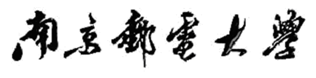
****

**数据库管理系统原型说明书**

**（2024 / 2025 学年 第 二 学期）**

**课程名称：数据库管理系统实现**

**学院： 计算机、软件、信息安全**

**专业: 软件工程**

**指导教师: 韩丽萍**

**班级: B220418**

**组长联系方式（邮箱，手机）:**

**张新建（278254081@qq.com.com，15349365773）**

**组员学号、姓名、评分模式:**

|  |  |  |  |
| --- | --- | --- | --- |
| **B22041822** | **B22041823** | **B22041824** | **B22041825** |
| **吴树成** | **吴浩** | **张新建** | **杨睿哲** |
| **W-1** | **W-1** | **W+3** | **W-1** |

**完成时间：2025 年 6 ⽉ 6 日**

**成绩评定：**

|  |  |  |  |
| --- | --- | --- | --- |
| **分数** |  | **评定人** | **韩丽萍** |

目录

[第一部分 程序架构 3](#_Toc200381654)

[1.1整体架构图 3](#_Toc200381655)

[第二部分 模块说明 4](#_Toc200381656)

[2.1 各个模块说明 4](#_Toc200381657)

[2.1.1.Main Module ( main\_db.py ) 4](#_Toc200381658)

[2.1.2Query Processor( query\_plan\_db.py ) 4](#_Toc200381659)

[2.2新增模块描述 4](#_Toc200381660)

[2.2.1 index\_manager.py 4](#_Toc200381661)

[2.2.2 log\_db.py 5](#_Toc200381662)

[2.3修改模块描述 6](#_Toc200381663)

[2.3.1. main\_db.py 6](#_Toc200381664)

[2.3.2 schema\_db.py 7](#_Toc200381665)

[2.3.3 query\_plan\_db.py 7](#_Toc200381666)

[2.3.4parser\_db.py 8](#_Toc200381667)

[2.3.5 head\_db.py 9](#_Toc200381668)

[2.3.6 storage\_db.py 9](#_Toc200381669)

[2.3.7 common\_db.py 9](#_Toc200381670)

[2.3.8lex\_db.py 10](#_Toc200381671)

[2.3.9 index\_db.py 10](#_Toc200381672)

[第三部分 实测数据与截图 10](#_Toc200381673)

[3.1 运行主程序 10](#_Toc200381674)

[3.2 创建表 11](#_Toc200381675)

[3.3 插入数据 11](#_Toc200381676)

[3.4 更新一行数据 12](#_Toc200381677)

[3.5 查看表数据 13](#_Toc200381678)

[3.6 删除一行数据 13](#_Toc200381679)

[3.7 删除表 14](#_Toc200381680)

[3.8删除表 14](#_Toc200381681)

[3.9 查看日志 14](#_Toc200381682)

[3.10 SQL语句 15](#_Toc200381683)

[3.11索引 17](#_Toc200381684)

