# 电子科技大学计算机科学与工程学院

实 验 报 告

课程名称： 数 据 库 新 技 术

实验名称： 分 布 式 数 据 库

学 号： 202422900232、202422900233

姓 名： 王冉恒、王涛

电 子 科 技 大 学

# 实 验 报 告

## 一、 课程名称

数据库新技术。

## 二、 实验项目名称

小型分布式图书馆系统的数据库系统

## 三、 实验原理

为了完成此实验，主要采用了RPC远程函数调用技术和SQL解析技术。

**（一）SQL 词法解析**

SQL 的词法解析可以分为如下步骤：

1. Lexer（词法分析器）：输入SQL查询字符串，输出一系列的标记(token)流。每个标记包括类型、值以及对应的行号。例如，'select \* from my\_table' 将被转化为一系列如['SELECT', 'select', 1]等标记，便于后续处理。
2. Parser（语法解析器）：目前仅支持SELECT查询，但能构建出Select对象，其中包括where、group、order和limit等属性。这些属性清晰地反映了SQL查询的结构。此外，调用Select对象的.toString()方法可以将其还原为原始的SQL语句。

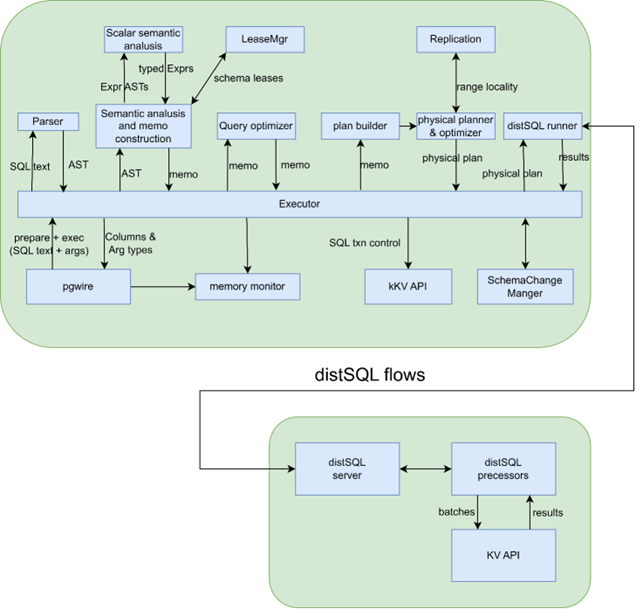


图 3.1: SQL 执行流程

如图所示为一条 SQL 语句的整体处理流程，总体来说，一条 SQL 语句需要经过下述步骤：parser—构建逻辑计划—构建优化计划—构建物理计划—构建执行计划。

parser 过程最主要的目的就是解析输入的 SQL 语句，通过词法解析器解析为 token，通过语法解析器生成抽象语法树，而后即可传入 SQL 执行引擎进行识别和处理。

接着进入下一步，首先要对输入的数据进行有效性验证，解析并转换 AST 树，构建逻辑计划。接下来就是对生成的计划进行优化以找到代价最小的执行方式，根据优化好的逻辑计划构建可执行的物理计划，最后下发到各个节点进行分布式执行。

**（二）RPC 远程函数调用**

RPC 是远程过程调用（Remote Procedure Call）的缩写形式，Birrell 和 Nelson 在 1984 发表于[ACM](https://so.csdn.net/so/search?q=ACM&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/lgxzzz/article/details/_blank) Transactions on Computer Systems 的论文《Implementing remote procedure calls》对 RPC 做了经典的诠释。RPC 是指计算机 A 上的进程，调用另外一台计算机 B 上的进程，其中 A 上的调用进程被挂起，而 B 上的被调用进程开始执行，当值返回给 A 时，A 进程继续执行。调用方可以通过使用参数将信息传送给被调用方，而后可以通过传回的结果得到信息。而这一过程，对于开发人员来说是透明的。

