图书馆管理系统软件详细设计说明书 (SDD) v2

项目名称：图书馆管理系统 **依据文档**：《图书管理数据设计(MD版-Ver2.0)》、《DFD描述》 **技术栈**：Flask (Python) + SQLite + HTML/CSS/JS

软工四组：李炅阳、潘语博、张乐平、范奇俊

1. 引言 (Introduction)

1.1 编写目的

本文档旨在为软件开发团队提供一份详尽的、可执行的系统设计蓝图。它明确了系统的架构、数据库映射、各子系统的详细处理逻辑、接口定义及异常处理机制，确保开发出的软件严格符合数据流图 (DFD) 的业务逻辑及数据设计的约束。

1.2 范围与约束

- **核心业务**：覆盖采访、编目、流通、用户管理、Web交互、维护、统计七大子系统。
- **数据一致性**：系统必须保证库存、借阅状态、读者信用等核心数据在并发操作下的强一致性。
- **逻辑约定**：对于DFD中存在但数据设计中未定义的“新书通报表”和“超期通报表”，本设计采用 **动态查询视图 (Logical View)** 方案实现，不新增物理表，以减少数据冗余。

2. 系统架构设计 (System Architecture)

2.1 技术架构

采用 **B/S (Browser/Server)** 分层架构，为适应课程作业的轻量化需求，数据库选用 SQLite：

- **表现层 (Presentation Layer)**: 对应 DFD 1.5 Web子系统及后台管理界面。
- **业务逻辑层 (Business Logic Layer)**: 实现 DFD 1.1 ~ 1.7 的核心加工逻辑。
- **数据持久层 (Data Persistence Layer)**: 基于 SQLite，严格对应物理表设计。

2.2 模块划分

M01 采访管理模块 (DFD 1.1) **M02 编目管理模块** (DFD 1.2) **M03 流通管理模块** (DFD 1.3)
M04 用户管理模块 (DFD 1.4) **M05 Web交互门户** (DFD 1.5) **M06 系统维护模块** (DFD 1.6)
M07 统计分析模块 (DFD 1.7)

3. 数据库设计映射 (Database Mapping)

基于《图书管理数据设计(MD版)》，我们将 DFD 中的“数据存储”映射为具体的物理表：

DFD 数据存储	物理表名 (Table Name)	备注
采访清单	ACQ_SUGGESTION (2.1)	包含读者荐购和采编录入
订单/字典	ORDER_HEAD(2.2), ORDER_LINE(2.3), SUPPLIER(1.1), PUBLISHER(1.2)	字典拆分为基础信息表
验收清单	ACCEPT_LIST (2.4)	合格品中间表
退货清单	RETURN_LIST (2.5)	不合格品记录
编目数据	CATALOG_BUFFER (3.1), CLC_TYPE (1.3)	编目缓冲及分类字典
图书流通库	CIRCULATION_HEAD (3.2头), CIRCULATION_DETAIL (3.2明细)	**核心资产表**
期刊库表	JOURNAL_HEAD (3.3头), JOURNAL_DETAIL (3.3明细)	
读者/处罚	READER (4.1), CREDIT_LEVEL (4.2), PUNISH_RECORD (5.3)	
管理员	ADMIN (4.3)	
借阅/预约	BORROW_RECORD (5.1), RESERVE_INFO (5.2)	
延期申请	EXTENSION_APP (5.5)	
公告/留言	NOTICE (6.1), MESSAGE_BOARD (6.2)	
新书/超期通报	**(无物理表)**	**逻辑视图**：通过查询 CIRCULATION_DETAIL 和 BORROW_RECORD 实时生成

4. 模块详细设计 (Module Design)

4.1 M01 采访管理子系统 (Acquisitions)

4.1.1 采访验收 (Process 1.1.1)

- **输入**：读者荐购信息（Web端传入）。
- **处理逻辑**：

1. 校验输入合法性（书名、ISBN）。
2. 写入 ACQ_SUGGESTION 表，字段 STATUS 默认为 0 (未处理)。
3. **排重检查**：若 ISBN 已存在于 CIRCULATION_HEAD，提示“馆内已有馆藏”，但允许继续荐购（作为复本依据）。

4.1.2 订单生成 (Process 1.1.2)

- **输入**：采编员勾选 `ACQ_SUGGESTION` 记录。
- **处理逻辑**：

1. 创建 `ORDER_HEAD` 记录：生成唯一 `ORDER_NO`，关联 `SUPPLIER_ID`，记录 `PURCHASER`。
2. 遍历勾选条目，生成 `ORDER_LINE` 记录：
    - 填入 ISBN, PRICE, QUANTITY 等书籍基本信息。
    - **关联荐购ID**：将 ACQ_SUGGESTION.ACQ_ID 写入 ORDER_LINE.SUGGESTION_ID 字段，确保后续能追踪到是为谁买的书。
3. 回写 `ACQ_SUGGESTION`：将对应记录的 `STATUS` 更新为 `1` (已加入订单)。
4. 更新 `ORDER_HEAD.TOTAL_PRICE`。