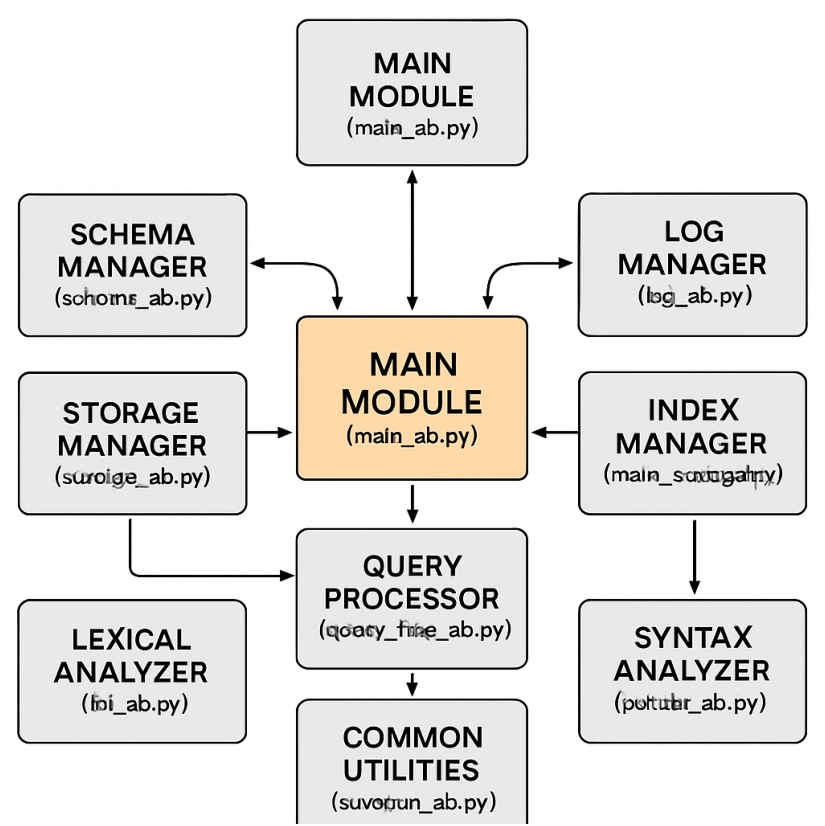
第一部分 程序架构

1.1整体架构图

引言

本说明书旨在详细描述数据库管理系统原型的设计、实现及功能。该原型系统在mini\_base\_blank\_python3.8 的基础上进行了功能扩展和优化，主要新增了日志管理和索引管理模块，并对现有模块进行了修改和完善，以提供更全面的数据库操作能力。

程序架构

本数据库管理系统采用模块化设计，各个模块职责清晰，相互协作。 main\_db.py 作为主入口模块，负责用户交互和调度其他模块完成数据库操作。其主要模块及其相互调用关系如下图所示：

第二部分 模块说明

2.1 各个模块说明

2.1.1.Main Module ( main\_db.py )

系统的核心入口，提供用户界面，处理用户输入，并协调其他模块执行相应的数据库操作。

Schema Manager ( schema\_db.py ): 负责管理数据库的元数据，包括表的结构、字段信息等，并将这些信息持久化到 all.sch 文件中。

Storage Manager ( storage\_db.py ): 负责数据的物理存储和检索，每个表的数据存储在一个独立的 .dat 文件中，并提供记录的插入、删除、更新和查询功能。

2.1.2Query Processor( query\_plan\_db.py )

负责将SQL语句解析后的语法树转换为查询计划，并执行查询操作。

Lexical Analyzer( lex\_db.py ): 对输入的SQL语句进行词法分析，将其分解为一系列

的词法单元（tokens）。

Syntax Analyzer( parser\_db.py ): 对词法分析器生成的tokens进行语法分析，构建

SQL语句的语法树。

Common Utilities ( common\_db.py ): 提供全局变量、常量和一些通用的辅助函数和数据结构（如 Node 类），供其他模块共享使用。

Log Manager ( log\_db.py ): （新增模块） 负责事务日志的管理，记录数据修改前后

的镜像以及事务状态，用于故障恢复。

Index Manager ( index\_manager.py ): （新增模块） 负责索引的创建、管理和性能比较，支持B-tree索引和Hash索引。

2.2新增模块描述

2.2.1 index\_manager.py

功能： 该模块提供简化的索引管理接口，包括索引性能比较、删除所有索引、列出所有索引和返回主菜单等功能。它支持B-tree索引和Hash索引的创建和管理。

关键函数和类：

handle\_index\_management(schemaObj)

◦ 输入： schemaObj (Schema对象，包含表元数据)

◦ 输出： 无 (交互式菜单界面)

◦ 功能： 索引管理的主菜单处理函数，根据用户选择调用不同的索引操作函数。

comprehensive\_index\_comparison(schemaObj)

◦ 输入： schemaObj (Schema对象，包含表元数据)

◦ 输出： 无 (显示综合性能比较结果)

◦ 功能： 进行全面的索引性能比较。它会为指定表的指定字段创建B-tree索引和Hash索引，并与顺序扫描进行性能对比，显示创建时间、总搜索时间、平均搜索时间、加速比、存储空间占用以及结果一致性。

delete\_all\_indexes(schemaObj)

