简易单用户关系数据库管理系统
设计报告

计 82　郑凯文　2018011314

2021 年 1 月 16 日

目录

1 项目简介

本项目是一个 C++ 实现的简易的单用户数据库管理系统，支持一些常见的 SQL 类型、语句和功能。使用提供的页式文件系统进行缓存和文件存储，使用 B+ 树进行索引管理，使用 Flex/Bison 进行命令解析。

截止目前，除页式文件系统外的 C++ 代码行数约为 6600。

1.1 编译运行方式

在 Linux/WSL 环境下，依赖 Flex/Bison，并需要编译器支持 C++11 特性。项目基于 CMake 进行自动构建，运行 build.sh 后可在 build/build 文件夹下找到可执行文件 db。

程序运行后，在终端持续接收输入流并解析。若语法错误，程序控制 Flex/Bison 跳过错误部分并继续解析。值得注意的是，程序不会在运行过程中主动将缓存写回文件，若在终端中使用 Ctrl+C 终止程序，再次运行数据库后之前数据将丢失。若希望保留上次数据，应键入 EXIT; 命令退出。

1.2 功能概述

1.2.1 数据类型

支持的数据类型有：

- 整型（INT, INTEGER），长度为 4 字节
- 浮点型（FLOAT, DOUBLE, NUMERIC, DECIMAL），长度为 4 字节
- 日期型（DATE），长度为 4 字节，格式为'YYYY-MM-dd'
- 字符串型（CHAR, CHARACTER）
- 变长字符串型（VARCHAR），支持超过页大小限制的长度

1.2.2 SQL 语句

支持的 SQL 语句包括：

- 系统管理：创建数据库，使用数据库，删除数据库，显示数据库信息，创建表，删除表，显示表信息
- 查询解析：增删查改，COPY FROM

- 表修改：添加删除主键、外键、UNIQUE 约束、索引、列；表的易名

具体文法见 sql.y 文件。

1.2.3 约束

支持 6 种类型的约束：

- 主键约束，一个表只能有一个主键，可以是联合主键；主键需满足 UNIQUE 和 NOT NULL
- 外键约束，支持联合外键，外键指向的必须是其它表的主键
- UNIQUE 约束，作用于某一列，限制列的值不能重复
- NOT NULL 约束，限制某一列不能为空
- DEFAULT 约束，插入行/添加列未指定时，赋予一个默认值
- CHECK 约束，检查插入的行是否满足指定表达式

1.2.4 复杂表达式

在 INSERT 语句、CHECK 约束、WHERE 子句、UPDATE 语句的 SET 子句中可以使用复杂表达式。使用 Bison 文法进行表达式解析，支持嵌套。基本运算包括：

- 四则运算：+, -, *, /
- 比较运算：<, >, =, <=, >=, !=, <>
- 模糊匹配运算：LIKE，支持%, _, ., * 以及使用 \ 进行转义
- 范围匹配运算：IN
- 判空运算：IS NULL 和 IS NOT NULL
- 逻辑运算：NOT AND OR
- 聚集函数：COUNT SUM AVG MIN MAX，其中 COUNT 的对象可以是 *，其余 4 个必须是某一整型或浮点型的列

1.2.5 多表连接与查询优化

支持多表联合的 SELECT 查询，并进行查询优化，通过了最终 5 表联合的测例。

1.2.6 简单 WHERE 嵌套

支持在 WHERE 子句的 IN 语句中使用嵌套的 SELECT，但要求次级 SELECT 的选择的列数为 1。

2 架构设计

整个系统可以分为如下几个模块：

- 页式文件系统：使用提供的页式文件系统，每个文件使用一个缓存管理器
- 记录管理器：使用一个文件进行完整记录的顺序存储并维护 RID，提供 RID 有关的记录添加、删除、更新和遍历，底层为页式文件系统
- VARCHAR 管理器：使用一个文件进行 VARCHAR 的页式存储，底层为页式文件系统
- 索引管理器：使用一个文件存储与管理被索引部分的页式 B+ 树，提供判重、第一关键字的范围查询，主键、UNIQUE 约束默认创建，底层为页式文件系统
- 表管理器：维护一张表，内部有一个记录管理器、一个 VARCHAR 管理器和若干索引管理器
- 数据库管理器：采用单例模式，进行系统管理、处理查询与优化，并将表相关操作转发给对应表管理器
- 语句解析器：使用 Flex 和 Bison 进行语句解析，并调用数据库管理器执行

3 模块设计与功能实现

3.1 记录管理

使用一个文件进行完整记录的存储（tablename.record）。文件的第 0 页为文件头，之后的每一页页头用一个 bitmap 存储页中所有空余位置，其余空间被划分为若干槽，每个槽内放置一条记录，格式为标识各列是否为 null 的 bitmap+ 各列数据。记录的 RID 可以由页号和槽号进行运算得到，也就是说 RID 与存放在文件中的位置一一对应。

