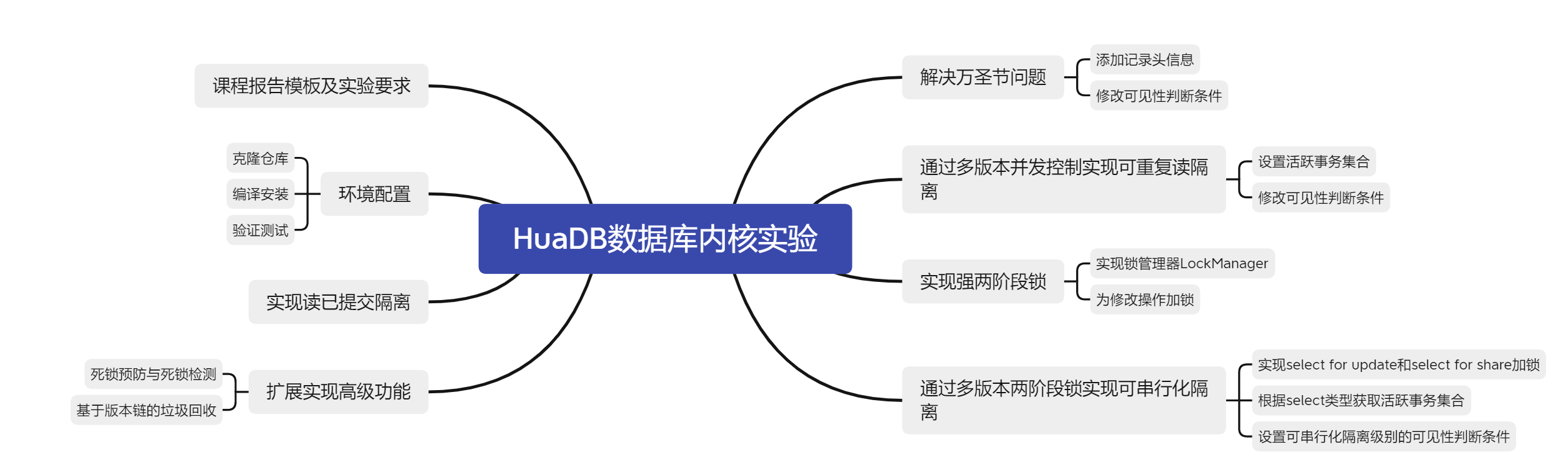
实验八：

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

本次实验为数据库系统并发控制的实验，意图通过实现多版本并发控制机制来更好地理解数据库系统如何以非阻塞方式处理并发查询中的读写和写读冲突，以及如何通过两阶段锁处理写写冲突。

并发查询处理是数据库系统维持高效率、高性能运行的必要条件。但是在并发查询处理中不可避免地会遇到不同事务同时访问相同数据区域的情景，可能发生读写、写读或者写写冲突。虽然严格按照读写锁的机制执行查询可以保证查询处理的绝对正确性，但是阻塞的处理方式将让查询性能受到严重限制。在此基础上，为了避免读写和写读冲突造成的系统阻塞问题，多版本并发控制技术得以发展。通过维持多个版本的记录信息，结合版本可见性的控制，数据库系统可以在无阻塞处理读写和写读冲突。

并发控制信息的维护和记录可见性分析是本次实验的难点。多版本并发控制依赖于记录所额外存储的版本信息，这要求在记录发生变更时需要同时更新记录的版本信息；同时需要实现记录级别或者页面级别的锁机制来应对写写冲突。在多版本并发控制的架构中，单个记录可以同时维持多个不同的版本来应对不同的事务上下文环境，读取的效率一方面要求系统能够根据版本信息和事务上下文快速确定不同记录版本对于特定事务的可见性，另一方面随着记录版本数量的增加，高效的记录查找机制和过期数据的清理机制才能保证查询性能不会受到严重的负面影响。本次实验要求补全解决读写和写读冲突的多版本并发控制，以及两阶段锁协议解决基本写写冲突，而涉及死锁问题的复杂写写冲突以及垃圾回收机制作为可选的高级功能。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。