◦ 输入：schemaObj (Schema对象，包含表元数据)

◦ 输出： 无 (删除所有索引文件)

◦ 功能： 查找并删除所有存在的B-tree索引文件（.ind ）和Hash索引文件（ .hash ）。

list\_all\_indexes(schemaObj)

◦ 输入： schemaObj (Schema对象，包含表元数据)

◦ 输出： 无 (显示所有索引文件的列表)

◦ 功能： 列出当前系统中所有存在的索引文件，包括表名、索引类型、文件名和文件大小。

2.2.2 log\_db.py

功能： 该模块是日志管理模块，负责处理事务日志，包括记录数据修改前后的镜像（Before Image和After Image）以及事务的活动状态和提交状态。这些日志用于支持数据库的故障恢复。

关键函数和类：

LogManager 类

◦ 作用： 这是一个静态类，包含了所有日志操作的方法，通过类方法实现日志的写入、读取和管理。

◦ 变量：

▪lock : 线程锁，用于保证日志文件写入的线程安全。▪ LOG\_DIR : 日志文件存放的目录。

▪ BEFORE\_IMAGE\_FILE , AFTER\_IMAGE\_FILE , ACTIVE\_TX\_FILE ,

COMMIT\_TX\_FILE : 各类日志文件的路径。

\_write\_log(file\_path, msg)

◦ 输入： file\_path (目标日志文件路径), msg (要写入的日志内容)

◦ 输出： 无

◦ 功能： 将消息写入指定的日志文件，并确保线程安全。

\_format\_data(data)

◦ 输入： data (要格式化的数据，可以是字节串、字符串、列表等)

◦ 输出： str (格式化后的字符串，移除了 b'' 前缀)

◦ 功能： 格式化数据，使其适合写入日志文件，特别是处理字节串数据。

log\_before\_image(tx\_id, table, record, operation="UNKNOWN")

◦ 输入：tx\_id (事务ID), table (表名), record (操作前的数据记录),

operation (操作名称，如CREATE\_TABLE, INSERT等)

◦ 输出： 无

◦ 功能： 记录事务操作前的“前镜像”日志，包含事务ID、操作、表名和操作前的数据。

log\_after\_image(tx\_id, table, record, operation="UNKNOWN")

◦ 输入： tx\_id (事务ID), table (表名), record (操作后的数据记录),

operation (操作名称)

◦ 输出： 无

◦ 功能： 记录事务操作后的“后镜像”日志，包含事务ID、操作、表名和操作后的数据。

add\_active\_tx(tx\_id, operation="UNKNOWN")

◦ 输入： tx\_id (事务ID), operation (操作名称)

◦ 输出： 无

◦ 功能： 记录活动事务日志，表示一个事务开始执行。

add\_commit\_tx(tx\_id, operation="UNKNOWN")

◦ 输入： tx\_id (事务ID), operation (操作名称)

◦ 输出： 无

◦ 功能： 记录提交事务日志，表示一个事务成功完成。

read\_log\_file(file\_path)

◦ 输入： file\_path (要读取的日志文件路径) ◦ 输出： list (日志条目列表)

◦ 功能： 读取指定日志文件的内容并返回。

clear\_all\_logs()

◦ 输入： 无

◦ 输出： 无

◦ 功能： 清空所有日志文件的内容。

get\_log\_statistics()

◦ 输入： 无

◦ 输出： dict (包含所有日志文件统计信息的字典)

◦ 功能： 获取并返回各类日志文件的统计信息，如日志条目数量。

2.3修改模块描述

2.3.1. main\_db.py

修改内容：

导入新增模块： 增加了对 log\_db 和 index\_manager 模块的导入。

菜单选项扩展：在主菜单PROMPT\_STR中新增了选项 8: View log files 和 9: Index Management ，以支持日志查看和索引管理功能。

