C409物料管理说明

数据库创建与使用说明文档。这份文档总结了如何创建数据库、表结构、使用 Python 脚本进行库存管理、项目管理、物料预留等功能。

实验室元器件管理系统 - 数据库创建与使用说明

1. 数据库结构概述

1.1 数据库文件

- **数据库路径**：`G:\LabInventory\lab_inventory.db`
- **数据库管理**：所有操作通过 Python 脚本进行，不直接在数据库管理工具中操作。

1.2 表结构

数据库包含以下主要表：

1.2.1 `parts` — 元器件信息

- **描述**：存储所有物料（元器件）的基本信息。
- **字段**：
  - `mpn`：物料编号（唯一）
  - `name`：物料名称
  - `category`：物料类别（如电阻、电容等）
  - `package`：封装类型
  - `params`：物料参数（JSON 或文本）
  - `datasheet`：数据手册的 URL 或本地路径
  - `note`：备注
  - `created_at`：创建时间

1.2.2 `stock` — 库存信息

- **描述**：存储每个物料在不同库位的库存数量。
- **字段**：

- `part_id`：物料 ID，关联 `parts` 表
- `location`：库存所在库位
- `qty`：库存数量
- `condition`：库存状态（如新旧状态）
- `note`：备注
- `updated_at`：更新时间

1.2.3 `locations` — 库位信息

- **描述**：存储所有库存库位信息（如：C409-G01-S01-P01）。
- **字段**：
  - `location`：库位编码
  - `note`：库位描述（如：柜子型号、位置）

1.2.4 `projects` — 项目信息

- **描述**：存储所有项目的基本信息。
- **字段**：
  - `code`：项目代码（唯一）
  - `name`：项目名称
  - `owner`：负责人
  - `status`：项目状态（active/archived）
  - `note`：备注
  - `created_at`：创建时间

1.2.5 `project_bom` — 项目物料清单（BOM）

- **描述**：存储项目所需的物料及数量。
- **字段**：
  - `project_id`：项目 ID，关联 `projects` 表
  - `part_id`：物料 ID，关联 `parts` 表
  - `req_qty`：需求数量
  - `priority`：优先级（1=高，2=中，3=低）
  - `note`：备注

1.2.6 `project_alloc` — 项目物料预留

- **描述**：存储项目预留的物料信息，并允许按库位分配库存。
- **字段**：
  - `project_id`：项目 ID，关联 `projects` 表
  - `part_id`：物料 ID，关联 `parts` 表
  - `location`：库位
  - `alloc_qty`：预留数量
  - `status`：预留状态（reserved, released, consumed）
  - `note`：备注
  - `updated_at`：更新时间

---

2. 数据库初始化与表创建

在第一次运行时，确保数据库和表结构已正确创建。可以使用以下脚本初始化数据库：

代码块

```
def init_db(conn: sqlite3.Connection):
    conn.executescript(DDL)  # 执行数据库创建脚本
    conn.commit()
```

DDL（数据定义语言） 是表结构创建和初始化的 SQL 脚本。

2.1 表结构和约束

- 所有表均使用 **外键约束** 来保证数据的完整性（如 `parts` 和 `stock` 中的 `part_id`，`projects` 和 `project_bom` 中的 `project_id`）。
- 使用触发器来保证 **库存预留不能超出库存数量**，且库位存在于 `locations` 表中。

2.2 强约束触发器

- **禁止超预留**：通过触发器，**在插入或更新 `project_alloc` 表时，自动检查库存是否足够**。

---

3. Python 脚本使用说明

3.1 依赖安装

首先安装脚本依赖库：

```
pip install requests beautifulsoup4 lxml
```

3.2 运行脚本操作

脚本的交互方式通过命令行参数实现。以下是常用功能及命令：

3.2.1 从立创导入物料并下载数据手册

通过商品链接导入物料信息并自动下载数据手册：

代码块

```
python G:\LabInventory\inv.py --db "G:\LabInventory\lab_inventory.db" lcsc --url "https://item.szlcsc.com/8143.html"
```

