CA389-StockWise Warehouse Management System

CA389-智仓易仓库管理系统

Project Number:CA389

Powered by :Xander Ray , Caffz.com

Index

StockWise Warehouse Management System Design File

Here's an example of a typical PHP repository system feature split, including common core modules, main features, and mapping each feature to a specific PHP file in the tradition of "one module per folder". This structure is for reference only and can be added, deleted, or merged according to actual business needs.

1. Overview of module division

Generally, a small and medium-sized WMS (Warehouse Management System) can be split into the following 9 modules:

1. Auth (Certification)
2. User
3. Role & Permission
4. Inventory
5. Procurement
6. Sales
7. Logistics
8. Reports
9. Settings

Illustrate:
For the sake of demonstration, roles and permissions are placed in a separate module (which can also be put together with user management).
Procurement and sales can be regarded as two major business links, which manage suppliers, purchase orders, customers, sales orders, etc.
The report module can be expanded to more varieties according to needs.
System settings can include warehouse information, product information, global configuration, and so on.

2. the functions of each module are separated from the file

The following is a brief description of the main functions of each module, and the corresponding PHP file examples are listed. The file name can be adjusted by yourself, e.g. 'create.php' can be named 'add.php' and

so on.

2.1. Auth (Authentication Module)

Function description: Responsible for system login, logout, registration, password retrieval and other operations.
File structure (example):
```
/Auth
├── login.php // User login
├── logout.php // User logout
├── register.php // New user registration
└── reset_password.php // Password Reset (Retrieve)
```
Total number of files: 4

2.2. User (User Management Module)

Function description: Manage internal users in the system, including adding new users, editing users, deleting users, and viewing user lists.
File structure (example):
```
/User
├── index.php // User list, search
├── create.php // Add a new user
├── edit.php // Edit user
├── delete.php // Delete user
└── profile.php // View and modify your personal information
```
Total number of files: 5

2.3. Role & Permission

Function description: It is used to manage user roles and their permissions, and can assign permissions to different roles or users to control the access scope of the system.

File structure (example):

```
/Role
├── index.php // Role List & View Permissions
├── create.php // Add role & set permissions
├── edit.php // Edit roles and permissions
├── delete.php // Delete a role
└── assign_permission.php // Assign permissions (to role / to user)
```

Total number of files: 5

2.4. Inventory

Function description: It covers core functions such as warehousing, warehousing, allocation, inventory, and inventory inquiry.

File structure (example):

```
/Inventory
├── index.php // Inventory overview, showing the overall inventory situation
├── inbound.php // Inbound operation (docking purchase or return warehousing)
├── outbound.php // Outbound operation (docking sales and delivery)
├── transfer.php // Transfer between warehouses
├── adjust.php // Inventory, inventory variance adjustment
```

└── detail.php // View the inventory details of a single product

```

Total number of files: 6

## 2.5. Procurement (Procurement Management Module)

Function Description: Manage the creation, modification, and arrival confirmation of purchase orders, as well as the maintenance of supplier information.
File structure (example):
```

/Procurement

├── index.php // Purchase order list

├── create.php // Create a new purchase order

├── edit.php // Edit the purchase order

├── receive.php // Purchase arrival confirmation

└── supplier.php // Supplier Management (Add, Delete, Modify and Check)

```

Total number of files: 5

## 2.6. Sales (Sales Management Module)

Function Description: Manage sales orders and their fulfillment processes, and maintain customer information.
File structure (example):
```

/Sales

├── index.php // List of sales orders

├── create.php // Create a new sales order

```
├── edit.php // Edit the sales order
├── ship.php // Shipping operation (generation of outbound order/logistics order)
└── customer.php // Customer Management (Add, Delete, Modify and Search)
```

Total number of files: 5

2.7. Logistics (Logistics Management Module)

Function Description: Involves logistics company maintenance and logistics tracking functions; It can be connected with the third-party logistics interface to query the progress.

File structure (example):

```
/Logistics
├── index.php // List of logistics companies
├── create.php // Add logistics company information
├── edit.php // Edit logistics company information
├── delete.php // Delete the logistics company
└── track.php // Logistics tracking
```

Total number of files: 5

2.8. Reports

Function description: Generate and view multi-dimensional data reports such as inventory, procurement, sales, logistics, and finance, and export them are supported.