日志查看功能：

在 main 函数中，当用户选择 8 时，调用 log\_db.LogManager 的相关方法读取并打印Before Image、After Image、Active Transaction和Commit Transaction日志。索引管理功能在 main 函数中，当用户选择9 时，调用

index\_manager.handle\_index\_management(schema\_obj) 进入索引管理子菜单。

2.3.2 schema\_db.py

修改内容：

导入 log\_db 模块： 增加了对 log\_db 模块的导入，用于在表结构操作中记录日志。appendTable 函数修改：

功能： 在创建表结构时增加了事务日志记录功能。在表创建前记录BEFORE\_IMAGE ，创建成功后记录 AFTER\_IMAGE 和 COMMIT 日志。同时，优化了字段信息的格式化处理。

delete\_table\_schema 函数修改：

功能： 优化了删除表结构时的逻辑，确保在删除表后， all.sch 文件中的元数据能够正确更新，并处理了表名编码问题。

2.3.3 query\_plan\_db.py

SQL语句执行扩展： 增加了对 CREATE TABLE , INSERT INTO , DELETE FROM , UPDATE SET , DROP TABLE 等SQL语句的执行逻辑。

execute\_create\_table(syn\_tree, schema\_obj)函数新增：

输入： syn\_tree (CREATE TABLE语句的语法树节点), schema\_obj (Schema对象)

输出： 无 (在schema和存储中创建表)

功能： 根据语法树信息执行CREATE TABLE语句，创建新的表结构和数据文件。

execute\_insert\_into(syn\_tree) 函数新增：

输入： syn\_tree (INSERT INTO语句的语法树节点)

输出： 无 (向表中插入记录)

功能： 根据语法树信息执行INSERT INTO语句，向指定表中插入数据记录。

execute\_delete\_from(syn\_tree) 函数新增：

输入： syn\_tree (DELETE FROM语句的语法树节点)

输出： 无 (从表中删除记录)

功能： 根据语法树信息执行DELETE FROM语句，从指定表中删除符合条件的记录。

Execute\_update\_set(syn\_tree) 函数新增：

输入：syn\_tree (UPDATE SET语句的语法树节点)

输出： 无 (更新表中记录)

功能： 根据语法树信息执行UPDATE SET语句，更新指定表中符合条件的记录。

execute\_drop\_table(syn\_tree, schema\_obj) 函数新增：

输入： syn\_tree (DROP TABLE语句的语法树节点), schema\_obj (Schema对象)

输出： 无 (删除表)

功能： 根据语法树信息执行DROP TABLE语句，删除指定的表结构和数据文件。

execute\_sql\_statement(schema\_obj) 函数修改：

功能： 增加了对新增SQL语句类型（CREATE TABLE, INSERT INTO, DELETE FROM, UPDATE SET, DROP TABLE）的判断和分发，根据语法树的类型调用相应的执行函数。

