IF;

    -- 生成入库单号
    SET v_order_code = CONCAT('IN-', DATE_FORMAT(NOW(), '%Y%m%d'), '-',
        LPAD((SELECT COUNT(*) FROM inbound_order WHERE DATE(order_time) =
CURDATE()) + 1, 3, '0'));

    -- 生成批次号
    SET v_batch_no = CONCAT('BATCH-', DATE_FORMAT(NOW(), '%Y%m%d'), '-',
        LPAD(v_product_id, 4, '0'));
```

```
    -- 插入入库单
    INSERT INTO inbound_order (order_code, product_id, location_id, quantity,
                               batch_no, supplier, operator, order_time)
    VALUES (v_order_code, v_product_id, v_location_id, p_quantity,
            v_batch_no, p_supplier, p_operator, NOW());

    -- 更新库存
    INSERT INTO inventory (product_id, location_id, batch_no, quantity,
inbound_time)
    VALUES (v_product_id, v_location_id, v_batch_no, p_quantity, NOW())
    ON DUPLICATE KEY UPDATE
    quantity = quantity + p_quantity,
    updated_at = NOW();

    -- 更新货位状态
    UPDATE storage_location
    SET status = 1
    WHERE location_id = v_location_id;

    SET p_order_code = v_order_code;
END //
DELIMITER ;
```

### 6.1.2 FIFO出库过程

```
DELIMITER //
CREATE PROCEDURE sp_fifo_outbound(
    IN p_sku VARCHAR(50),
    IN p_quantity INT,
    IN p_operator VARCHAR(50),
    IN p_customer VARCHAR(100),
    OUT p_order_code VARCHAR(50)
)
BEGIN
    DECLARE v_product_id INT;
    DECLARE v_remaining_quantity INT DEFAULT p_quantity;
    DECLARE v_available_quantity INT;
    DECLARE v_location_id INT;
    DECLARE v_batch_no VARCHAR(50);
    DECLARE v_order_code VARCHAR(50);
    DECLARE done INT DEFAULT FALSE;

    -- 游标：按入库时间升序获取库存批次
    DECLARE cur_inventory CURSOR FOR
    SELECT i.location_id, i.batch_no, i.quantity
    FROM inventory i
    JOIN product p ON i.product_id = p.product_id
    WHERE p.sku = p_sku
      AND i.quantity > 0
    ORDER BY i.inbound_time ASC;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    -- 获取产品ID
```

```
    SELECT product_id INTO v_product_id
    FROM product
    WHERE sku = p_sku AND is_active = 1;

    IF v_product_id IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '产品不存在或已停用';
    END IF;

    -- 检查总库存是否足够
    SELECT COALESCE(SUM(quantity), 0) INTO v_available_quantity
    FROM inventory i
    WHERE i.product_id = v_product_id;

    IF v_available_quantity < p_quantity THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = CONCAT('库存不足，可用数量: ', v_available_quantity);
    END IF;

    -- 生成出库单号
    SET v_order_code = CONCAT('OUT-', DATE_FORMAT(NOW(), '%Y%m%d'), '-',
        LPAD((SELECT COUNT(*) FROM outbound_order WHERE DATE(order_time) =
CURDATE()) + 1, 3, '0'));

    OPEN cur_inventory;

    inventory_loop: LOOP
        FETCH cur_inventory INTO v_location_id, v_batch_no, v_available_quantity;

        IF done THEN
            LEAVE inventory_loop;
        END IF;

        IF v_available_quantity >= v_remaining_quantity THEN
            -- 当前批次足够
            INSERT INTO outbound_order (order_code, product_id, location_id,
quantity,
                                        batch_no, customer, operator, order_time)
            VALUES (v_order_code, v_product_id, v_location_id,
v_remaining_quantity,
                    v_batch_no, p_customer, p_operator, NOW());

            -- 更新库存
            UPDATE inventory
            SET quantity = quantity - v_remaining_quantity,
                updated_at = NOW()
            WHERE product_id = v_product_id
              AND location_id = v_location_id
              AND batch_no = v_batch_no;

            -- 如果库存为0，更新货位状态
            IF v_available_quantity - v_remaining_quantity = 0 THEN
                UPDATE storage_location
                SET status = 0
                WHERE location_id = v_location_id;
            END IF;
```

```
            SET v_remaining_quantity = 0;
            LEAVE inventory_loop;
        ELSE
            -- 当前批次不足，全部出库
            INSERT INTO outbound_order (order_code, product_id, location_id,
quantity,
                                        batch_no, customer, operator, order_time)
            VALUES (v_order_code, v_product_id, v_location_id,
v_available_quantity,
                    v_batch_no, p_customer, p_operator, NOW());