File structure (example):

```
/Reports
├── index.php // Report overview, select to view different types of reports
```

```
├── inventory_report.php // Inventory report
├── procurement_report.php // Purchase report
├── sales_report.php // Sales report
├── logistics_report.php // Logistics report
└── finance_report.php // Financial related (e.g. profits, expenses, etc.)
```

Total number of files: 6

2.9. Settings

Function description: It is mainly used for warehouse information, commodity information, system global configuration, data backup, etc.
File structure (example):

```
/Settings
├── index.php // System Settings Overview (Global Settings Entry)
├── warehouse.php // Warehouse information maintenance (adding, deleting, modifying)
├── product.php // Product & Category Maintenance
├── site_config.php // System/site basic configuration
└── backup.php // Database backup and restore
```

Total number of files: 5

3. Total number and summary of documents

Adding the number of PHP files for each module in the example above gives you a total of 46 files (as shown below):

Auth： 4

User： 5

Role & Permission： 5

Inventory： 6

Procurement： 5

Sales： 5

Logistics： 5

Reports： 6

Settings： 5

Total: 4 + 5 + 5 + 6 + 5 + 5 + 5 + 6 + 5 = 46

The above structure is only an example reference, and the actual project will be merged or split according to the company's business process and team development habits (for example, 'Auth' and 'User' may be merged together, 'Role & Permission' may be placed inside the 'User' module, etc.).

Additional additions

Dashboard
Usually the back-end management system will also have a "dashboard" or "dashboard" that shows an overview of key statistics (e.g. total inventory, purchases/sales for the month, etc.). It can be managed in the root directory with 'dashboard.php' or in a dedicated folder.
Public Functions / Class Libraries
Things like database connections, generic functions, model classes, etc., tend to be placed in separate 'lib', 'common', or 'app' folders, and don't belong to individual modules.
Naming style
If it's based on a framework (e.g., Laravel, CodeIgniter, etc.), the file structure will be different; If you are using native PHP, you can use the traditional "one function, one PHP file" model described above for quick development and maintenance.

Conclusion:
The above is an example of a typical small and medium-sized warehouse management system (WMS) with one folder per module and one PHP file per function, totaling about 9 modules and 46 files.
The actual project can be fine-tuned according to the business complexity and team development model: it is common to merge and simplify or further disassemble molecular modules.

The ultimate goal is to ensure that the responsibilities of the module are clear, the code is easy to read, and the maintainability is high.

4. MYSQL 数据库设计部分

以下是基于 MySQL 8 设计的 仓库管理系统（WMS）数据库架构，涵盖 用户管理、角色权限、库存管理、采购、销售、物流、报表 等核心模块。

4.1. 数据库命名

数据库名称： `warehouse_management`

4.2. 数据库表设计

按照模块划分，设计以下 12 张核心数据表，可根据业务需求扩展。

4.2.1. 用户与权限管理

4.2.1.1 用户表 (`users`)

用于存储 系统用户（管理员、仓库管理员、采购员等） 的信息。

```sql
CREATE TABLE users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,    -- 用户名
    password VARCHAR(255) NOT NULL,          -- 密码（加密存储）
    email VARCHAR(100) UNIQUE,               -- 邮箱
    phone VARCHAR(20) UNIQUE,                -- 手机号
    role_id INT,                             -- 角色 ID（外键）
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (role_id) REFERENCES roles(id) ON DELETE SET NULL
);
```

4.2.1.2 角色表（`roles`）

存储不同用户的角色，如 管理员、仓库管理员、采购员。

```sql
CREATE TABLE roles (
    id INT PRIMARY KEY AUTO_INCREMENT,
    role_name VARCHAR(50) UNIQUE NOT NULL    -- 角色名称
);
```

4.2.1.3 权限表（`permissions`）

存储不同角色的 权限（增删改查等）。

```sql
CREATE TABLE permissions (
    id INT PRIMARY KEY AUTO_INCREMENT,
    permission_name VARCHAR(100) UNIQUE NOT NULL
);
```

4.2.1.4 角色-权限关联表（`role_permissions`）

用于多对多存储 角色与权限的映射。

```sql
CREATE TABLE role_permissions (
    role_id INT,
    permission_id INT,
    PRIMARY KEY (role_id, permission_id),
    FOREIGN KEY (role_id) REFERENCES roles(id) ON DELETE CASCADE,
    FOREIGN KEY (permission_id) REFERENCES permissions(id) ON DELETE CASCADE
);
```

4.2.3. 库存管理

4.2.3.1 仓库表（`warehouses`）

存储 不同仓库 的信息。

```sql
CREATE TABLE warehouses (
    id INT PRIMARY KEY AUTO_INCREMENT,
    warehouse_name VARCHAR(100) NOT NULL,  -- 仓库名称
    location VARCHAR(255),                 -- 位置
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4.2.3.2 商品表（`products`）

存储 商品信息（SKU、条形码、名称、分类）。

