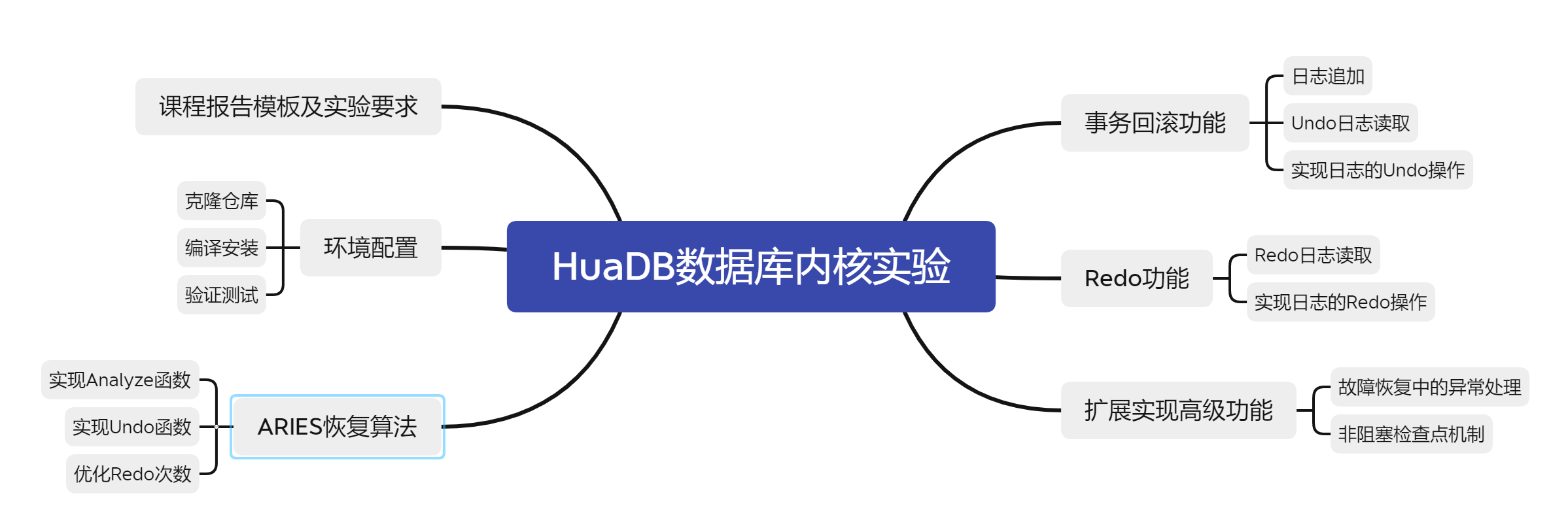
实验七：

HuaDB数据库内核实验



本实验概览图

# HuaDB数据库内核实验手册

* 1. 实验介绍
     1. 关于本实验

本次实验为事务处理与故障恢复的实验，意图通过实现数据库日志机制和经典故障恢复算法来更好地理解数据库系统维持事务ACID特性的底层实现。事务处理事数据库系统最核心的功能之一。数据库系统必须保证事务处理的任意过程中意外的系统宕机不会影响事务的ACID特性，但出于性能因素考量的steal+no-force的脏页刷盘机制让故障情况下磁盘页面的数据违背原子性和持久性。此时，事务处理过程中的日志机制和故障后的恢复算法就起到了至关重要的作用。过顺序追加的日志记录和故障恢复算法对于页面状态的恢复在保证了系统高效性的同时维持了事务的正常运行。日志格式的设计和故障恢复算法实现是本次实验的难点。日志设计上需要考虑自身保存的数据需要能够支持重做（Redo）或撤销（Undo）操作，并且需要兼顾存储空间和恢复速度（物理和逻辑日志的差异）。故障恢复算法则需要避免记录无用的日志记录以此来维持尽可能低的故障恢复时间。同时，日志和故障恢复的机制要求能覆盖系统在任意时刻宕机的异常情景。本次实验要求完成物理日志的设计和经典的 ARIES 故障恢复算法。图1为数据库系统整体层次划分图，左侧为功能示意图，右侧为层次架构图。