            -- 更新库存（设置为0）
            UPDATE inventory
            SET quantity = 0,
                updated_at = NOW()
            WHERE product_id = v_product_id
              AND location_id = v_location_id
              AND batch_no = v_batch_no;

            -- 更新货位状态（可能变为空闲）
            UPDATE storage_location
            SET status = 0
            WHERE location_id = v_location_id;

            SET v_remaining_quantity = v_remaining_quantity -
v_available_quantity;
        END IF;
    END LOOP;

    CLOSE cur_inventory;

    SET p_order_code = v_order_code;
END //
DELIMITER ;
```

## 6.2 触发器设计

### 6.2.1 库存自动更新触发器

```
-- 入库触发库存增加
DELIMITER //
CREATE TRIGGER trg_after_inbound_insert
AFTER INSERT ON inbound_order
FOR EACH ROW
BEGIN
    -- 插入或更新库存记录
    INSERT INTO inventory (product_id, location_id, batch_no, quantity,
inbound_time)
    VALUES (NEW.product_id, NEW.location_id, NEW.batch_no, NEW.quantity,
NEW.order_time)
    ON DUPLICATE KEY UPDATE
    quantity = quantity + NEW.quantity,
    updated_at = NOW();
```

```
    -- 更新货位状态
    UPDATE storage_location
    SET status = 1
    WHERE location_id = NEW.location_id;
END //
DELIMITER ;

-- 库存检查触发器（防止负库存）
DELIMITER //
CREATE TRIGGER trg_before_inventory_update
BEFORE UPDATE ON inventory
FOR EACH ROW
BEGIN
    IF NEW.quantity < 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = '库存数量不能为负数';
    END IF;
END //
DELIMITER ;
```

# 7. 备份与恢复策略

## 7.1 备份策略

```
-- 1. 全量备份（每日凌晨2点）
mysqldump -u root -p warehouse_ms > /backup/warehouse_ms_full_$(date +%Y%m%d).sql

-- 2. 增量备份（启用二进制日志）
-- my.cnf配置：
[mysqld]
log-bin = /var/log/mysql/mysql-bin.log
expire_logs_days = 7
max_binlog_size = 100M

-- 3. 备份脚本示例
#!/bin/bash
# backup_wms.sh
BACKUP_DIR="/backup/mysql"
DATE=$(date +%Y%m%d_%H%M%S)
DB_NAME="warehouse_ms"

# 全量备份
mysqldump -u root -p${MYSQL_PWD} --single-transaction --routines --triggers
${DB_NAME} > ${BACKUP_DIR}/${DB_NAME}_full_${DATE}.sql

# 压缩备份文件
gzip ${BACKUP_DIR}/${DB_NAME}_full_${DATE}.sql

# 保留最近7天备份
find ${BACKUP_DIR} -name "${DB_NAME}_full_*.gz" -mtime +7 -delete

# 刷新二进制日志
mysql -u root -p${MYSQL_PWD} -e "FLUSH BINARY LOGS;"
```

## 7.2 恢复策略

```
-- 全量恢复
mysql -u root -p warehouse_ms < warehouse_ms_full_20251216.sql

-- 时间点恢复（使用二进制日志）
mysqlbinlog --start-datetime="2025-12-16 10:00:00" --stop-datetime="2025-12-16
12:00:00" mysql-bin.000001 | mysql -u root -p
```

---

# 8. 性能监控与优化

## 8.1 监控指标

```
-- 1. 查询性能监控
SHOW GLOBAL STATUS LIKE 'Slow_queries';
SHOW VARIABLES LIKE 'long_query_time';

-- 2. 连接数监控
SHOW GLOBAL STATUS LIKE 'Threads_connected';
SHOW VARIABLES LIKE 'max_connections';

-- 3. InnoDB性能监控
SHOW ENGINE INNODB STATUS;

-- 4. 表空间使用
SELECT
    table_schema,
    table_name,
    ROUND(((data_length + index_length) / 1024 / 1024), 2) AS total_mb,
    ROUND((data_length / 1024 / 1024), 2) AS data_mb,
    ROUND((index_length / 1024 / 1024), 2) AS index_mb,
    table_rows
FROM information_schema.tables
WHERE table_schema = 'warehouse_ms'
ORDER BY total_mb DESC;
```

## 8.2 优化建议

```
-- 1. 定期优化表
OPTIMIZE TABLE inventory;
OPTIMIZE TABLE inbound_order;
OPTIMIZE TABLE outbound_order;

-- 2. 更新统计信息
ANALYZE TABLE warehouse, storage_location, product, inventory, inbound_order,
outbound_order;