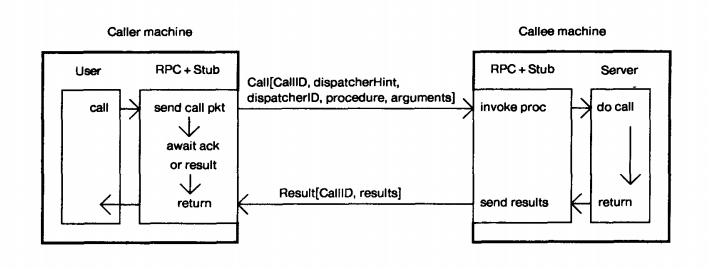


图 3.2: RPC 原理

如上图所示，远程过程调用采用客户机/服务器(C/S)模式。请求程序就是一个客户机，而服务提供程序就是一台服务器。和常规或本地过程调用一样，远程过程调用是同步操作，在远程过程结果返回之前，需要暂时中止请求程序。使用相同地址空间的低权进程或低权线程允许同时运行多个远程过程调用。

## 四、 实验目的

实现一款简易的分布式数据库系统，能够支持基本的数据库操作，提升自己对数据库系统的理解和工程能力。

## 五、 实验内容

设计和开发小型分布式图书馆系统的数据库系统，要求如下:

1. 数据必须存放在至少 3 个站点上，数据的划分对用户完全透明。
2. 系统必须支持基本的 SQL 查询语句 (可以不包含嵌套)，并能将查询结果反映到界面上。
3. 系统必须支持数据划分，可以选择支持水平划分，垂直划分，水平十垂直划分或者混合划分。
4. 用户查询要进行分解和优化，然后转交底层数据库管理系统执行。各分站点服务器界面上必须能显示出所接受和发送的命令，以及命令的处理过程。必须直接显示到界面上，不允许记录到 log 文件中。
5. 主站点要显示生成查询分解和优化过程。

## 六、 实验环境

## 操作系统：Ubuntu 20.04 开发环境：Python 3.10、Docker 24.0.7、docker-compose 1.29.2、Anytree 2.11.1、pandas 1.5.3、Grpc 4.2.7

## 开发工具：Visual Studio Code 1.93.1

## 七、 实验步骤

**（一）分布式系统的设计**

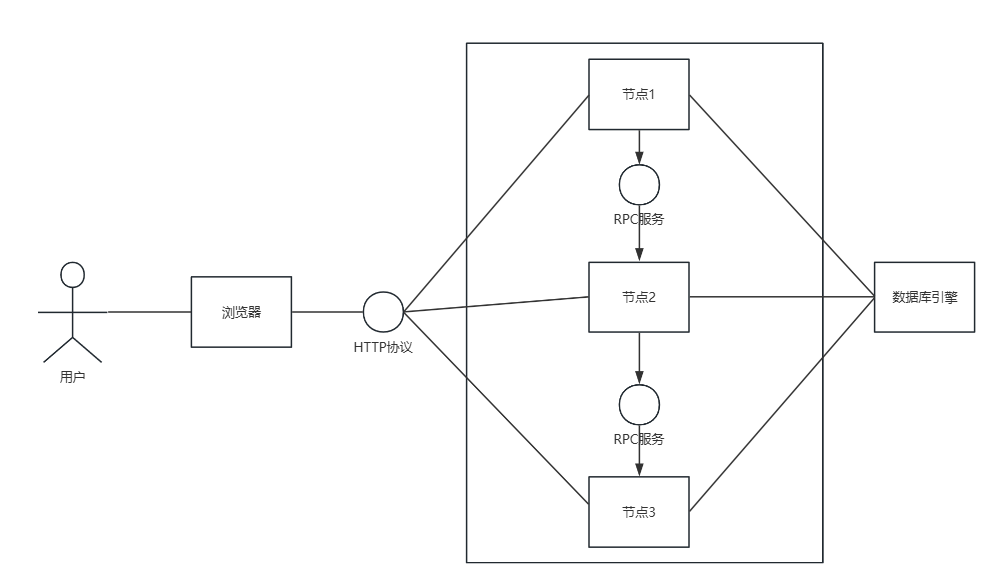


图 7.1: 分布式系统的架构

如图本系统总体架构为B/S架构，分为HTTP服务、RPC服务、数据库引擎三部分实现，HTTP服务负责向发送服务端发送http请求；RPC服务由gRPC实现，负责接收并处理RPC请求，用于实现各个节点直接数据同步与查询；数据库引擎负责解析sql语句后，按解析结果，对数据进行存储，查询，删除等操作。