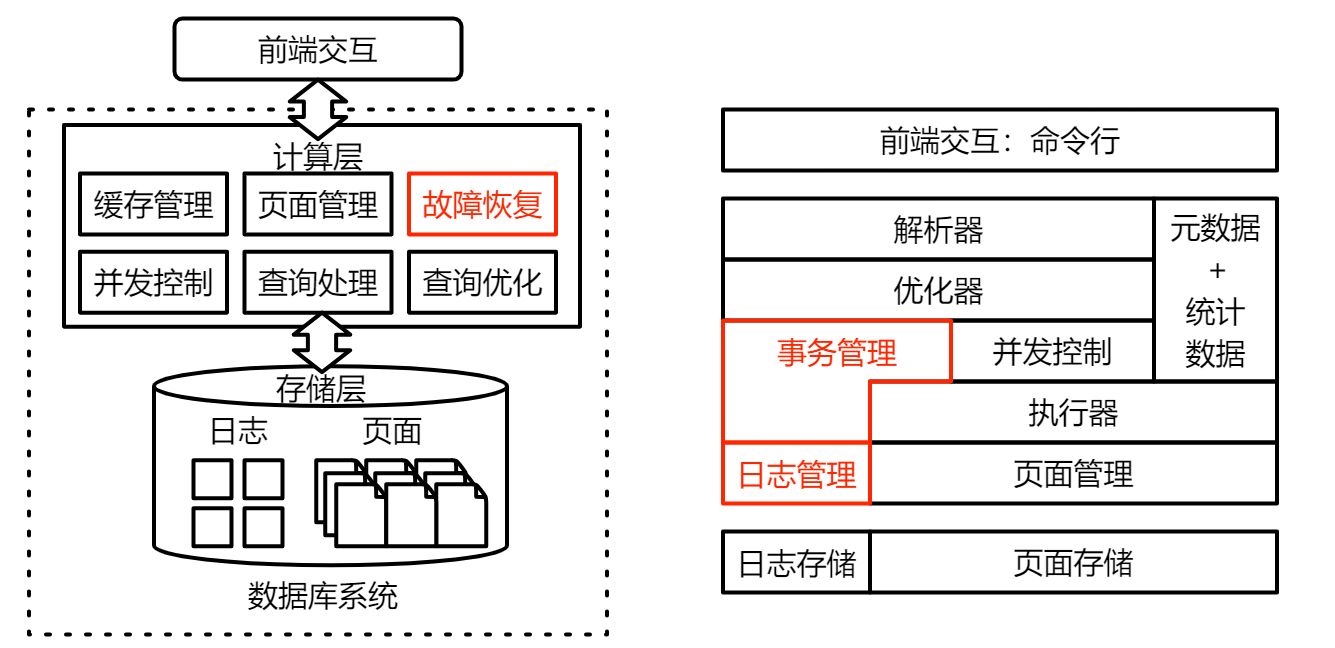


图1 数据库系统整体层次划分图

* + 1. 实验目标

本次实验要求完成如下基础功能：

* 事务的回滚：实现事务的 rollback 操作。
* 物理日志的设计：设计物理日志的存储格式，并实现单个日志的重做和撤销函数。
* ARIES 故障恢复算法：补全分析、重做、撤销三个阶段的函数实现，实现 ARIES 故障恢复算法。

**（扩展）**在基础功能之上，实验框架支持完成以下高级功能：

故障恢复过程中的异常处理：实验框架中仅考虑系统仅考虑系统在查询执行中的异常处理，在此基础上添加恢复过程中的异常处理机制，核心为补偿日志机制的实现。

非阻塞检查点机制：实验框架中 Checkpoint 采用阻塞的信息写出方式，在此基础上单开线程实现非阻塞的 Checkpoint 机制，核心为修改 BeginCheckpointLog 和 EndCheckpointLog 的实现。

* 1. 实现事务处理与故障恢复功能概述
     1. 实现事务回滚功能

本次实验中，你首先需要实现事务回滚的功能，主要涉及 log\_manager.cpp 的 Rollback 函数，insert\_log.cpp, delete\_log.cpp 和 new\_page\_log.cpp 的 Undo 函数，此外，还需要修改 lab 1 中对 table.cpp 的实现，添加日志相关部分，并在 table\_page.cpp 中增加 RedoInsertRecord 和 UndoDeleteRecord 的逻辑，完整实现后可以通过10-rollback.test测例。