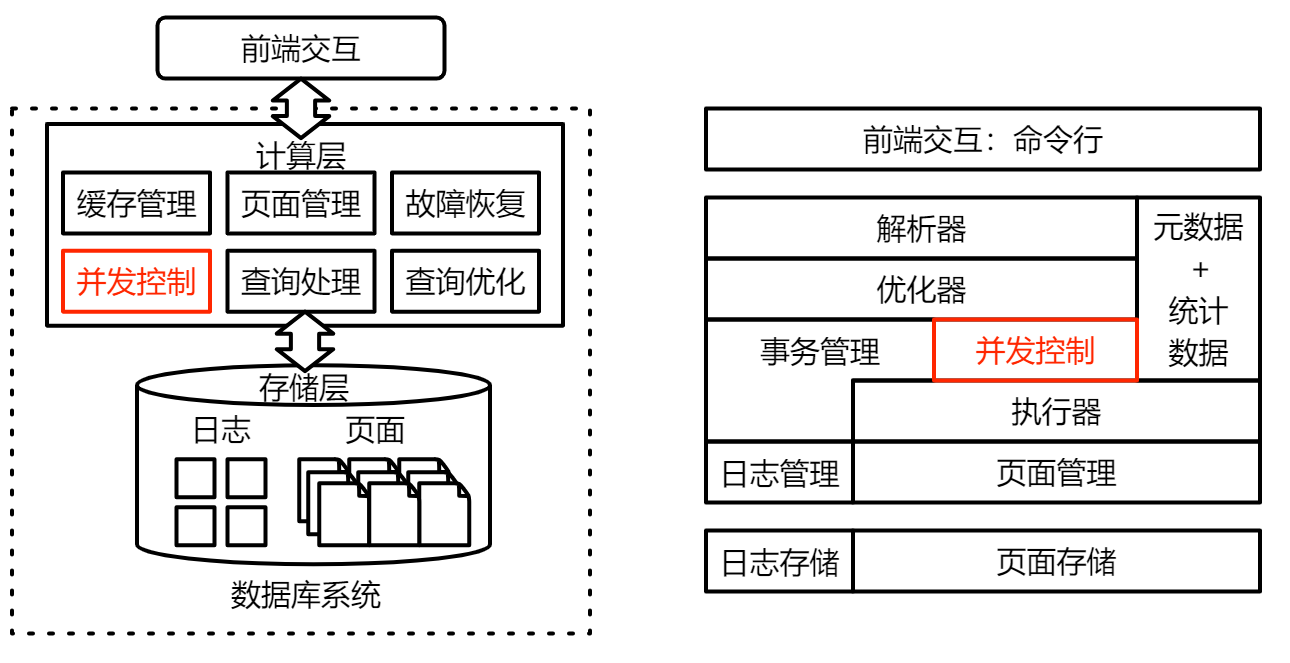


图1 数据库系统整体层次划分图

* + 1. 实验目标

**本次实验要求完成如下基础功能：**

* 多版本并发控制：修改数据表的增删改查操作，添加版本信息的记录以及记录对于事务的可见性判断。
* 不同隔离级别的实现：通过设置 isolation\_level 参数，控制事务的隔离级别，本次实验你需要实现读已提交、可重复读和可串行化事务隔离级别。
* 两阶段锁：补全锁管理器和算子的锁申请，实现强两阶段锁协议，解决并发查询处理中的写写冲突问题。

**（扩展）在基础功能之上，实验框架支持完成以下高级功能：**

* 写写冲突的死锁处理：实验框架和测例没有涉及并发查询中写写冲突可能导致的死锁问题，加入死锁预防或死锁检测机制来解决死锁问题。
* 乐观并发控制：实验框架采用悲观并发控制解决写写冲突，高级功能要求添加基于严格时间戳协议的乐观并发控制功能。
  1. 实现多版本并发控制功能概述
     1. 解决万圣节问题

还记得在实验1中提到的万圣节问题吗，本次实验的第一个任务便是解决这个问题，完成后你将通过05-halloween.test测例。

**步骤 1：添加记录头信息**

记录头信息位于record\_header.h文件，具体包括以下几个字段：

| **变量名** | **变量类型** | **长度** | **作用** |
| --- | --- | --- | --- |
| deleted\_ | bool | 1 | 标注记录是否删除（实验 3 之后弃用） |
| xmin\_ | xid | 4 | 插入该数据的事务 id |
| xmax\_ | xid | 4 | 删除该数据的事务 id |
| cid\_ | cid\_t | 4 | 插入该数据的事务内部 command id |

实验框架的 TransactionManager 类会在每个事务开启时为其分配一个事务 id，记为 xid。在每个事务内部，TransactionManager 会为每条 SQL 语句分配一个 command id，记为 cid，我们将通过 cid 来解决万圣节问题。