4.1.3 验收与退货 (Process 1.1.3 & 1.1.4)

（保持原版逻辑：更新 `ORDER_LINE` 状态，写 `ACCEPT_LIST` 或 `RETURN_LIST`）

4.2 M02 编目管理子系统 (Cataloging)

4.2.1 取采访与查重 (Process 1.2.1 & 1.2.2)

- **数据源**：查询 `ACCEPT_LIST` 中未被编目的记录。
- **核心逻辑**：

1. 读取 `ACCEPT_LIST` 关联的 ISBN。
2. **查重**：`SELECT COUNT(*) FROM CIRCULATION_HEAD WHERE ISBN = ?`
3. **分支处理**：
    - **复本编目**：若 Count > 0，自动带出 CALL_NUMBER、CLC_CODE 等头表信息，直接进入明细录入。
    - **新书编目**：若 Count == 0，需人工录入头表信息。

4.2.2 编目缓冲 (Process 1.2.3)

- **操作**：采编员完善 MARC 信息（分类号、作者、出版社等）。
- **存储**：数据暂时存入 CATALOG_BUFFER (TEMP_ID, ISBN, CALL_NUMBER, BOOK_NAME...)，允许反复修改，不污染正式库。

4.2.3 移送入库 (Process 1.2.4 & 1.2.5)

- **功能**：将缓冲区数据变为正式馆藏。
- **事务处理 (Transaction)**：

1. `BEGIN TRANSACTION`
2. **Merge Header**：
    - 检查 CIRCULATION_HEAD 中是否存在该 ISBN（ISBN现为主键）。
    - 若不存在，插入 CATALOG_BUFFER 中的头信息。
    - 若存在，可选择性更新元数据。
3. **Insert Detail**：根据 `ACCEPT_LIST` 的数量，循环插入 `CIRCULATION_DETAIL`：
    - 生成唯一 BARCODE (规则：ISBN + 4位流水 或 时间戳)。
    - 设置 LOCATION (馆藏地)。
    - 设置 STATUS = 1 (在馆)。
    - 设置 ENTRY_DATE = NOW()。
4. **Delete Buffer**：删除 `CATALOG_BUFFER` 对应记录。
5. `COMMIT`

- **新书通报逻辑**：系统无需额外写表，Web端直接查询 CIRCULATION_DETAIL 中 ENTRY_DATE 为最近 30 天的记录即可。

4.2.4 注销报损 (Process 1.2.6)

- **输入**：`BARCODE`，注销类型（报损/丢失/剔旧）。
- **逻辑**：

1. 检查该书是否在借 (STATUS=2)，若在借禁止直接报损（需先还书或走遗失赔偿流程）。
2. 更新 CIRCULATION_DETAIL.STATUS 为 4(报损) 或 5(遗失)。
3. 插入 LOSS_RECORD (报损_注销记录)。

4.3 M03 流通管理子系统 (Circulation)

4.3.1 借书 (Process 1.3.1)

- **输入**：`READER_ID`，`BARCODE`。
- **前置校验链**：

1. **读者状态**：READER.STATUS != 0 (封禁) 且 READER.EXPIRY_DATE > NOW()。
2. **权限额度**：SELECT COUNT(*) FROM BORROW_RECORD WHERE READER_ID=? AND STATUS=1 < CREDIT_LEVEL.MAX_BORROW_NUM。
3. **信用分检查**：(新增) READER.CURRENT_CREDIT > 0 (或其他最低借书分)。
4. **图书状态**：CIRCULATION_DETAIL.STATUS 必须为 1 (在馆)。若为 3 (预约)，检查预约人是否为当前读者。

- **执行逻辑**：

1. 插入 BORROW_RECORD。
2. 更新 CIRCULATION_DETAIL：STATUS=2 (借出)。
3. 若存在对应的预约记录，更新其 STATUS=3 (已取)。

4.3.2 还书与罚款 (Process 1.3.2)

- **输入**：`BARCODE`。
- **执行逻辑**：

1. 查询未还记录。
2. **逾期计算**：
   - 若逾期，计算罚金，插入 PUNISH_RECORD。
   - **触发扣分**：调用 M04 模块的信用扣分接口（如每逾期1天扣1分）。
3. **还书落库**：
   - 更新 BORROW_RECORD 和 CIRCULATION_DETAIL。
4. **预约触发**：
   - 查询 RESERVE_INFO 是否有人排队该 ISBN。
   - 若有，将书即刻改为 STATUS=3 (预约留书)，**锁定给该预约人**。

4.3.3 预约管理

- **预约动作**：仅允许对 `STATUS`!=1（非在馆）的书发起。写入 `RESERVE_INFO`。
- **到期扫描（定时任务）**：
  - **预约未取**：扫描 `RESERVE_INFO` 中 `STATUS=2`（待取书）且 `EXPIRE_DATE` < NOW() 的记录。
    - 动作：释放图书 (CIRCULATION_DETAIL.STATUS=1)，记录爽约罚款 (金额0，记入信用扣分)，更新预约 STATUS=4 (失效)。