**步骤1：日志追加**

实验框架通过读取日志的方式来实现事务回滚，当一个事务执行 rollback 语句时，数据库会逆序读取该事务产生的日志记录，将该事务所作的修改操作按倒序依次撤销。需要注意的是，实验框架采用了与课程讲授内容不同的方法，undo 日志和 redo 日志均使用物理日志。这一方面是为了减少本次实验的工作量，另一方面在实验框架的存储设计中，undo 日志可以采用物理日志来记录，你可以思考一下在实验框架中为何使用物理日志进行 undo 操作不会影响正确性。

要在事务执行 rollback 时完成回滚操作，我们必须保证事务从开始到回滚之间所作的页面修改操作都记录在日志中，正确记录日志是实现回滚的重要前提，因此第一步，我们需要修改 table.cpp 的 InsertRecord 和DeleteRecord 函数实现，在 write\_log 参数为 true 时，添加日志写入的代码。

与页面修改操作相关的日志共三种：

|  |  |  |
| --- | --- | --- |
| 日志名称 | 类名 | 记录的信息 |
| 记录插入日志 | InsertLog | 插入记录所属表的 oid，表所属数据库的 oid，插入数据的 page id，slot id，记录在页面中的偏移量，记录长度，以及记录的序列化数据 |
| 记录删除日志 | DeleteLog | 删除记录所属表的 oid，表所属数据库的 oid，删除数据的 page id，slot id |
| 新页日志 | NewPageLog | 新页所属表的 oid，表所属数据库的 oid，前一页的 page id，新页的 page id |

具体而言，你需要在对页面进行修改操作（记录插入、记录删除、新建页面）时，通过LogManager AppendInsertLog,AppendDeleteLog和AppendNewPageLog 函数进行日志追加，这些函数会返回日志的 lsn，得到lsn后，你还需要设置页面的 page lsn，这将用于任务3ARIES 算法的实现。

**步骤2：Undo日志读取**

执行 rollback 语句时，数据库会调用 LogManager 的 Rollback 函数，这一步骤中，我们需要实现 Rollback 函数。在步骤 1 调用 AppendLog 函数的过程中，已帮你维护了 Logmanager 中的活跃事务表 att\_ 和脏页表 dpt\_ 变量，你可以在活跃事务表中查找到事务最后一条日志记录的 LSN，每条日志记录都有一个 prev\_lsn\_ 字段，表示该事务前一条日志记录的 LSN，通过活跃事务表 att\_ 和 prev\_lsn\_，便可以实现对一个事务相关日志记录的倒序遍历。

在课程中，LSN的表示方法使用递增的连续序号表示，而在实验框架中，LSN 使用的是日志记录在日志文件中的位置表示，这种表示方法可以方便地通过 LSN 直接获取日志，无需再单独存一份 LSN 到日志位置的映射表。

得到一条日志记录的 LSN 后，你需要判断 LSN 与 flushed\_lsn\_ 的大小关系。flushed\_lsn\_ 表示刷新到磁盘的最大 LSN，通过比较 LSN 与 flushed\_lsn\_，可以得知日志记录位于磁盘中还是 log 缓存中，进而从对应的位置获取到完整的日志记录。获取到日志记录后，调用日志记录的 Undo 函数进行回滚。

**步骤3：实现日志的Undo操作**

本步骤中，你将实现 InsertLog, DeleteLog和NewPageLog的Undo函数，以及TablePage类的 UndoDeleteRecord函数，来撤销事务对页面的修改。

对于InsertLog，你需要将插入的记录删除；对于DeleteLog，你需要清除记录的删除标记（调用 TablePage 类的 UndoDeleteRecord 函数）；对于NewPageLog，你需要重新设置前一页的next\_page\_id\_字段，正确实现这些函数后，你将可以通过测例10-rollback.test。

测例10-rollback.test分别进行了插入、删除、更新和新建页面的回滚测试，建议每实现完一种日志的回滚函数，进行一次测试，观察是否通过了对应操作的回滚测试。

* + 1. 实现Redo功能

本任务中，你将实现日志的redo函数，通过20-recover.test测例。

**步骤 1：Redo 日志读取**

你需要补充LogManager的Redo函数，从头至尾顺序读取日志，直至 flushed\_lsn\_，调用每个日志记录的Redo函数。

**步骤 2：实现日志的 Redo 操作**

与任务1的步骤3类似，你需要实现InsertLog,DeleteLog和NewPageLog的Redo函数，以及TablePage类的RedoInsertRecord函数，来重做事务对页面的修改。

由于我们没有记录DDL的日志，因此在重做过程中，你需要判断表的oid是否存在，如果不存在，说明日志记录对应的表已经被删除，无需进行重做。正确实现这些函数后，你将通过20-recover.test测例。