**（二）分布式系统的实现**

分布式系统对外通过 HTTP 协议进行通信，从而接收用户命令以进行处理。

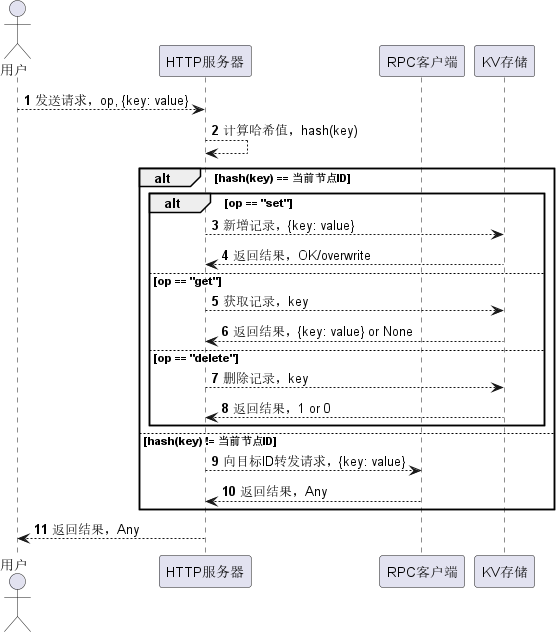


图 7.2: 接收 HTTP 请求流程

如图所示即为分布式系统中节点接收http请求的详细过程。用户通过浏览器客户端发送http请求后，http服务器通过一个hash函数计算出一个id，再通过这个id选择与系统中哪个节点的KV存储进行交互，通过操作字段op判断操作类型，选择相应的交互方法。

分布式系统内部不同的节点之间通过RPC协议进行通信，接收到客户端请求后，通过RPC调用对应节点的函数来执行请求，在本次实验中选择了第三方工具Grpc来实现。

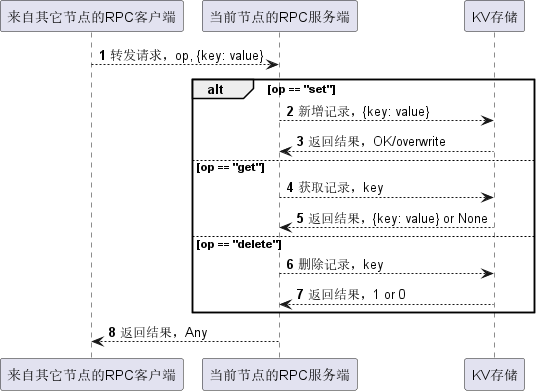


图 7.3: 接收 RPC 请求流程

如图所示为接收RPC请求的流程，当服务端接收到客户端的请求后，就会通过RPC协议与对应的节点进行交互，服务端解析sql语句后，判断出操作的类型，与本地KV进行交互，然后将处理结果返回http客户端。

**（三）数据库系统的设计**

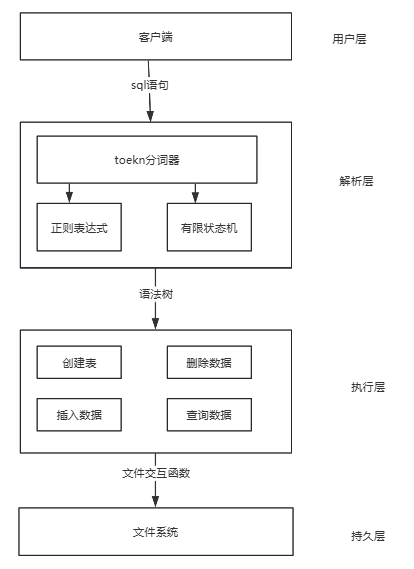


图 7.4: 数据库系统架构

数据库引擎以分层架构来实现，本系统共分为4层，分别为用户层、解析层、执行层、持久层：

用户层：用户层负责接收分布式系统中各个节点传来的http请求，并将http请求中的sql语句解析出来，传递给解析层。

解析层：负责解析sql命令，包括词法分析、语法分析、语义分析、查询重写、生成逻辑查询计划，解析后的sql命令将会生成一个语法树，然后传递给执行层。具体解析步骤如下：