3.2.2 入库（按库位）

将物料入库到指定库位：

代码块

```
python G:\LabInventory\inv.py --db "G:\LabInventory\lab_inventory.db" stock-in --mpn "LM1117-3.3" --loc "C409-G01-S01-P01" --qty 10
```

3.2.3 创建新项目

创建一个新项目：

代码块

```
python G:\LabInventory\inv.py --db "G:\LabInventory\lab_inventory.db" proj-new --code "PJ-001" --name "无线充电发射端V1"
```

3.2.4 设置项目的 BOM

设置项目所需的物料清单（BOM）：

代码块

```
python G:\LabInventory\inv.py --db "G:\LabInventory\lab_inventory.db" bom-set --proj "PJ-001" --mpn "LM1117-3.3" --req 2
```

3.2.5 带库位预留

将物料按项目预留到指定库位（强约束，防止超预留）：

代码块

```
python G:\LabInventory\inv.py --db "G:\LabInventory\lab_inventory.db" reserve --proj "PJ-001" --mpn "LM1117-3.3" --loc "C409-G01-S01-P01" --qty 2
```

3.2.6 释放预留

释放物料的预留状态：

代码块

```
python G:\LabInventory\inv.py --db "G:\LabInventory\lab_inventory.db" release --id 1
```

3.2.7 消耗物料

标记物料为已消耗，并扣减库存：

代码块

```
python G:\LabInventory\inv.py --db "G:\LabInventory\lab_inventory.db" consume --id 1
```

3.2.8 查看项目备料状态

查看某个项目的备料状态，包括需求、库存、已预留等：

代码块

```
python G:\LabInventory\inv.py --db "G:\LabInventory\lab_inventory.db" proj-status --proj "PJ-001"
```

3.2.9 查看项目预留明细（带库位）

查看项目的所有物料预留记录及库位信息：

代码块

```
python G:\LabInventory\inv.py --db "G:\LabInventory\lab_inventory.db" proj-alloc --proj "PJ-001"
```

3.2.10 初始化库位

初始化库位（C409 两个柜子共 40 个库位）：

```
python G:\LabInventory\inv.py --db "G:\LabInventory\lab_inventory.db" init-locations --room C409
```

---

4. 库位初始化与项目管理

4.1 库位初始化

运行 `init-locations` 命令后，**自动生成库位**（根据给定的房间号、柜子信息）并插入到 `locations` 表中。

- 默认库位：`C409-G01-S01-P01` ~ `C409-G01-S03-P10`（3 层柜子，共 30 个位置）
- 你可以调整 `positions` 和 `shelves` 来满足不同场景需求。

4.2 项目管理

- **创建项目**：为每个项目生成一个唯一的 `project_id`，并将项目所需的物料通过 BOM 关联。
- **物料预留**：通过 `reserve` 命令进行物料的预留操作，库存将根据物料 `part_id` 和库位 `location` 扣减。

---

5. 常见问题与注意事项

5.1 库位管理

- **库位合法性**：每个库位必须在 `locations` 表中提前注册，且物料预留时必须指定有效的 `location`。
- **强约束**：如果预留超出库存（全局或库位层面），系统会报错并拒绝插入。

5.2 库存与消耗

- **库存扣减**：物料标记为 `consumed` 时，库存数量将从 `stock` 表中相应库位扣减。请确保库存不为负数。

5.3 数据手册下载

- **数据手册**：脚本会自动从立创页面下载 PDF 手册并保存在本地目录（默认为 `G:\LabInventory\datasheets`），并将路径存储在 `datasheet` 字段中。

---

6. 扩展与优化

- **细粒度预留（库位）**：如果后期需要更复杂的“物料预留”管理，可以扩展到每个物料在多个库位的占用情况。
- **数据追溯**：在 `project_alloc` 表中记录了每次物料预留、释放和消耗的时间、备注等信息，有助于后期的审计和追溯。

完成！你现在可以通过 Python 脚本 管理所有操作了。

通过这个系统，你能高效地进行 **库存管理**、**项目管理**、**物料预留**，并且一切都在本地数据库中自动同步，极大提升了数据的完整性与可追溯性。