与测例10-rollback.test类似，测例20-recover.test分别对插入、删除、更新和新建页面操作进行了重做测试，每实现完一种重做日志便可以进行测试。

1.2.3 实现ARIES恢复算法

本任务中，你将实现完整的ARIES恢复算法，通过测例30-aries.test，该测例与课程PPT案例说明中的例子是一致的。

**步骤 1：实现Analyze函数**

首先，你需要实现Analyze函数，对应ARIES算法的分析阶段。我们已为你提供了读取master record部分的代码，你需要从checkpoint\_lsn开始，正序读取日志，读取过程中根据日志类型，来对活跃事务表att\_和脏页表中的数据dpt\_进行维护。

**步骤2：实现 Undo 函数**

经过 Analyze 阶段，你将得到数据库崩溃时的活跃事务表和脏页表，接下来，你将在 Undo 函数中调用任务 1 实现的 Rollback 函数将活跃事务回滚。

完成以上步骤后，在30-aries.test中，数据库恢复后将读取到正确的数据。

**步骤 3：优化 Redo 次数**

你需要根据分析阶段维护的脏页表，在redo时根据脏页表中的rec\_lsn\_，页面的 page\_lsn\_，以及日志的 lsn 的大小关系，判断 redo 操作是否有必要进行。每进行一次 redo 操作，你需要调用 LogManager 的 IncrementRedoCount 函数，统计 redo 操作次数。

正确实现本步骤后，你将通过测例30-aries.test，从而完成本次实验。

* + 1. 高级功能概述

**（1）任务 1：故障恢复中的异常处理**

【实验描述】

基础功能仅考虑到查询执行过程中的异常处理过程，高级功能要求在此基础上考虑到故障恢复过程中出现异常时的处理流程，需要引入补偿日志的机制来保证 Undo 过程异常恢复的正确性和执行效率。

【实现思路】

步骤 1：修改数据库故障恢复逻辑，在数据库故障恢复过程中随机发生故障。

步骤 2：理解课程中撤销阶段异常恢复的特殊性，理解补偿日志的机制。并在故障恢复算法中添加补偿日志的记录和处理能力。

**（2）任务 2：非阻塞检查点机制**

【实验描述】

基础功能中检查点日志按照阻塞的方式追加到磁盘，当 ATT 和 DPT 表容量较大时将影响到系统的查询处理运行，在高级功能中要求采用多线程的方式添加对于非阻塞检查点功能的支持。

【实现思路】

步骤 0：理解实验框架中阻塞检查点的信息记录方法，思考非阻塞检查点的信息记录方式。

步骤 1：修改 LogManager::Checkpoint 函数，修改 BeginCheckpointLog 和 EndCheckpointLog 的实现，确定 ATT 和 DPT 实际存储位置。

步骤 2：修改 LogManager 类，添加多线程检查点任务结束的异步通知机制，添加多线程情况下的检查点完成的处理函数。

1.2.5 代码结构

实验框架主要包含以下几个模块，加粗部分为实验中涉及到的，需要你来补充的模块：

* binder: 语义解析模块
* catalog: 系统表模块
* common: 工具模块，包含字符串处理函数、异常相关类等
* database: 数据库引擎
* **executors: 查询执行模块**
* **log: 日志模块**
* operators: 查询计划树节点
* **optimizer: 优化器**
* planner: 查询计划生成模块
* **storage: 存储模块**
* **table: 表相关类及函数**
* **transaction: 事务模块**
  + 1. 相关代码模块

本次实验涉及到代码中如下的功能模块：

|  |
| --- |
| * [log](https://github.com/thu-db/huadb/tree/main/src/log)：日志相关类   + [log\_record](https://github.com/thu-db/huadb/tree/main/src/log/log_record.h)：各类日志的抽象类，用于衍生出各种具体日志，已经完成。   + [log\_manager](https://github.com/thu-db/huadb/tree/main/src/log/log_manager.h)：日志管理器，负责日志的记录以及故障恢复的具体执行过程，需要补充恢复算法。   + [log\_records](https://github.com/thu-db/huadb/tree/main/src/log/log_records)：各类具体的物理日志类，需要补充其中记录增删改日志的重做和撤销。 * [table](https://github.com/thu-db/huadb/tree/main/src/table)：数据表相关类   + [table](https://github.com/thu-db/huadb/tree/main/src/table/table.h)：在实验 1 的基础上添加记录变更时的日志记录功能。 |