2.3.4parser\_db.py

SQL语法规则扩展： 增加了对 CREATE TABLE , INSERT INTO , DELETE FROM ,

UPDATE SET , DROP TABLE 等SQL语句的语法解析规则。

p\_create\_query 函数新增：

功能： 解析 CREATE TABLE 语句，构建相应的语法树节点，包含表名和字段定义。

p\_field\_defs 和 p\_field\_def 函数新增：

功能： 解析CREATE TABLE 语句中的字段定义，包括字段名、数据类型和长度。

p\_data\_type 函数新增：

功能： 解析 CHAR 和 INTEGER 数据类型及其长度。p\_insert\_query 函数新增：

功能： 解析 INSERT INTO 语句，构建相应的语法树节点，包含表名和值列表。

p\_value\_list 函数新增：

功能： 解析 INSERT INTO 语句中的值列表。p\_delete\_query 函数新增：

功能： 解析 DELETE FROM 语句，构建相应的语法树节点，包含表名和可选的WHERE条件。

p\_update\_query 函数新增：

功能： 解析 UPDATE SET 语句，构建相应的语法树节点，包含表名、赋值列表和可选的WHERE条件。

p\_assignment\_list 和 p\_assignment 函数新增：◦ 功能： 解析 UPDATE SET 语句中的赋值列表。

p\_drop\_query 函数新增：

功能： 解析 DROP TABLE 语句，构建相应的语法树节点，包含表名。p\_sfw 函数修改：

功能： 调整了SFW（SELECT-FROM-WHERE）语句的语法树构建方式，使其更符合逻辑。

p\_opt\_where 函数修改：

功能： 优化了可选WHERE子句的解析，使其能够正确处理WHERE条件。

p\_simple\_cond 函数修改：

功能： 优化了简单条件（如column = value）的解析。p\_sel\_list\_recursive ,

p\_sel\_list\_single , p\_sel\_list\_star 函数修改：

功能： 优化了SELECT列表的解析，确保能够正确处理单个字段、多个字段和 \* 。

p\_from\_list\_recursive , p\_from\_list\_single 函数修改：

功能： 优化了FROM列表的解析。

2.3.5 head\_db.py

修改内容：

Header 类构造函数修改：

功能：调整了构造函数中的打印信息，使其更清晰。

2.3.6 storage\_db.py

导入新增模块： 增加了对 log\_db 和 uuid 模块的导入，用于事务日志记录和生成唯一

事务ID。

\_\_init\_\_ 函数修改：

功能： 优化了文件初始化和元数据读取逻辑，特别是增加了对field\_list\_from\_create\_table 参数的处理，使得在执行 CREATE TABLE语句时可以直接传入字段列表。

Insert\_record 函数修改：

功能： 增加了事务日志记录功能。在插入记录前记录 BEFORE\_IMAGE ，插入成功后记录 AFTER\_IMAGE 和 COMMIT 日志。同时，优化了记录的验证和填充逻辑，确保数据类型和长度的正确性。

Delete\_record\_by\_field 函数修改：

功能： 增加了事务日志记录功能。在删除记录前记录 BEFORE\_IMAGE ，删除成功后记录 AFTER\_IMAGE 和 COMMIT 日志。同时，优化了删除记录的逻辑，确保能够正确找到并删除符合条件的记录。

Update\_record\_by\_field 函数修改：

功能： 增加了事务日志记录功能。在更新记录前记录 BEFORE\_IMAGE ，更新成功后记录 AFTER\_IMAGE 和 COMMIT 日志。同时，优化了更新记录的逻辑，确保能够正确找到并更新符合条件的记录。

show\_table\_data 函数修改：

功能： 优化了显示表数据的格式，使其更易读。delete\_table\_data 函数修改：

功能： 优化了删除表数据文件的逻辑。

2.3.7 common\_db.py

全局变量新增：增加了 global\_logical\_tree 全局变量，用于存储逻辑查询计划树。

Node 类修改：

功能： 增加了 varList 参数，允许在节点中存储额外的变量信息，这对于SQL语

句的语法树节点非常有用。

2.3.8lex\_db.py

Tokens扩展： 增加了对 CREATE , TABLE , INSERT , INTO , VALUES , DELETE , FROM , UPDATE , SET , DROP , CHAR , INTEGER 等SQL关键字的token定义。t\_TCNAME 函数修改：

功能：优化了表名/列名的识别逻辑，使其能够正确识别新增的SQL关键字。

2.3.9 index\_db.py

修改内容：

Index 类新增：

作用： 用于管理和操作索引，支持B-tree和Hash索引的创建、搜索和删除。◦ 变量：

table\_name : 索引所属的表名。

index\_type : 索引类型（B-tree或Hash）。

index\_file : 索引文件对象。

index\_data : 索引数据（字典形式，键为字段值，值为记录偏移量列表）。

BTREE\_INDEX , HASH\_INDEX : 索引类型常量。

关键函数：

\_\_init\_\_(self, table\_name, index\_type) : 构造函数，初始化索引对象，并尝试加载现有索引文件。

create\_index(self, field\_name, index\_type) : 根据指定字段和索引类型创建索引。它会遍历表数据，构建索引结构，并将索引数据写入文件。

search\_by\_index(self, field\_name, search\_value) : 根据索引查找符合条件的记录。它会利用索引数据快速定位记录，并从存储文件中读取完整记录。