我还写了一个页式的 VARCHAR 系统来存储 VARCHAR，它的数据部分被存放在一个单独的文件里（tablename.varchar），在记录管理器中只保留一个 4 字节页号。VARCHAR 文件第 0 页为文件头，之后每页有页头，剩余为 VARCHAR 数据，超出一页的数据会被放在多个页中，在页头中用类似于链表的结构管理。但它存在一个问题，当 VARCHAR 较短时，每个

VARCHAR 仍然会占用 1 页，因此我在测试小型数据集时，VARCHAR 文件动辄上百 MB 甚至 GB，占用磁盘空间过大，且拖慢了数据库的速度。因此，最后的代码里我仍将 VARCHAR 当做 CHAR 对待。看起来页式的 VARCHAR 并不是一个好的解决方案。

3.2 索引管理

索引管理模块可以对多列建立索引。文件的第 0 页为文件头，其余每页为 B+ 树的一个节点。页中除页头外，有若干个 4 字节整数，指向孩子节点的页号，还有同样数目的数据槽。数据槽中，键以 RID+ 标识各列是否为 null 的 bitmap+ 各列数据的格式存在。进行比较时，先按顺序对各列数据进行比较（非 null>null），数据相同时对 RID 进行比较，这样可以保证两个键必一大一小。

索引管理模块提供被索引列重复性的判定以及第一关键字的范围查询。索引以三种形式存在：

- 主键自动建立索引，文件名为 tablename.primary
- 具有 UNIQUE 约束的列自动建立索引，文件名为 tablename.unique.columnname
- 手动创建索引，文件名为 tablename.index.indexname

3.3 系统管理

系统管理模块由表管理器和数据库管理器共同完成。根目录下.database 文件中记录各个数据库名称，每个数据库在同名文件夹下。数据库文件夹下，.tables 文件记录了各个表的名称，tablename.header 记录了表头。

创建、删除数据库即文件夹的创建删除，使用数据库即切换当前目录。创建表时，首先构造表头，再利用表头创建空表（包括创建记录管理器、VARCHAR 管理器、主键和 UNIQUE 约束的索引管理器）。删除表时，删除表对应的所有文件。在以上操作中，进行各种指令合法性检查，并维护各处数据的一致性。

添加删除主键、外键、UNIQUE 约束、索引时，进行各种合法性检查，若不合法（如删除的主键被其他表 refer，添加 UNIQUE 约束时已有数据存在重复），则拒绝改变。否则，修改表信息，并进行诸如创建主键索引的操作。

添加列和删除列是很昂贵的操作。首先进行合法性检查（如添加列时若要求 NOT NULL 则必有 DEFAULT，删除的列不能被其它表 refer），并修改表信息（如重新计算各列的偏移，删除含有被删除列的 CHECK 约束），得到一个新的表头。将原先的表易名，利用新表头创建新表，将原表中所有记录全部插入新表，再删除原表。

3.4 查询解析

插入和更新记录时，对各种约束进行检查。主键和 UNIQUE 约束的唯一性都是通过查询索引来保证的，外键则查询 refer 的表的主键索引来确保存在性。

更新和删除记录时，没有实现级联更新删除，而是采用 restricted 模式，即若被 refer 则禁止更新和删除。

除插入之外，均可能涉及到单个表的 WHERE 筛选。对于单个表，将 WHERE 表达式用 AND 分割为子表达式，对于 column comparator constant 形式的子表达式，若存在列为第一关键字的索引（无论是主键索引、UNIQUE 索引或主动创建的索引），则利用索引进行查询。否则，通过评估表达式的值进行筛选。

对于设计到多个表的 WHERE，采取如下优化策略：

- 将表达式分割为 AND 子句
- 收集只涉及单个表的子句，并进行单个表的筛选
- 收集涉及两个表的子句，对于 A.col1=B.col2 形式，若某一列具有索引，则固定另一列的值，利用索引查询；否则，进行排序后使用二分搜索。对于非 A.col1=B.col2 的形式，普通地计算一下表达式的值并进行筛选。
- 将以上两步的结果合并
- 利用剩余（涉及 3 个及以上表）子句进行最后的筛选

由于取决于具体情况，这里并没有对表的顺序按某种策略进行重排，而是由用户手动调整表的顺序。

对于 WHERE 子句的 IN 中嵌套的 SELECT，首先递归地执行次级 SELECT，将得到的结果作为一个列表，再当做普通的 IN 那样进行判断。

4 实验结果

通过了助教 34 条测例中的 32 条，未通过的有 Group by 和外键的级联更新删除。此外，额外编写了一些测例来验证复杂表达式、CHECK 约束及一些边界情况。

在测试过程中，需对助教提供的语句进行适当修改以符合我的文法。由于文法只支持 YYYY-MM-dd 格式的日期，我写了一个简单的 Python 脚本来对 tbl 文件进行转换。此外，由于 tbl 文件中每行的最后一列数据（一般是一个很长的字符串）后没有分隔符，程序在 COPY FROM 时没有读取而是插入为 null，不过这对于测试无伤大雅，因此没有做多余的处理。

5 小组分工

我独立完成了所有工作。

6 参考资料

文法与表达式解析部分参考了往届的开源项目TrivialDB。