《大型平台软件设计实习》

指导书

编撰人：姚鑫、桂宁、邓磊

中南大学计算机学院

2025 年 4 月

一、 实习目的

本实习以分步构建“数据库系统”为总体目标：首先运用编译原理知识实现一个 SQL 编译器，再结合操作系统知识设计一个简化存储系统，最终在数据库知识的支撑下，基于前两部分内容实现一个小型数据库系统。

通过循序渐进的任务设计，本实习旨在贯通编译原理、操作系统与数据库的核心知识点，使学生能够系统化理解并体验“从语言到系统”的完整实现过程。

1. SQL 编译器

- 掌握词法分析、语法分析、语义分析及执行计划生成的基本方法；熟悉 SQL 语句到执行计划的转换流程；能够独立实现一个简化 SQL 编译器，为后续数据库查询子系统提供编译前端支撑；
- 培养将编译原理理论应用于实际问题求解的能力，提升独立完成小型编译系统的设计、实现与调试能力；
- 通过系统化训练，强化逻辑抽象、模块化设计与问题分解能力，为后续整合操作系统与数据库知识的综合实训奠定基础。

2. 操作系统知识的实践

- 通过简化存储系统设计与缓存机制实现，加深理解操作系统在数据库系统中的支撑作用；
- 理解文件系统与页式存储管理的基本原理；
- 掌握缓存管理与替换策略的实现方法；
- 能够设计并实现一个简化的页式存储模型，为数据库系统提供底层数据访问与管理接口。

3. 数据库系统的综合构建

- 在前两个模块的支撑下，独立完成一个简化数据库系统的核心功能（表定义、数据存储、基本查询）。
- 加深对数据库执行器、存储引擎与查询语言之间交互机制的理解。

4. 个人综合能力培养

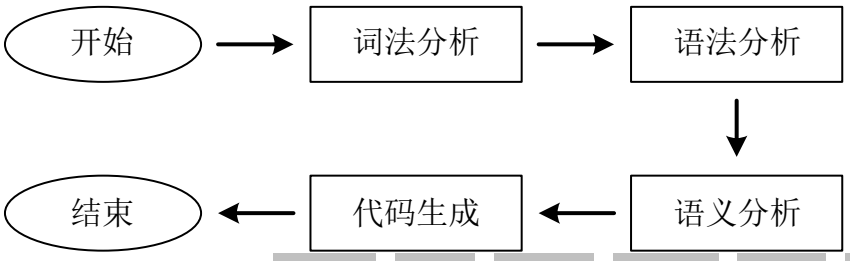
- 通过独立开发过程，提升编程能力、系统设计能力和问题解决能力。
- 训练逻辑思维与实验报告撰写能力，为后续科研或工程项目打下基础。

二、 实践内容

1. SQL 编译器设计与实现：

针对 SQL 语言，完成其词法分析、语法分析、语义分析与执行计划生成的基本实现。

- **词法分析（Lexical Analysis）**



- 识别 SQL 关键字（SELECT，FROM，WHERE，CREATE，TABLE，INSERT，

INTO，VALUES，DELETE 等）、标识符（表名、列名）、常量（数字、字符串）、运算符、分隔符。

- 输出 Token 流，即将源代码切分为一系列有序的基本单元。每个 Token 应包含以下信息：
  - 种别码（Token Type）：如 KEYWORD、IDENTIFIER、CONST、OPERATOR、DELIMITER；
  - 词素值（Lexeme）：源代码中的具体字符串，如 SELECT、student、20；
  - 位置信息：行号与列号，便于错误定位与调试。
- 遇到非法字符或不合法输入时，应输出错误提示（错误类型+位置）。

➢ **语法分析（Syntax Analysis）**

- 基于 SQL 子集文法构建语法树（AST），可采用递归下降、LL(1)、LR 等方法。
- 至少支持 CREATE TABLE、INSERT、SELECT、DELETE 四类基本语句。
- 遇到语法错误时，输出错误信息（含出错位置与期望符号）。

➢ **语义分析（Semantic Analysis）**

- 在 AST 基础上进行：
  - 表/列存在性检查；
  - 类型一致性检查（列类型与输入值是否匹配）；
  - 列数/列序检查（INSERT 值的数量与列数是否一致）。
- 构建并维护模式目录（Catalog）。
- 出错时，输出错误类型、位置与原因说明。