你需要在修改实验 1 插入、删除记录和实验 2 中撤销删除记录的代码，涉及 TablePage 的 InsertRecord, DeleteRecord 和 UndoDeleteRecord 函数，在插入、删除以及撤销删除记录时正确设置记录头信息，在插入记录时设置 xmin 和 cid，删除和撤销删除记录时设置 xmax。

**步骤 2：修改可见性判断条件**

下一步你需要修改 table\_scan.cpp 中 GetNextRecord 函数关于可见性判断的部分，为了解决万圣节问题，我们不仅需要判断记录是否被删除，还要判断读入的记录是不是由当前正在执行的 SQL 语句插入的，判断依据便是记录的 xid 和 cid，当前正在执行的 SQL 语句的 xid,和 cid。正确修改判断条件后，你将通过10-halloween.test 测例。同时，你也要保证代码的修改不影响前两次实验的正确性。

* + 1. 通过多版本并发控制实现可重复读隔离

本任务中，你将使用多版本并发控制，来实现可重复读隔离级别，正确实现后将通过

20-mvcc\_insert.test,21-mvcc\_delete.test,22-mvcc\_update.test,23-write\_skew.test和30-repeatable\_read.test测例。

**步骤 1：设置活跃事务集合**

在可重复读隔离级别中，你需要获取事务开始时的活跃事务集合，用于可见性判断。TransactionManager 已为你保存了这些信息，你只需要传入 xid 即可获取事务 xid 开始时的活跃事务集合，将该信息补充到seqscan\_executor.cpp的 Next 函数中。

**步骤 2：修改可见性判断条件**

修改 table\_scan.cpp 中的 GetNextRecord 函数，根据隔离级别，活跃事务 id，以及当前事务的 xid，cid，判断记录可见性。由于可见性判断的部分较为复杂，可能涉及多个分支，你可以将此部分单独设置一个函数。正确设置后将通过20-mvcc\_insert.test,21-mvcc\_delete.test,22-mvcc\_update.test,23-write\_skew.test和30-repeatable\_read.test测例。

你可能会发现，尽管通过了实验3的20-30测例，但实验 2 的30-aries.test再次出错。这是因为实验框架的默认隔离级别为读已提交，而本任务中我们对可重复读隔离级别进行了设置。你需要在判断可见性时考虑到当前的隔离级别，根据不同级别使用不同的可见性判断条件。在本任务中，你可以先忽略这个问题，后续任务我们将指导你完成其他隔离级别的可见性判断条件。

* + 1. 实现读已提交隔离

与任务2类似，你需要做的也是设置活跃事务集合和修改可见性判断条件，你需要思考读已提交和可重复读的判断条件有哪些不同，设置正确后将通过40-read\_committed.test测例。

* + 1. 实现强两阶段锁

任务2中，我们实现了多版本并发控制，无阻塞地处理了读写冲突，本任务中，我们将使用强两阶段锁(SS2PL)解决写写冲突问题。

**步骤 1：实现锁管理器 LockManager**

你需要实现 LockManager 类的 LockTable, LockRow 和 ReleaseLocks 函数，这几个函数分别实现了加表锁、加行锁和释放锁的功能，你可能需要添加适当的数据结构来保存锁信息。

此外，LockManager 类还提供了两个私有函数 Compatible 和 Upgrade 接口，Compatible 函数用于判断不同事务对同一个对象（行或表）加的锁是否兼容，Upgrade 用于同一个事务对相同对象重复加锁时实现锁的升级。

**步骤 2：为修改操作加锁**

完成 LockManager 后，你需要在 insert\_executor.cpp, delete\_executor.cpp 和 update\_executor.cpp 中为修改操作加锁，如果加锁失败，即其他事务已经获取相应对象的锁，且锁类型不兼容，则抛出异常。

正确实现以上步骤后，你将通过50-lock.test测例。

* + 1. 通过多版本两阶段锁实现可串行化隔离

最后一个任务，你将综合以上完成的所有任务，通过多版本两阶段锁协议，实现可串行化隔离。

**步骤 1：实现 select for update 和 select for share 的加锁**

你需要在lock\_rows\_executor.cpp中为 select for update 和 select for share 添加支持，根据 SQL 语句类型加读锁或写锁。

**步骤 2：根据 select 类型获取活跃事务集合**

修改seqscan\_executor.cpp，在可串行化隔离级别下，根据 SQL 语句的类型，判断应采用当前读还是快照读，进而设置正确的活跃事务集合。

**步骤 3：设置可串行化隔离级别的可见性判断条件**

修改table\_scan.cpp，为可串行化级别设置可见性判断条件。正确完成以上步骤后，你将通过60-mv2pl.test和70-write\_skew\_serializable.test，完成本次实验的所有基础功能。