### **1. 词法分析**

**任务**：将输入的SQL字符串转换为一系列**Token**，这些Token是更高层分析的基础单位。如SELECT name FROM users WHERE age > 30 分解为：

|  |  |
| --- | --- |
| SELECT | 关键字 |
| name | 标识符 |
| FROM | 关键字 |
| users | 标识符 |
| WHERE | 关键字 |
| age | 标识符 |
| > | 操作符 |
| 30 | 常量 |

**方法**：

1. **有限状态机**：

扫描SQL字符串，逐字符进行状态转移。

定义状态机的规则，例如：

遇到字母开头的字符串 → 识别为标识符或关键字。

遇到数字 → 识别为常量。

遇到符号 → 识别为操作符。

**2）正则表达式**：

使用正则表达式快速匹配Token，例如：

关键字匹配：SELECT|FROM|WHERE。

标识符匹配：[a-zA-Z\_][a-zA-Z0-9\_]\*。

数字匹配：\d+。

**2. 语法分析**

**任务**：根据SQL语法规则将词法单元序列转换为**语法树（Parse Tree）**

**语法树**：用来表示语句的语法结构，树的每个节点表示一个语法成分。例如对 SELECT name FROM users WHERE age > 30，语法树如下：

SELECT

/ | \COLUMN FROM WHERE

| | \

name users >

/ \

age 30

**方法**：

1. **递归下降解析器**：

解析器根据语法规则逐步解析语句。如：检测 SELECT 开头 → 匹配列名 → 匹配表名 → 检测 WHERE 子句。

1. **生成工具**：

使用解析工具（如ANTLR、Bison、Flex）自动生成解析器。语法规则通过工具转化 为代码，能快速支持复杂的SQL标准。

### **3. 语义分析**

### **任务：确保SQL语句符合数据库的语义规则，验证表、列、权限等是否有效。**

### **具体分析**

### **元数据检查：**

### **从数据库的元数据（Metadata）中获取表和列的定义。**

### **确认 users 表是否存在，name 和 age 是否是合法的列名。**

### **权限验证：检查当前用户是否有权限访问 users 表或相关列。**

### **数据类型检查：确保条件中的操作符和数据类型兼容。如：**

SELECT \* FROM users WHERE age > 'abc';

**报错：**Error: Incompatible types for comparison. Column 'age' is of type INT.

### **别名与子查询验证：确保别名在作用域内可见。确认子查询结果 是否符合预期。**

### **查询重写**

### **任务：对SQL语句进行逻辑优化，生成等价的、更高效的查询表达式。如：**

### **条件合并，合并重复的条件：**

SELECT \* FROM users WHERE age > 30 AND age > 25;

### **优化为：**

SELECT \* FROM users WHERE age > 30;

### **5. 生成逻辑查询计划**

#### **任务：**将解析后的语法树转化为逻辑操作计划（Logical Plan），描述查询执行的逻辑顺序。

#### **逻辑查询计划**

如对 SELECT name FROM users WHERE age > 30：

* 扫描 users 表。
* 过滤 age > 30。
* 投影（选择）列 name。

#### **优化逻辑计划：**通过谓词下推等技术将过滤操作尽可能提前执行，以减少中间数据量。

执行层：负责执行解析出来的各种 sql 命令，包括创建命令、插入命令、查询命令、删除命令，以及 where 条件的筛选策略。将语法树转换为与文件系统的若干交互函数。