➢ **执行计划生成（Intermediate Representation / Plan Generation）**

- 将语法树转换为执行计划，常见算子包括：CreateTable、Insert、SeqScan、Filter、Project。
- 执行计划输出格式可为树形结构、JSON 或 S 表达式。
- 遇到不支持的语法或缺失的语义信息时，输出 Error 提示。

➢ **输入输出与测试**

- 输入：SQL 文件或标准输入，支持多条语句。
- 输出：Token 流、AST、语义检查结果、执行计划。
- 测试语句示例：
  - ✧ CREATE TABLE student(id INT, name VARCHAR, age INT);
  - ✧ INSERT INTO student(id,name,age) VALUES (1,'Alice',20);
  - ✧ SELECT id,name FROM student WHERE age > 18;
  - ✧ DELETE FROM student WHERE id = 1;
- 错误测试：缺分号、列名拼写错误、类型不匹配、值个数不一致、未闭合字符串等。

➢ **可选扩展**

- 语法扩展：支持 UPDATE、JOIN、ORDER BY、GROUP BY。
- 查询优化：如谓词下推。
- 错误诊断增强：如智能纠错提示。

2. 操作系统知识的实践

理解文件系统与存储管理的基本机制，掌握以“页”为单位的数据组织方法。

- **页式存储模型设计与实现**
  - 设计一个简化的页式存储系统，支持页的分配、释放、读写操作。
  - 每页大小固定（如 4KB），页编号唯一。
  - 支持将数据表映射到页集合中，模拟数据库的物理存储结构；
  - 提供接口供数据库模块调用，如 read_page(page_id)、write_page(page_id,data)。
- **缓存管理与替换策略实现**
  - 实现页缓存机制以提升数据访问效率。
  - 支持常见替换策略，如 LRU（最近最少使用）、FIFO（先进先出）。
  - 设计缓存命中统计与替换日志输出功能。
  - 提供接口供数据库模块调用，如 get_page(page_id)、flush_page(page_id)。
- **接口设计与数据库集成**
  - 提供统一的存储访问接口供数据库系统调用；
  - 支持页级读写、缓存命中统计与页替换日志输出；
  - 与 SQL 编译器生成的执行计划对接，实现数据的物理访问。

3、数据库系统的设计与实现

在前两个模块（SQL 编译器与页式存储系统）的支撑下，完成一个具备核心功能的简化数据库系统。该系统应能接收 SQL 语句，并正确执行数据定义、数据操作与基本查询。

- **执行引擎(Execution Engine)**
  - 解析并执行由编译器生成的逻辑执行计划。
  - 至少实现 CreateTable、Insert、SeqScan、Filter、Project 等基本算子。
- **存储引擎(Storage Engine)**
  - 调用底层页式存储系统提供的接口（get_page,write_page 等），实现表数据在磁盘上的组织、存储与访问。
  - 设计记录（Row）与页（Page）之间的映射关系。
- **系统目录(System Catalog)管理**
  - 维护数据库的元数据（表名、列名、列类型等）。
  - 系统目录本身作为一张特殊的表进行存储和管理。
- **接口与交互**
  - 提供命令行界面（CLI）或 API，允许用户输入 SQL 语句，并返回执行结果或错误信息。
- **需实现的核心操作**
  - CREATE TABLE：根据语句创建表结构，并在系统目录中注册元数据。

- INSERT：将数据记录转换为二进制格式，并通过存储引擎接口写入到相应的数据页中。
- SELECT（含 WHERE 条件）：通过存储引擎接口读取数据页，由执行引擎调用 SeqScan 和 Filter 等算子进行处理，返回符合条件的数据。
- DELETE：定位到满足条件的记录，进行删除标记或物理删除。