delete\_index(self) : 删除当前索引文件。

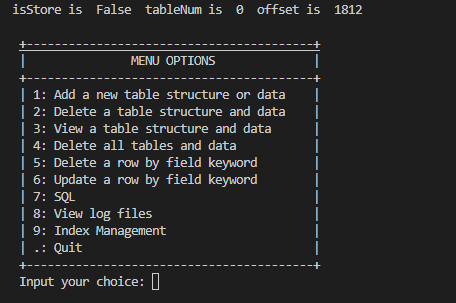
第三部分 实测数据与截图

为了验证系统的功能和性能，我们进行了以下实测，并提供了相应的截图。

3.1 运行主程序

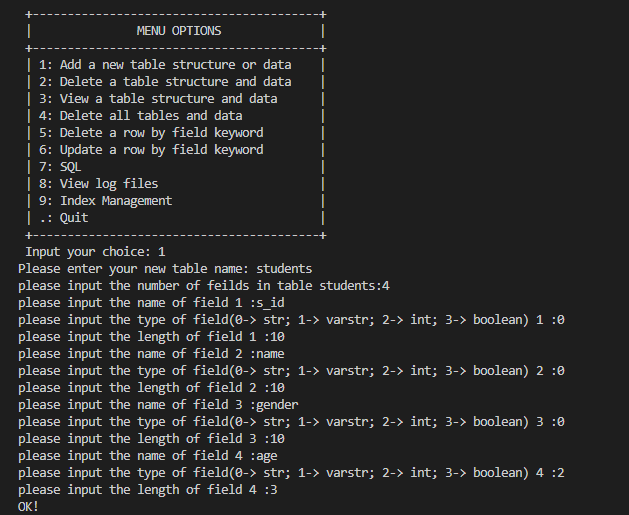
直接运行 main\_db.py 文件，系统将进入交互式菜单界面。主菜单提供了多种数据库操作选项，包括表的增删改查、SQL查询、日志查看和索引管理。