持久层：与文件系统进行交互，将数据存储在磁盘上。。

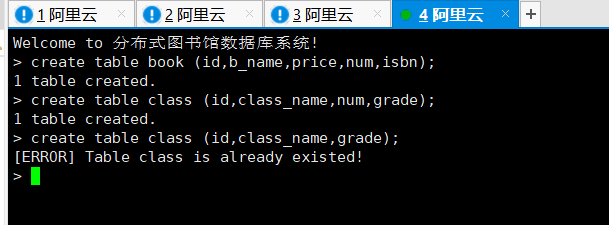
## 八、 实验结果及数据分析

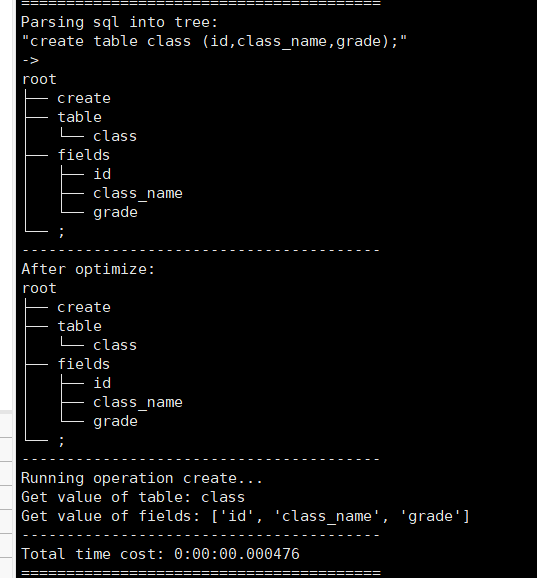
本章对实验结果进行展示，进行的测试包括创建表测试、添加数据测试、查询测试、删除测试。

**（一）创建表测试**

当系统成功运行后，在客户端直接输入创建表格的sql命令，即可创建table，，因为系统基于python开发，所以实现了动态字段类型，在创建表格时无需指定表中各个字段的类型，当要向表格插入数据时，会根据插入的数据自动指定。

如下图所示，在客户端中分别执行创建图书表、班级表的sql命令。

图 8.1: 创建表测试

如图所示，book 表和class 表都被成功创建，之后再次创建class 表时，因为class表已经被创建过了，就会创建失败，提示”Table class is already existed!” 错误。

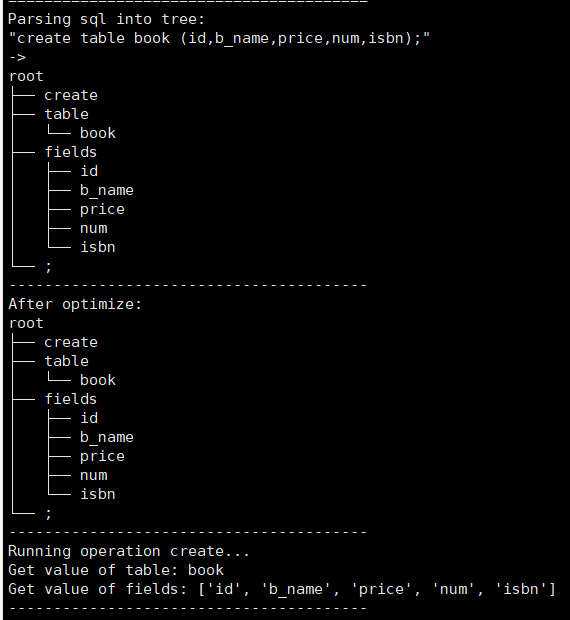


图 8.2: 8002 节点输出

如图展示插入表后，服务端的输出。客户端指定的接口为8000接口，然后8000接口服务器直接获得客户端请求，获得请求后，通过hash函数为这两个创建表请求分配节点，此处经过hash函数后，分配的节点为8002节点，于是在这个节点上经过sql语句解析后，进行文件操作，在8002的本地文件创建了book表和class表。