三、 实训要求：

1. SQL 编译器：
   - 必修对 SQL 语言的**关键字、标识符、常量、运算符、分隔符**等的词法规则、语法规则和语义规则作出明确规定；
   - 正确设计和维护**模式目录（Catalog）**，编写的词法分析器能够识别输入 SQL 源程序中的单词符号，并以**[种别码，词素值，行号，列号]**的形式输出。遇到非法字符或错误情况，应输出错误提示及其在源程序中的位置；
   - 编写**语法分析器**，能够根据 SQL 子集文法（支持 CREATE TABLE、INSERT、SELECT、DELETE 四类语句）输出对应的抽象语法树（AST）。若出现语法错误，应给出错误提示、出错位置及提示期望符号；
   - 编写**语义分析器**，能够对语法树进行**存在性检查、类型一致性检查、列数/列序检查**，并维护 Catalog。输出结果应为语义分析结论或错误提示，格式为**[错误类型，位置，原因说明]**；
   - 编写**执行计划生成器（目标代码生成器）**，能够将 SQL 语句的 AST 转换为逻辑执行计划，输出形式可以是树形结构、JSON 或 S 表达式。遇到不支持的语法或缺失的语义信息时，应给出错误提示。
2. 操作系统页面管理
   - 页式存储模型设计与实现
     - 设计一个简化的页式存储系统，支持页的分配、释放、读写操作；
     - 每页大小固定（如 4KB），页编号唯一；
     - 支持将数据表映射到页集合中，模拟数据库的物理存储结构；
     - 提供接口供数据库模块调用，如 read_page(page_id)、write_page(page_id, data)。
   - 缓存管理与替换策略实现
     - 实现页缓存机制，提升数据访问效率；
     - 支持常见替换策略，如 LRU（最近最少使用）、FIFO（先进先出）；
     - 设计缓存命中统计与替换日志输出功能；
     - 提供接口供数据库模块调用，如 get_page(page_id)、flush_page(page_id)。
   - 接口设计与数据库集成
     - 提供统一的存储访问接口供数据库系统调用；
     - 支持页级读写、缓存命中统计、页替换日志输出；
     - 与 SQL 编译器生成的执行计划对接，实现数据的物理访问。

3. 数据库系统：
   - 在 SQL 编译器与页式存储系统的基础上，构建一个完整的、可交互的数据库系统原型。
   - 系统集成：将 SQL 编译器生成的执行计划与存储引擎的访问接口进行对接，确保执行引擎能正确调用 read_page、write_page 等函数完成数据的物理读写。
   - 数据持久化：所有表数据和元数据（系统目录）必须通过页式存储系统进行持久化存储，程序重启后数据不丢失。
   - 核心功能实现：系统必须正确支持 CREATE TABLE、INSERT、SELECT（含条件查询）、DELETE 这四条核心 SQL 语句的执行。
   - 正确性验证：需设计多种测试用例，验证数据库系统在各种场景下的行为是否正确，包括但不限于：
     - 表的创建与重复创建错误处理。
     - 大量数据的插入与查询。
     - 条件查询的准确性。
     - 删除特定记录后再次查询的结果。
     - 程序重启后的数据持久性。
   - 输出与反馈：系统应能清晰地向用户返回执行结果（如查询结果集、操作成功的提示）或详细的错误信息（如语法错误、语义错误、I/O 错误）。

四、 实训步骤

1. SQL 编译器：
   - **规则定义**
     - 明确 SQL 子集的词法规则、语法规则与语义规则，建立关键字表、符号表（标识符表）、常量表示方式，并初始化 Catalog 结构；
   - **词法分析**
     - 从源程序依次读入字符，对 SQL 语句进行单词切分与识别，直至程序结束；
     - 对正确的单词，以四元式的形式输出结果：**[种别码，词素值，行号，列号]**；
     - 对非法单词或未识别符号，给出错误提示，并标明代码位置；
   - **语法分析**
     - 对合法 Token 流进行语法分析，构建抽象语法树（AST）；
     - 输出分析过程或最终的语法树结构；
     - 对不正确的源程序，输出语法错误提示，包括出错位置和期望符号；
   - **语义分析**
     - 在 AST 基础上，进行表/列存在性检查、类型一致性检查、列数/列序检查；
     - 对正确的语句，输出语义检查通过信息；
     - 对存在问题的语句，输出错误提示，格式为**[错误类型，位置，原因说明]**；