系统显示主菜单界面，等待用户输入选择。



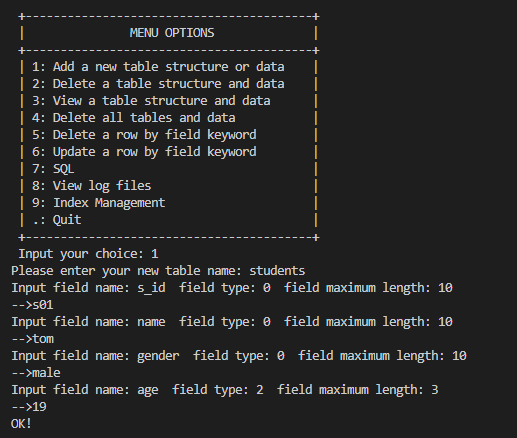
3.2 创建表

通过主菜单选项1创建新表。

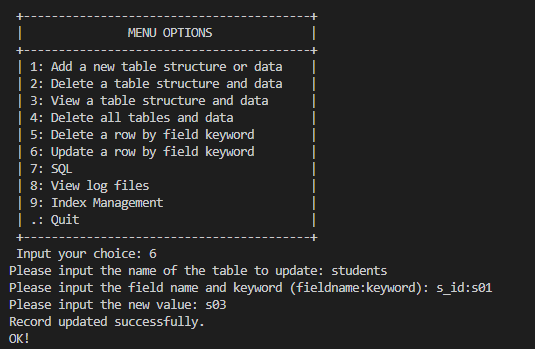


3.3 插入数据

通过主菜单选项1向表中插入数据。

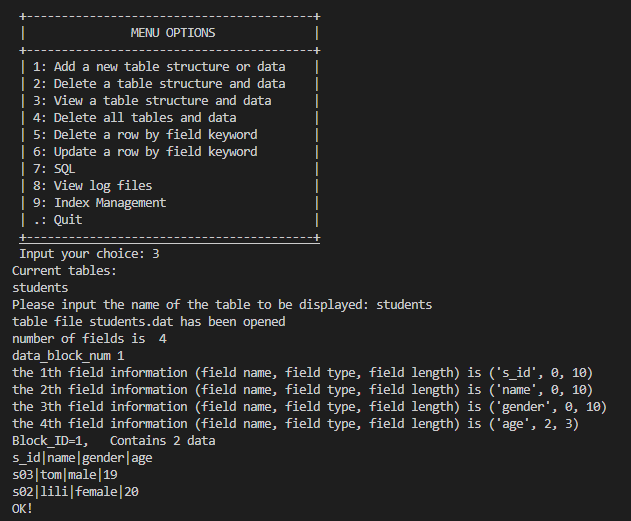


3.4 更新一行数据

通过主菜单选项 6更新表中的一行数据。

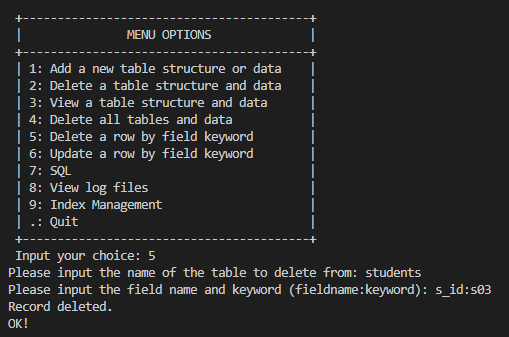
3.5 查看表数据

通过主菜单选项 3查看表中的数据。



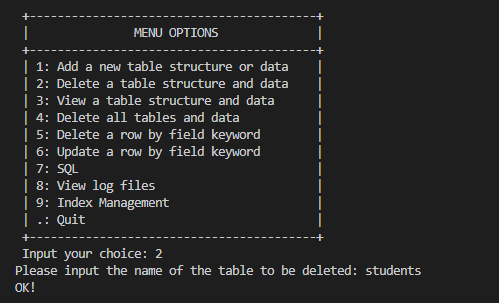
3.6 删除一行数据

通过主菜单选项 6删除表中的一行数据