**（二）新增数据测试**

创建好表以后就可以通过sql命令，在对应的表中插入数据，如下图所示，在客户端中分别执行向图书表和班级表中插入数据的命令。

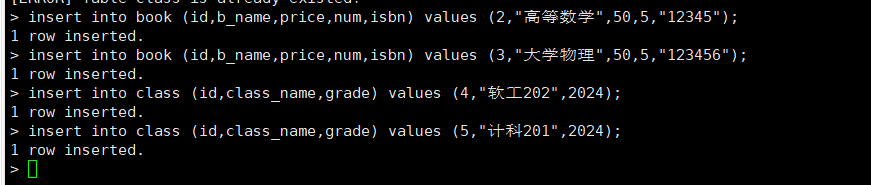
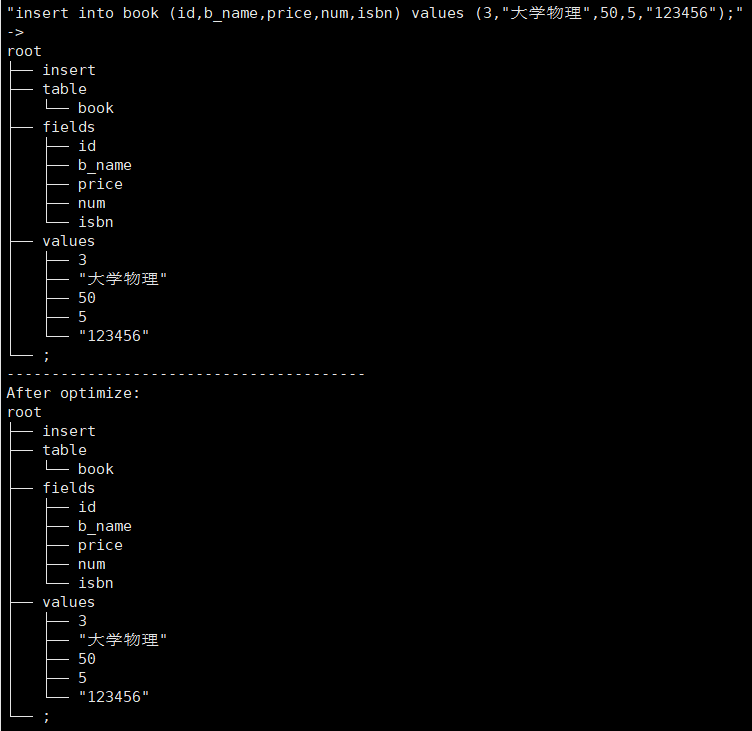


图 8.3: 新增数据

由于两个表都在8002节点，所以对应的增添操作都在8002节点进行，经过8002节点的sql语句解析后，进行文件操作，在8002节点的本地文件填入数据。数据插入后，8002节点屏幕以垂直划分输出信息如下图所示，表示以成功插入。



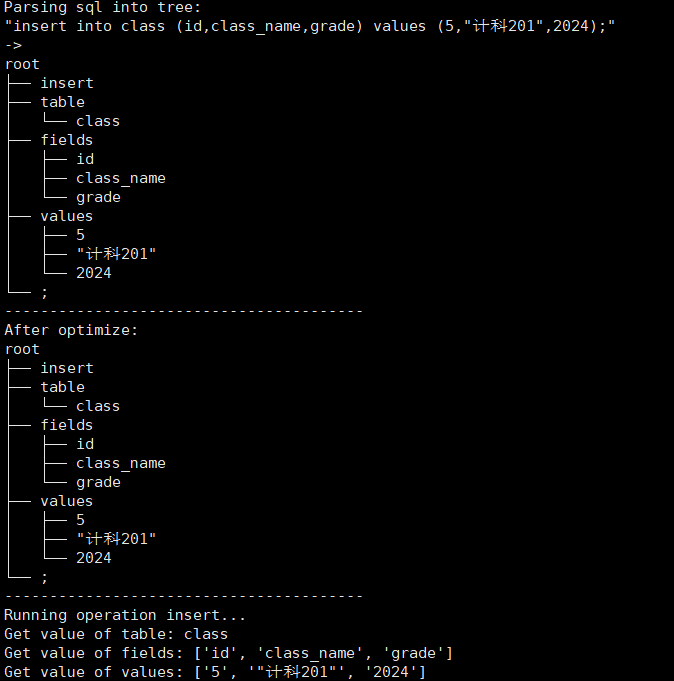


图 8.4: 新增数据后8082输出

1. **查询数据测试**

当数据表中存在数据后，就可以对其进行查询，本系统支持普通的无条件查询和带where的条件查询，暂不支持嵌套查询。如下图所示，在客户端分别执行查询图书表和班级表中数据的sql命令。