```sql
CREATE TABLE products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    sku VARCHAR(50) UNIQUE NOT NULL,       -- SKU
    barcode VARCHAR(50) UNIQUE,            -- 条形码
    product_name VARCHAR(100) NOT NULL,    -- 商品名称
    category VARCHAR(50),                  -- 商品分类
    unit VARCHAR(20),                      -- 计量单位（件、箱、吨）
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4.2.3.3 库存表（`inventory`）

存储 每个仓库的库存数量。

```sql
CREATE TABLE inventory (
    id INT PRIMARY KEY AUTO_INCREMENT,
    warehouse_id INT NOT NULL,             -- 仓库 ID
```

```
    product_id INT NOT NULL,                    -- 商品 ID
    quantity INT NOT NULL DEFAULT 0,            -- 库存数量
    last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (warehouse_id) REFERENCES warehouses(id) ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE CASCADE
);
```

4.2.4. 采购管理

4.2.4..1 供应商表 (`suppliers`)

存储 供应商信息。

```sql
CREATE TABLE suppliers (
    id INT PRIMARY KEY AUTO_INCREMENT,
    supplier_name VARCHAR(100) NOT NULL,    -- 供应商名称
    contact_person VARCHAR(100),            -- 联系人
    phone VARCHAR(20),                      -- 联系电话
    email VARCHAR(100),
    address VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4.2.4.2 采购订单表 (`purchase_orders`)

存储 采购订单（对应多个采购商品）。

```sql
CREATE TABLE purchase_orders (
    id INT PRIMARY KEY AUTO_INCREMENT,
    supplier_id INT NOT NULL,                   -- 供应商 ID
    order_date DATE NOT NULL,                   -- 订单日期
    status ENUM('Pending', 'Received', 'Canceled') DEFAULT 'Pending', -- 采购状态
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (supplier_id) REFERENCES suppliers(id) ON DELETE CASCADE
);
```

```

#### 4.2.4.3 采购明细表（`purchase_order_items`）

存储 采购订单中的商品。

```sql
CREATE TABLE purchase_order_items (
    id INT PRIMARY KEY AUTO_INCREMENT,
    purchase_order_id INT NOT NULL,          -- 采购订单 ID
    product_id INT NOT NULL,                 -- 商品 ID
    quantity INT NOT NULL,                   -- 采购数量
    unit_price DECIMAL(10,2) NOT NULL,       -- 单价
    FOREIGN KEY (purchase_order_id) REFERENCES purchase_orders(id) ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE CASCADE
);
```

4.2.5. 销售管理

4.2.5.1 客户表（`customers`）

存储 客户信息。

```sql
CREATE TABLE customers (
    id INT PRIMARY KEY AUTO_INCREMENT,
    customer_name VARCHAR(100) NOT NULL,
    contact_person VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100),
    address VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```
```

4.2.5.2 销售订单表 (`sales_orders`)

存储 销售订单（对应多个商品）。

```sql
CREATE TABLE sales_orders (
    id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT NOT NULL,
    order_date DATE NOT NULL,
    status ENUM('Pending', 'Shipped', 'Completed', 'Canceled') DEFAULT 'Pending',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (customer_id) REFERENCES customers(id) ON DELETE CASCADE
);
```

4.2.5.3 销售明细表 (`sales_order_items`)

存储 销售订单的商品。

```sql
CREATE TABLE sales_order_items (
    id INT PRIMARY KEY AUTO_INCREMENT,
    sales_order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (sales_order_id) REFERENCES sales_orders(id) ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE CASCADE
);
```

4.2.6. 物流管理

4.2.6.1 物流公司表 (`logistics_providers`)

存储 物流公司信息。

```sql
CREATE TABLE logistics_providers (
    id INT PRIMARY KEY AUTO_INCREMENT,
```

```
    provider_name VARCHAR(100) NOT NULL,
    contact_person VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4.2.6.2 物流记录表（`shipments`）

存储 物流记录。

```sql
CREATE TABLE shipments (
    id INT PRIMARY KEY AUTO_INCREMENT,
    sales_order_id INT NOT NULL,
    tracking_number VARCHAR(50),
    provider_id INT NOT NULL,
    shipped_date DATE,
    delivery_status ENUM('In Transit', 'Delivered', 'Failed') DEFAULT 'In Transit',
    FOREIGN KEY (sales_order_id) REFERENCES sales_orders(id) ON DELETE CASCADE,
    FOREIGN KEY (provider_id) REFERENCES logistics_providers(id) ON DELETE CASCADE
);
```

结论

- 共 12 张核心表，涵盖 用户管理、库存、采购、销售、物流 等功能。
- 索引优化：主键 `id` + 外键约束，支持高效查询。
- 数据完整性：`ENUM` 控制订单状态，`FOREIGN KEY` 保证数据一致性。

Xander Ray CEO

Website:www.caffz.com

Guangdong zhizhou digital technology company

Cellphone 13826867328