* + 1. 扩展高级功能

**（1）任务 1：死锁预防与死锁检测**

【实验描述】

悲观并发控制机制在框架中加入两阶段锁机制来解决写写冲突问题，但是缺少对于死锁情况的处理，高级功能中需要在框架基础上添加死锁检测机制解决死锁问题。

【实现思路】

步骤 0：阅读教材相关内容，理解死锁发生的条件并设计产生死锁的测例文件。

步骤 1：涉及死锁检测机制，可以考虑使用锁申请的等待时间作为检测标准。

步骤 2：当检测到死锁后撤销当前事务并重做，注意重做的启动时间问题，避免相同事务重复陷入死锁状态。

**（2）任务 2：基于版本链的垃圾回收**

【实验描述】

与实验 1 的高级功能类似，本任务需要对无用元组进行回收。然而与实验 1 通过全表扫描进行垃圾回收的方法不同，本任务需要基于 MVCC 版本链实现记录扫描过程中的垃圾回收操作。

【实现思路】

步骤 1：为记录头添加版本链信息，从旧版本指向新版本。

步骤 2：修改记录扫描的逻辑，在扫描过程中回收无用的旧版本数据。

1.2.7 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：

|  |
| --- |
| * [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)：修改实验 1 中记录的增删改接口，添加版本信息。   + [table\_scan](https://github.com/thu-db/huadb/tree/main/src/table/table_scan.h)：添加判断记录对事务可见性的接口，并修改查询函数。 * [transaction](https://github.com/thu-db/huadb/tree/main/src/transaction/)：事务相关类。   + [lock\_manager](https://github.com/thu-db/huadb/tree/main/src/transaction/lock_manager.h)：锁管理器，负责管理事务锁，需要补充表级别、行级别加解锁函数以及相关辅助函数。 * [excutors](https://github.com/thu-db/huadb/tree/main/src/executors/)：执行器相关类。   + [seqscan\_executor](https://github.com/thu-db/huadb/tree/main/src/executors/seqscan_executor.h)：需要补充获取事务列表的功能。   + [insert\_executor](https://github.com/thu-db/huadb/tree/main/src/executors/insert_executor.h)：添加插入的写事务锁。   + [delete\_executor](https://github.com/thu-db/huadb/tree/main/src/executors/delete_executor.h)：添加删除的写事务锁。   + [update\_executor](https://github.com/thu-db/huadb/tree/main/src/executors/update_executor.h)：添加更新的写事务锁。   + [lock\_rows\_executor](https://github.com/thu-db/huadb/tree/main/src/executors/lock_rows_executor.h)：添加行级别加锁保护功能。 |

基础功能需要补充约 100 行代码，本次实验依赖于实验1及实验2任务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：

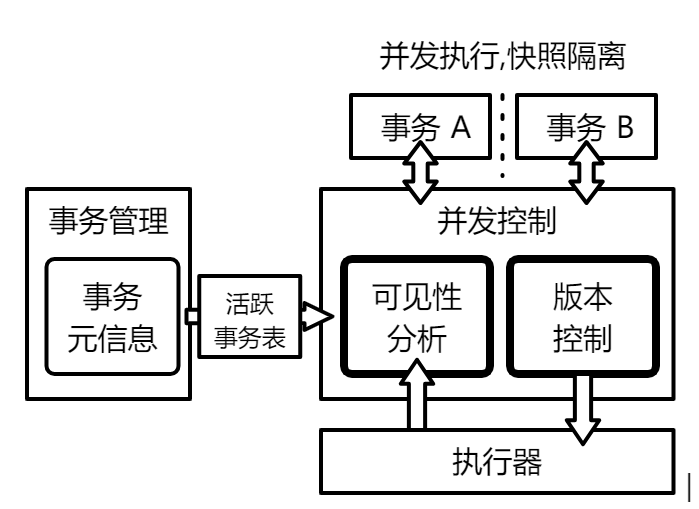


图2 相关功能模块的抽象示意图

* 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3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** | | | |

# 实验八：多版本并发控制

参考https://thu-db.github.io/huadb-doc/的实验三部分

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

# 一、实验要求

1、完成HuaDB数据库内核[实验3：多版本并发控制](https://thu-db.github.io/huadb-doc/lab3/1-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。