4.4 M04 用户管理子系统 (User Mgmt)

4.4.1 信用与降级 (Process 1.4.10) - 核心修改

- **逻辑**：

  1. **监听违规**：系统监听 `PUNISH_RECORD` 插入事件或逾期事件。
  2. **扣分操作**：
     - 读取当前分：SELECT CURRENT_CREDIT FROM READER WHERE READER_ID=?
     - 执行扣减：UPDATE READER SET CURRENT_CREDIT = CURRENT_CREDIT - ? WHERE READER_ID=?
  3. **降级检查**：
     - 获取当前等级的阈值：`SELECT MIN_SCORE FROM CREDIT_LEVEL WHERE LEVEL_ID = (SELECT LEVEL_ID FROM READER ...)`
     - **判断**：若 `CURRENT_CREDIT` < `MIN_SCORE`：
       - 查找下一级 LEVEL_ID（如从研究生降为本科）。
       - 更新 READER.LEVEL_ID。
       - 发送系统消息通知读者“您的信用等级已下降”。

4.4.2 延期审批 (Process 1.4.11)

- **输入**：管理员操作 `EXTENSION_APP`。
- **同意**：
  - 更新 EXTENSION_APP.STATUS = 1。
  - 级联更新 BORROW_RECORD.DUE_DATE += 延期天数 (配置值)。
- **拒绝**：更新 `EXTENSION_APP.STATUS` = 2。

4.5 M05 Web交互子系统

4.5.1 图书检索 (Process 1.5.5)

- **SQL实现**：

```sql
SELECT h.*,
       (SELECT COUNT(*) FROM CIRCULATION_DETAIL d WHERE d.ISBN = h.ISBN
AND d.STATUS = 1) as available_count
FROM CIRCULATION_HEAD h
WHERE h.BOOK_NAME LIKE ? OR h.AUTHOR LIKE ?
```

逻辑说明：基于 `CIRCULATION_HEAD`（主键表）进行检索，关联计算在馆数量。

4.5.2 个人中心

- **我的借阅**：查询 BORROW_RECORD。
- **我的积分**：显示 READER.CURRENT_CREDIT 及当前等级名称。

4.6 M06 维护子系统

4.6.1 备份策略 (Process 1.6.1)

- **逻辑**：读取 BACKUP_CONFIG。系统调用 mysqldump 工具，将指定 TABLE_NAME 导出到 PATH 目录。
- **文件名规范**：Table_YYYYMMDD.sql。

4.7 M07 统计子系统

4.7.1 动态报表引擎

- **设计**：系统不硬编码统计SQL。
- **执行流**：

1. 前端请求统计代码 STAT_CODE (如 "TOP10_BORROW")。
2. 后端查 STAT_PRESET 表获取 SQL_LOGIC。
3. 执行 SQL，返回 JSON 数据。
4. 前端根据 CHART_TYPE (Pie/Bar) 渲染。

---

5. 接口设计 (API Specification)

所有接口采用 RESTful 风格，返回 JSON 格式。

方法	URL 路径	功能描述	关键参数
POST	/api/auth/login	用户登录	username, password
GET	/api/books	图书搜索	q (关键词), page
POST	/api/books/reserve	图书预约	isbn, readerId
POST	/api/circulation/borrow	借书操作	barcode, readerId
POST	/api/circulation/return	还书操作	barcode
GET	/api/profile/borrows	当前借阅列表	readerId
GET	/api/profile/credit	**信用积分**	readerId (新增接口)
POST	/api/admin/acq/order	生成订单	suggestionIds[], supplierId
GET	/api/stats/{code}	获取统计数据	code

6. 安全与异常处理 (Security & Exception)

6.1 并发控制 (Concurrency)

- **库存扣减**：在借书和预约环节，必须使用数据库事务 (Transaction) 并加行锁。
- **积分扣减**：扣分操作必须原子化，防止多线程下分数计算错误。

6.2 权限验证 (Auth)

- **RBAC模型**：基于 ADMIN.ROLE 和 READER 身份。

6.3 数据完整性

- **外键约束**：所有表定义必须建立物理外键。
- **SQL注入防护**：统计子系统中的 STAT_PRESET 表写入权限仅开放给最高管理员，防止恶意SQL注入。

7. 附录 (Appendix)

7.1 状态码定义

- Book Status: 1-在馆, 2-借出, 3-预约, 4-报损, 5-遗失
- Acq Status: 0-未处理, 1-已下单, 2-驳回
- Borrow Status: 1-借出, 2-已还, 3-超期

修订记录:

- Ver 1.0: 初始版本。
- Ver 1.1: 修正了信用分字段缺失、头表主键定义错误及荐购关联丢失的逻辑缺陷。