# 缓冲池管理实验总结

## 实验概述

本次实验主要涉及数据库缓冲池管理系统的两个核心组件：**LRU替换策略**和**缓冲池管理器**。通过分析和实现这些组件的关键方法，深入理解了数据库系统中内存管理的核心机制。

## 任务一：LRU替换策略实现

### 主要实现方法

* **LRUReplacer::victim()**: 选择最久未使用的页面进行淘汰
* **LRUReplacer::pin()**: 将页面标记为不可淘汰
* **LRUReplacer::unpin()**: 将页面加入可淘汰列表

### 核心技术要点

1. **数据结构设计**: 采用双向链表+哈希表的组合，实现O(1)时间复杂度的LRU操作
2. **线程安全**: 使用std::scoped\_lock确保并发访问的安全性
3. **LRU策略**: 链表头部存放最近访问页面，尾部存放最久未访问页面

### 实现难点

* 理解pin/unpin机制与LRU策略的关系
* 正确维护链表和哈希表的同步更新

## 任务二：缓冲池管理器实现

### 主要实现方法

* **find\_victim\_page()**: 查找可用frame（优先空闲帧，其次LRU淘汰）
* **fetch\_page()**: 页面获取（缓冲池命中或磁盘读取）
* **unpin\_page()**: 页面取消固定
* **flush\_page()**: 单页面刷新到磁盘
* **new\_page()**: 创建新页面
* **delete\_page()**: 删除页面
* **flush\_all\_pages()**: 批量页面刷新

### 核心技术要点

1. **页面生命周期管理**: 从磁盘读取→缓冲池驻留→写回磁盘→资源回收
2. **脏页处理**: 确保修改过的页面及时写回磁盘
3. **资源调度**: 空闲帧管理与页面替换策略的协调
4. **元数据维护**: page\_table映射关系、pin\_count引用计数

### 实现难点

* 页面置换时的脏页写回处理
* pin\_count机制与replacer的协调
* 多线程环境下的数据一致性保证

## 技术收获

1. **深入理解缓冲池原理**: 掌握了数据库系统中内存管理的核心机制，包括页面缓存、替换策略、脏页管理等
2. **并发编程实践**: 学会使用互斥锁保证多线程环境下数据结构的安全访问
3. **系统设计思维**: 理解了高效数据结构设计的重要性，如LRU的O(1)实现
4. **资源管理**: 掌握了系统资源（内存、磁盘）的有效调度和管理策略

## 实验意义

通过本次实验，全面理解了数据库系统缓冲池管理的完整流程，为后续学习数据库内核技术奠定了坚实基础。实验中涉及的内存管理、并发控制、I/O优化等技术，在实际的数据库系统开发中具有重要的应用价值。