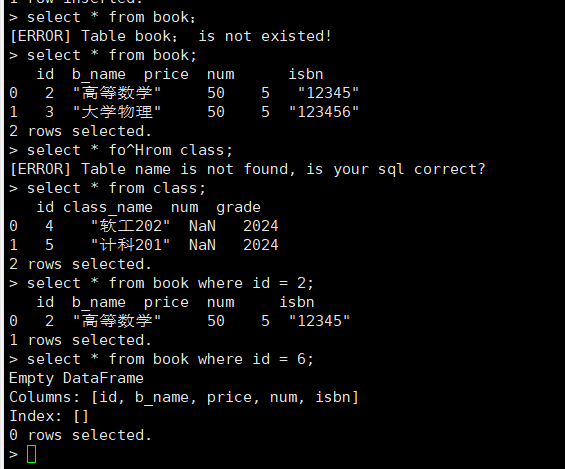
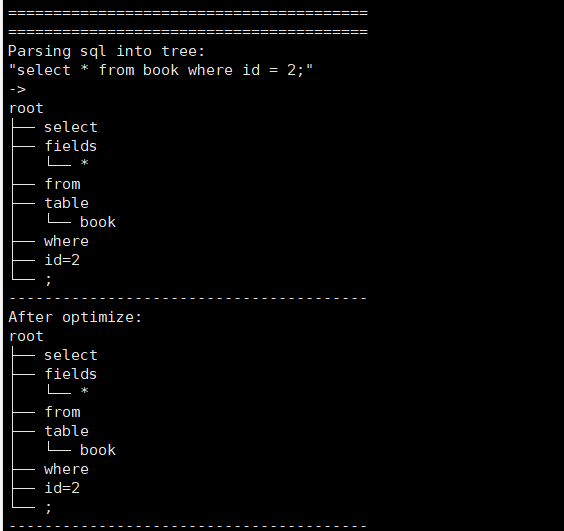


图 8.5: 查询数据

由于两个表都在8002节点，所以对应的查询操作都在8002节点进行，经过8002节点的sql语句解析后，进行文件操作，在8002节点的本地文件查询数据。对于有where条件限制的查询，对查询出来的数据筛选后再显示；对于不存在的数据的查询，会返回空。数据查询后，8002节点屏幕以垂直划分输出信息如下图所示，表示以成功查询。