基础功能需要补充约 200 行代码，本次实验依赖于实验 1，请确保完成以上任务再开始本次实验。

相关功能模块的抽象示意图如图2：

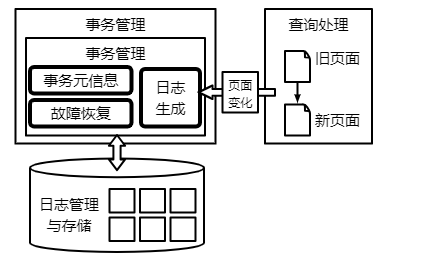


图2 相关功能模块的抽象示意图

* 1. 前置步骤

本实验均在HuaDB上进行，由C++语言编写，支持基础的页面存储、故障恢复、缓存管理、查询处理等功能。该实验提供HuaDB基础实验框架，在实验2中，需要在框架中填充相应的函数实现代码，使之通过该次实验的所有测例。实验前，需要先执行环境配置、克隆仓库、编译及测试等步骤。

### 环境配置

HuaDB 使用了 C++17 标准，开始实验前，请确保你的开发环境支持 C++17 标准。

目前 HuaDB 仅支持 macOS 和 Linux 操作系统，使用 Windows 的同学建议使用虚拟机进行实验。

HuaDB 代码下载与提交需要使用[git](https://git-scm.com/)工具，代码编译需要使用[CMake](https://cmake.org/)及[Make](https://www.gnu.org/software/make/)工具，且需要安装[gcc](https://gcc.gnu.org/)或[clang](https://clang.llvm.org/)编译器。此外，代码调试中可能会用到调试器[gdb](https://www.sourceware.org/gdb/)或[lldb](https://lldb.llvm.org/)。开始实验前，请确保你的开发环境安装了这些工具并可以正常使用，如没有，请根据你使用的操作系统选择对应的命令进行环境配置。

Ubuntu

|  |
| --- |
| sudo apt install git g++ make cmake gdb |

### 克隆仓库

从GitHub克隆代码，命令如下：

|  |
| --- |
| git clone https://github.com/thu-db/huadb.git |

克隆成功后，进入huadb目录下：

|  |
| --- |
| cd huadb |

### 编译安装

修改环境变量，命令如下：

|  |
| --- |
| vim CMakeLists.txt  set(CMAKE\_CXX\_FLAGS "-pthread")  CMAKE\_BUILD\_PARALLEL\_LEVEL=$(nproc) make |