- **执行计划生成**
  - 将语法树转换为逻辑执行计划，输出算子结构（如 CreateTable、Insert、SeqScan、Filter、Project 等）；
  - 输出可采用树形结构、JSON 或 S-表达式；
  - 对不支持或错误的语句，给出错误提示；
- **结果输出与测试**
  - 对正确的 SQL 语句，依次输出 **Token 流→AST→语义检查结果→执行计划**；
  - 对不正确的语句，输出对应阶段的错误提示；
  - 使用给定测试语句（CREATE TABLE student...、INSERT ...、SELECT ...、DELETE ...）和错误语句（缺分号、列名错误、类型不匹配、字符串未闭合等）进行验证。
  - SQL 编译器：

2. 磁盘存储系统设计
   - **页式存储系统设计**
     - 定义页结构（如页大小为 4KB）；
     - 实现页分配与释放函数；
     - 实现页读写函数，模拟磁盘 I/O。
   - **缓存机制实现**
     - 设计缓存结构（如使用 OrderedDict 实现 LRU）；
     - 实现缓存命中判断与替换策略；
     - 实现缓存刷新与持久化机制。
   - **接口设计与集成**
     - 设计统一的存储访问接口；
     - 与 SQL 编译器生成的执行计划对接；
     - 实现数据表与页的映射关系。
   - **测试与验证**
     - 构造模拟数据表，执行插入、查询、删除操作；
     - 验证页分配与释放是否正确；
     - 验证缓存命中率与替换策略效果；
     - 输出日志与统计信息。
3. 数据库系统设计
   - 系统架构设计
     - 设计数据库系统的整体架构，明确执行引擎、存储引擎、系统目录三大模块之间的调用关系和数据流。
     - 定义执行引擎与存储引擎之间的接口（如：如何请求下一页数据、如何写入一条记录）。
   - 系统目录实现

- 将系统目录实现为一张特殊的表（通常命名为 pg_catalog 或 sqlite_master），其本身也通过存储引擎进行读写。
- 实现目录的初始化、查询（如根据表名获取表结构）和更新（如新建表后注册）功能。

➢ 执行引擎实现

- 实现各执行算子（CreateTable，Insert，SeqScan，Filter，Project）。
- SeqScan 算子需迭代访问指定表的所有数据页。
- Filter 算子需根据 WHERE 条件对记录进行过滤。
- Project 算子需根据 SELECT 指定的列对记录进行投影。

➢ 存储引擎集成

- 实现记录（Row）与页（Page）的序列化（Serialization）与反序列化（Deserialization）方法。
- 在存储引擎中管理空闲页列表，支持表的扩展（分配新页）和回收。

➢ 系统测试与验证

- 编写综合测试脚本，按顺序执行一系列 SQL 语句（建表→插入→查询→删除→再查询）。
- 验证结果是否正确，并检查底层数据页的内容变化是否符合预期。
- 验证数据持久性：重启程序后，重新连接数据库，检查之前创建的表和数据是否依然存在并可查询。
- 输出测试报告，包括测试用例、执行过程、结果截图和结论。

五、 实训报告

1. 完成本实践后，应提交一份完整的实践报告。
2. 报告内容需包括：词法分析、语法分析、语义分析的详细设计思路，以及存储管理、缓存机制、访问接口设计与测试方法的完整说明。
3. 实验报告格式与要求见附件（供参考，可根据课程要求调整）。

附件：实践报告格式与要求

《XXXXX（课程名称）》

实践报告

项目名称 _____

专业班级 _____

学　　号 _____

姓　　名 _____

实验成绩：

批阅教师：

年　月　日

正文要求

《XXXXX（项目名称）》

一、实践目的

指出此次实践应该达到的学习目标。

二、实践内容

指出此次实践应完成的任务。

三、实践方法

包括实践方法、原理、技术、方案等。

四、实践步骤

指出完成该实践的操作步骤。

五、实践结果

记录实践输出数据和结果。

六、实践结论

对实践数据和结果进行分析描述，给出实践取得的成果和结论。

注：有程序的要求附上程序源代码，有图表的要有截图并有相应的文字说明和分析

七、实践小结

给出本次实践的体会，如学会了什么，遇到哪些问题，如何解决这些问题，存在哪些有待改进的地方。