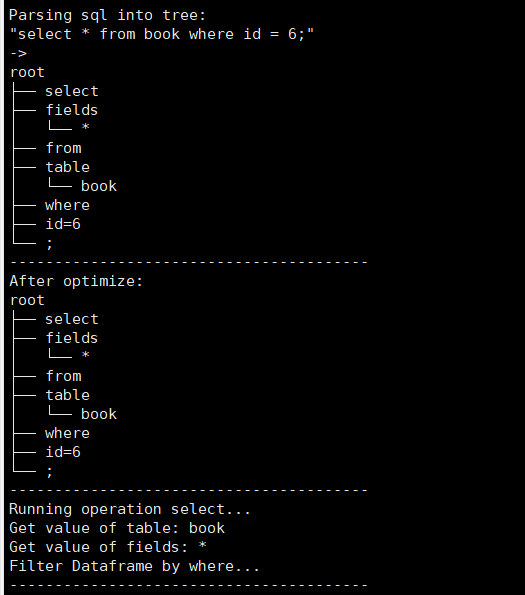


图 8.6: 查询数据8002的屏幕输出

**（四）删除数据测试**

当数据表中存在数据后，就可以通过sql命令，在对应的表中删除数据，如下图所示，在客户端中分别执行图书表和班级表执行删除数据的命令。

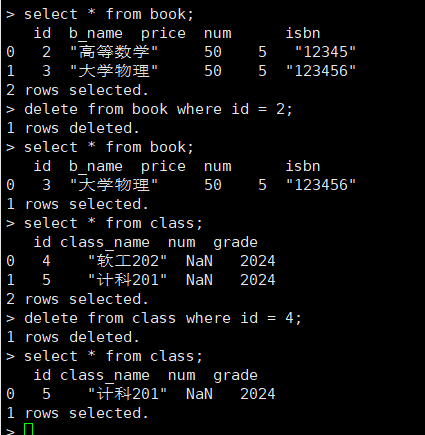
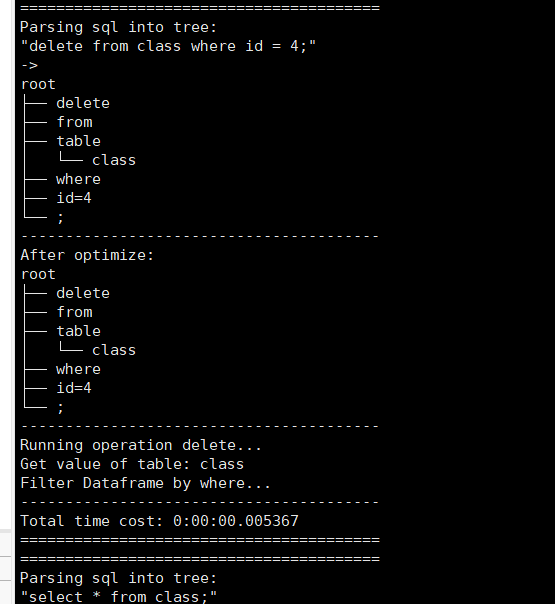


图 8.7: 删除数据

由于两个表都在8002节点，所以对应的删除操作都在8002节点进行，经过8002节点的sql语句解析后，进行文件操作，在8002节点的本地文件删除数据。数据删除后，8002节点屏幕以垂直划分输出信息如下图所示，表示以成功删除。



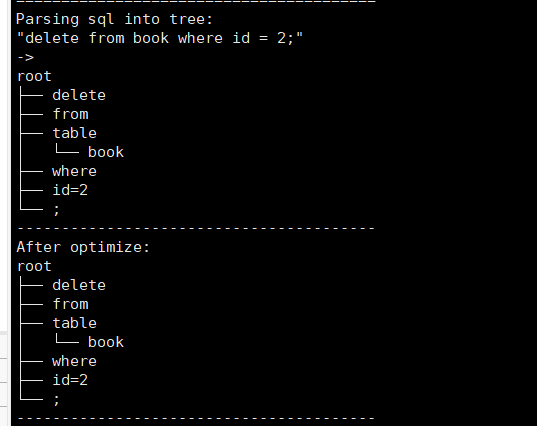


图 8.8: 删除数据8002节点的屏幕输出

最后，使用drop 命令删除 book 表后，再次使用查询命令，将返回”Table book is not existed!” 错误，说明表也已经被成功删除。

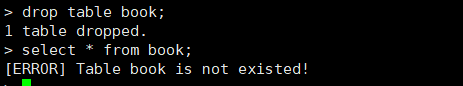


图 8.9: 删除表

至此，本系统的创建表、新增数据、查询数据、删除数据、删除表，以及 where

子句的解析、数据表的垂直拆分等功能都得以验证，满足实验要求！

## 九、 实验结论

本实验实现了一个小型分布式图书馆系统的数据库系统的设计和开发，成功完成了实验的全部要求。

# 十、 总结及心得体会

我本次分布式数据库系统实验是一次富有挑战性和实践性的学习经历。通过一系列精心设计的实验环节，我深入了解了分布式数据库系统的核心原理、关键技术以及实际应用。

我学习了如何在分布式环境中执行高效的查询操作，包括使用索引、优化查询语句、利用并行处理等技术来提高查询速度。同时，我也了解了如何在分布式数据库系统中处理复杂的事务和数据一致性问题，确保数据的准确性和一致性。

本次实验让我深刻体会到理论与实践相结合的重要性。只有将理论知识应用到实践中，才能真正理解和掌握分布式数据库系统的核心技术和应用方法。同时，实践也能帮助我们更好地理解和记忆理论知识，提高学习效果。