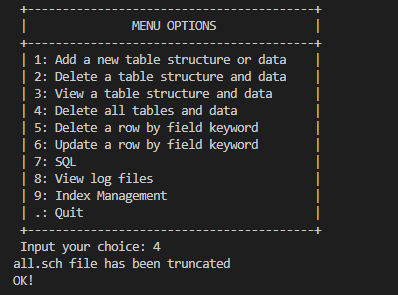
3.7 删除表

通过主菜单选项 2删除表



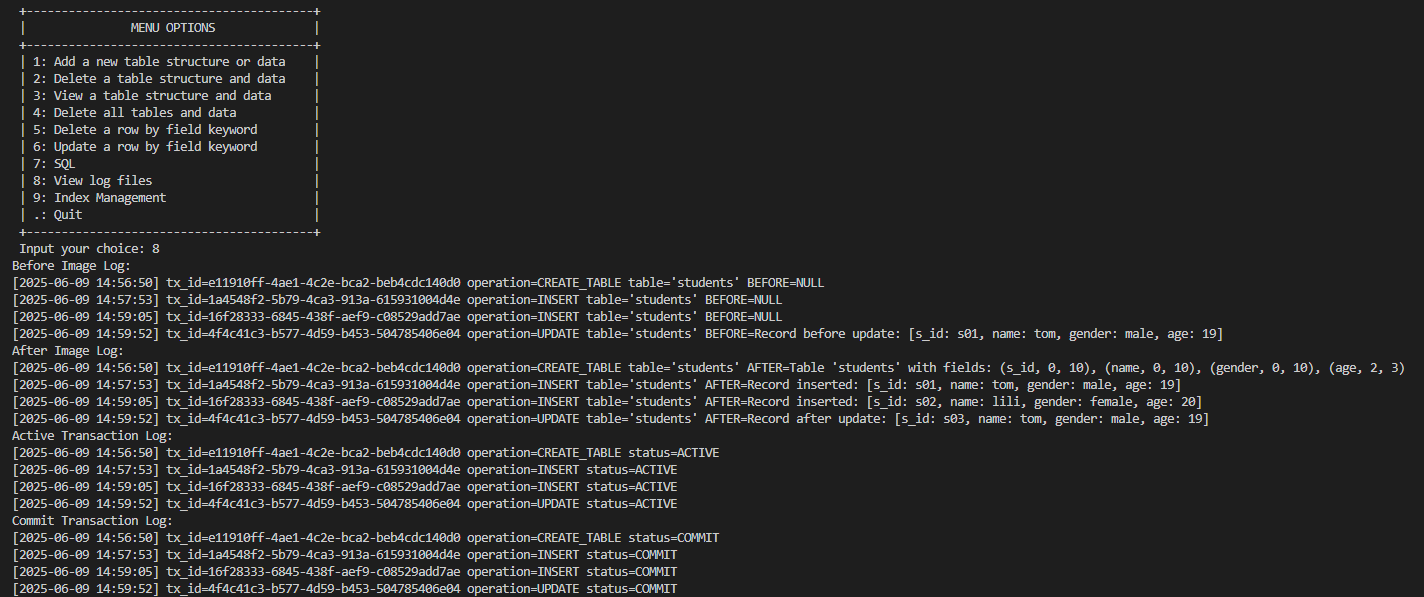
3.8删除表

通过主菜单选项 4删除所有表



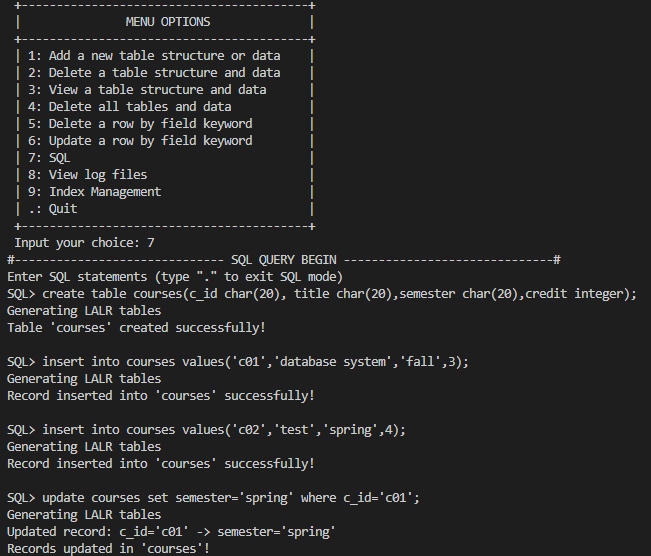
3.9 查看日志

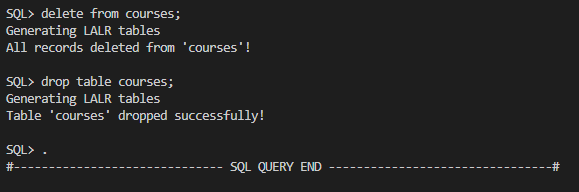
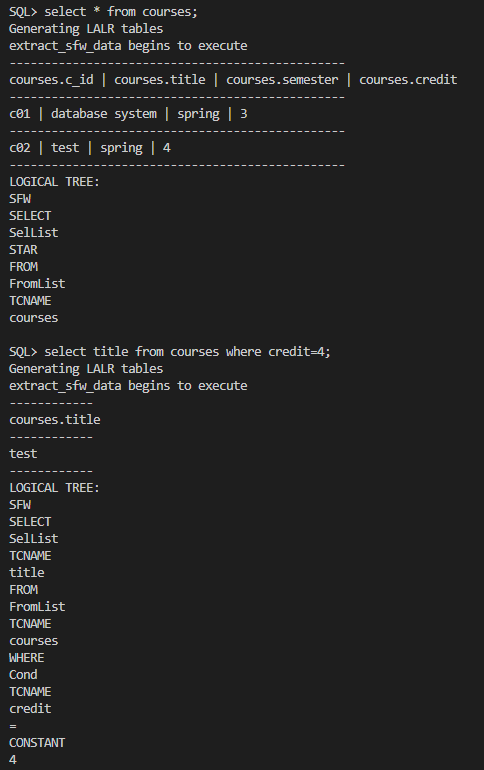
通过主菜单选项 8查看日志



3.10 SQL语句

通过主菜单选项 7支持SQL语句注入





3.11索引

通过主菜单选项 8开始索引选项

准备了20000条数据的表bigtest