-- 3. 缓存优化
SHOW VARIABLES LIKE 'query_cache%';
SET GLOBAL query_cache_size = 64*1024*1024;  -- 64MB查询缓存

-- 4. 连接池优化（应用层）
```

```
# Tomcat连接池配置（context.xml）
<Resource name="jdbc/WMSDB"
          auth="Container"
          type="javax.sql.DataSource"
          factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
          driverClassName="com.mysql.cj.jdbc.Driver"
          url="jdbc:mysql://localhost:3306/warehouse_ms?
useUnicode=true&amp;characterEncoding=utf8&amp;useSSL=false&amp;serverTimezone=UT
C"
          username="wms_user"
          password="StrongPass123!"
          maxTotal="100"
          maxIdle="30"
          maxWaitMillis="10000"
          validationQuery="SELECT 1"
          testOnBorrow="true"
          removeAbandoned="true"
          removeAbandonedTimeout="60"
          logAbandoned="true"/>
```

# 9. 安全设计

## 9.1 权限管理

```
-- 创建不同权限用户
-- 1. 只读用户（用于报表查询）
CREATE USER 'wms_report'@'localhost' IDENTIFIED BY 'ReportPass123!';
GRANT SELECT ON warehouse_ms.* TO 'wms_report'@'localhost';

-- 2. 操作员用户（只能操作出入库）
CREATE USER 'wms_operator'@'localhost' IDENTIFIED BY 'OperatorPass123!';
GRANT SELECT, INSERT, UPDATE ON warehouse_ms.inbound_order TO
'wms_operator'@'localhost';
GRANT SELECT, INSERT, UPDATE ON warehouse_ms.outbound_order TO
'wms_operator'@'localhost';
GRANT SELECT ON warehouse_ms.product TO 'wms_operator'@'localhost';
GRANT SELECT ON warehouse_ms.inventory TO 'wms_operator'@'localhost';

-- 3. 管理员用户（全权限）
CREATE USER 'wms_admin'@'localhost' IDENTIFIED BY 'AdminPass123!';
GRANT ALL PRIVILEGES ON warehouse_ms.* TO 'wms_admin'@'localhost';
```

### 9.2 数据加密

```sql
-- 敏感数据加密（如操作员密码，如果后续扩展用户表）
CREATE TABLE user_account (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE,
    password_hash VARCHAR(255),  -- 存储加密后的密码
    salt VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- 使用AES_ENCRYPT加密敏感数据
INSERT INTO user_account (username, password_hash, salt)
VALUES ('admin', AES_ENCRYPT('password123', 'encryption_key'), 'salt123');
```

---

## 10. 部署与维护

### 10.1 初始化脚本

```bash
#!/bin/bash
# init_wms_db.sh

echo "开始初始化WMS数据库..."

# 1. 创建数据库
mysql -u root -p -e "CREATE DATABASE IF NOT EXISTS warehouse_ms CHARACTER SET
utf8mb4 COLLATE utf8mb4_unicode_ci;"

# 2. 执行建表脚本
mysql -u root -p warehouse_ms < 01_create_tables.sql

# 3. 创建存储过程和触发器
mysql -u root -p warehouse_ms < 03_stored_procedures.sql

# 4. 导入初始数据
mysql -u root -p warehouse_ms < 02_sample_data.sql

# 5. 创建索引优化
mysql -u root -p warehouse_ms < 04_create_indexes.sql

echo "数据库初始化完成！"
```

### 10.2 维护计划

```sql
-- 每周维护任务
-- 1. 更新统计信息
ANALYZE TABLE warehouse, product, inventory;

-- 2. 检查并修复表
CHECK TABLE inventory;
REPAIR TABLE inventory;
```

```sql
-- 3. 清理临时表
FLUSH TABLES;

-- 每月维护任务
-- 1. 归档旧数据
CALL archive_old_data();

-- 2. 优化表空间
OPTIMIZE TABLE inbound_order, outbound_order;

-- 3. 清理二进制日志
PURGE BINARY LOGS BEFORE DATE_SUB(NOW(), INTERVAL 30 DAY);
```

---

# 11. 附录

## 11.1 连接池配置（JSP/Servlet）

```
<!-- web.xml配置 -->
<resource-ref>
    <description>WMS Database Connection</description>
    <res-ref-name>jdbc/WMSDB</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
</resource-ref>

<!-- JSP中使用 -->
<%
    Context initContext = new InitialContext();
    Context envContext  = (Context)initContext.lookup("java:/comp/env");
    DataSource ds = (DataSource)envContext.lookup("jdbc/WMSDB");
    Connection conn = ds.getConnection();

    // 使用连接...
    conn.close();
%>
```