通过以上指令可以编译出 debug 版本的程序，为便于调试，不建议编译 release 版本。

编译生成的文件位于 build/debug 目录，实验过程中只需要关注build/debug/bin 目录中的可执行程序即可，具体包括：

1. shell: 数据库程序，运行后可以与数据库进行交互。
2. sqllogictest: 测试程序，用于批量测试。

### 验证测试

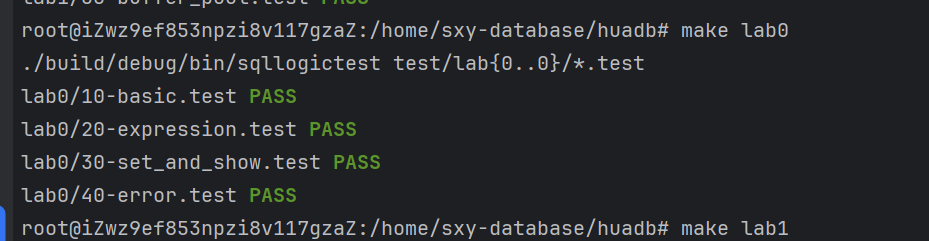
运行如下命令来验证你编译出的数据库程序和测试程序可以正常运行：

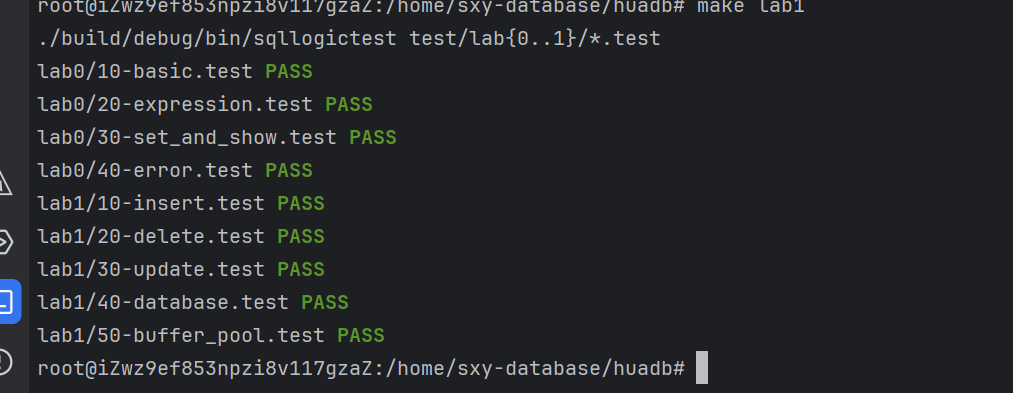
|  |
| --- |
| make lab0 |

如果产生如下输出，表示程序正常运行。

|  |
| --- |
| Test: 1/3  lab0/10-basic.test PASS  lab0/20-expression.test PASS  lab0/30-set\_and\_show.test PASS  lab0/40-error.test PASS  Test: 2/3  lab0/10-basic.test PASS  lab0/20-expression.test PASS  lab0/30-set\_and\_show.test PASS  lab0/40-error.test PASS  Test: 3/3  lab0/10-basic.test PASS  lab0/20-expression.test PASS  lab0/30-set\_and\_show.test PASS  lab0/40-error.test PASS |

如果程序报错，可以根据报错信息对照[测试说明](https://thu-db.github.io/huadb-doc/overview)进行排查。





此外，你可以通过如下命令进入数据库交互界面：

|  |
| --- |
| make shell |

你可以在交互界面中运行一些基础的 DDL 命令，如：

|  |
| --- |
| Welcome to HuaDB. Type "\?" or "\h" for help  huadb> \?  \? or \h show help message  \c [database\_name] change database  \d show tables  \d [table\_name] describe table  \l show databases  \q quit  huadb> \l  +---------------+  | database\_name |  +---------------+  | tmp |  +---------------+  (1 row)  huadb> create table test(id int, info varchar(10));  CREATE TABLE  huadb> \d  +------------+  | table\_name |  +------------+  | test |  +------------+  (1 row)  huadb> \d test  +------+---------+------+  | name | type | size |  +------+---------+------+  | id | int | 4 |  | info | varchar | 10 |  +------+---------+------+  (2 rows)  huadb> drop table test;  DROP TABLE  huadb> \d  +------------+  | table\_name |  +------------+  (0 rows)  huadb> \q |

# 课程报告模板

**数据库课程实验报告**

**(2023-2024学年第一学期)**

**提交日期：年 月 日 本科生姓名：**

|  |  |  |  |
| --- | --- | --- | --- |
| **学 号** |  | **学 院** |  |
| **课程名称** |  | **任课老师** |  |
| **教师评语：** | | | |
| **成绩评定： 分 任课教师签名： 年 月 日** | | | |

# 实验七：事务处理与故障恢复

参考https://thu-db.github.io/huadb-doc/的实验二部分

https://gitee.com/thu-db/huadb

# 一、实验要求

1、完成HuaDB数据库内核[实验2：事务处理与故障恢复](https://thu-db.github.io/huadb-doc/lab2/1-intro/)的基础功能部分，按照功能点描述基础功能的实现思路（如果实现思路与文档一致，简略描述即可，如果实现思路与文档不同，可以重点描述）。如果有新增的成员变量与成员函数，简要概括其作用，以及在实现过程中遇到的难点。

2、在实验过程中你遇到什么困难和问题，你怎么解决的？（没有则可以不写）

3、扩展：[尝试实现高级功能](https://thu-db.github.io/huadb-doc/lab2/3-advanced/)，若无法实现，说说你的遇到的问题。如果在本次实验中完成了高级功能，应将此部分作为报告的重点内容，详细阐述高级功能的设计思路与实现方法。可以按照如下流程进行介绍：

(1)相关原理与示意图：总结高级功能对应的知识点，并绘制示意图。

(2)新增类的描述：介绍在基础框架之上添加的新的类的功能，以及和已有一些类的关联关系。

(3)新增成员函数的描述：介绍新增或已有类上增加的重要成员函数，以及这些函数的功能描述和调用关系。

(4)高级功能效果展示：为高级功能添加测试用例，说明新增的高级功能的正确性或优化效果。