**《数据库原理》大作业要求**

## 一、概述

基于给出的ThssDB框架，补充、扩展、实现一个关系型数据库管理系统。

## 二、核心模块

## 2.1通信模块

系统整体采用客户端/服务器架构，RPC采用Apache Thrift。框架中的client和server包分别用来管理客户端和服务器，thrift包下的rpc.thrift文件中给了一个定义返回当前时间（getTime）服务的例子，还定义了连接数据库（connect）、断开连接（disconnect）和执行语句（executeStatement）这三个服务，需要同学们自行实现。此外，可以根据自身需要定义其他服务。

**基础要求：**

1. 实现thrift/rpc.thrift中的connect、disconnect和executeStatement服务。

## 2.2异常处理模块

框架中的exception包用来管理数据库中涉及到的各种异常类。已给出的两个异常类DuplicateKeyException和KeyNotExistException是B+树中用到的。同学们需要在实现下面几个模块时做好异常处理，并在最终的大作业报告中对自定义的异常类予以说明。

## 2.3存储模块

完成存储引擎，实现对数据的基本管理。框架中的schema包定义了一些基本概念：

Column类：一张表的某一列的元信息。成员变量包括该列的名字（name）、数据类型（type）、是否为主键（primary）、是否可以为空（notNull）以及最大长度（maxLength，仅限于String类型）。

Entry类：某一行记录在表中某一列的数据。

Row类：某一行记录在表中所有列的数据。

**基础要求：**

1. 利用java的序列化和反序列化实现记录的持久化；
2. 实现对记录的增加、删除、修改、查询；
3. 支持五种数据类型：Int，Long，Float，Double，String。

进阶要求：

1. 实现高效的文件存储格式，如页式存储等。

## 2.4元数据管理模块

在存储引擎的基础上，实现元数据管理引擎。框架中的schema包定义了一些基本概念：

Table类：表的信息。一个Database可以有多张Table。

Database类：数据库的信息。用户可以创建多个Database。

Manager类：管理Database。

此外，框架中的index包涉及索引的元数据，使用B+树实现。

BPlusTree类：用主键entry做key，用真实的row作为value。

BPlusTreeNode类：B+树结点抽象类。

BPlusTreeInternalNode类：B+树内部结点类。存储key数组和children数组来支持B+树的索引。

BPlusTreeInternalNode类：B+树叶子结点类。存储key数组和value数组来支持B+树的索引和真实数据的存储，同时维护一个next指针以支持所有叶子结点可以高效遍历。

BPlusTreeIterator类：B+树的迭代器类。通过在B+树最底层的的叶子结点维护一个链表就可以高效的实现范围查询和遍历。

**基础要求：**

1. 实现表的创建、删除、修改；
2. 实现数据库的创建、删除、切换；
3. 实现表和数据库的元数据（有哪些数据库，数据库里有哪些表，每个表的结构如何）的持久化。
4. 重启数据库时从持久化的元数据中恢复系统信息。

## 2.5查询模块

实现查询引擎，包括SQL解析和查询执行两部分。其中，parser包给出了利用antlr4实现SQL解析的例子。框架中的query包提供了一些可能用到的基础类：

MetaInfo类：存储一个表的名字和所有列，用来在Queryresult中构建用户想要获取的列的索引。

QueryResult类：选择运算中选择某些列或全部列的结果，需要对歧义列名或不存在列名、不存在表名等错误情况进行捕捉。

QueryTable类：选择运算中选择表的方法类，用迭代器的方式从当前表中不断获取新的行并判断其是否满足where中条件。

**基础要求：**

1. 支持下列查询语句（不区分大小写）：

* 创建表：CREATE TABLE *tableName(attrName1 Type1, attrName2 Type2,…, attrNameN TypeN NOT NULL, PRIMARY KEY(attrName1))*，实现“NOT NULL”和“PRIMARY KEY”这两个关键字，Type为Int，Long，Float，Double，String（必须给出长度）之一。主键仅在某一列定义。

示例: CREATE TABLE *person (name String(256), ID Int not null, PRIMARY KEY(ID))*

* 删除表：DROP TABLE *tableName*

展示某张表的模式信息(每一列的名字及其数据类型，格式自定)：SHOW TABLE *tableName*

* INSERT INTO [*tableName*(*attrName1, attrName2,…, attrNameN*)] VALUES (*attrValue1, attrValue2,…, attrValueN*)，字符串需要用单引号包围。

示例：INSERT INTO person VALUES (‘Bob’, 15)

INSERT INTO person(name) VALUES (‘Bob’)会提示字段ID不能为空。

* DELETE FROM *tableName* WHERE attrName = attValue
* UPDATE *tableName SET attrName = attrValue WHERE attrName = attrValue*
* SELECT *attrName1, attrName2, … attrNameN*  FROM  *tableName [* WHERE *attrName1 = attrValue ]*
* SELECT tableName1.AttrName1, tableName1.AttrName2…, tableName2.AttrName1, tableName2.AttrName2,… FROM tableName1 JOIN tableName2 ON tableName1.attrName1 = tableName2.attrName2 *[* WHERE *attrName1 = attrValue ]*

SELECT attrName1, attrName2,attrNameN FROM tableName WHERE attrName1 = attrValue

上述语句中，Where子句至多涉及一个比较，并且关系为‘’<“,”>”,”<=”,”>=”,”=”,”<>”之一。Select子句不包含表达式。Join至多涉及2张表，On子句的限制同Where子句。

进阶要求：

1. 应用课程中介绍的查询优化技术
2. 支持多列主键；
3. where条件支持逻辑运算符（and/or）；
4. 实现三张表以上的join；
5. 实现outer join等其他类型的join；
6. 其他标准SQL支持的查询语法。

## 2.6事务与恢复模块

实现简单的事务及恢复功能，支持小规模的并发。

**基础要求：**

1. 实现begin transaction和commit；采用普通锁协议，实现read committed的隔离级别即可；
2. 实现单一事务的WAL机制，要求实现写log和读log，在重启时能够恢复记录的数据即可。

进阶要求：

1. 实现多事务的并发和恢复机制；
2. 实现更高级的隔离级别（repeatable read/serializable）；
3. 实现rollback、savepoint等功能。
4. 实现2PL或MVCC协议。

## 2.7 说明

框架所提供的类及方法均为建议和参考，大家可以自行选择利用或者不利用或者部分利用这些类及方法，非常鼓励大家设计出性能更优的架构。

## 三、作业要求

1. 组队人数：2~3人
2. 逐步完成系统的各个模块，逐次提交设计文档，最终提交需要提交设计文档、展示PPT及源代码。

|  |  |
| --- | --- |
| 提交项 | 截止日期 |
| **存储模块** | 5月3日 |
| **元数据管理模块** | 5月10日 |
| **查询模块** | 5月21日 |
| **事务与恢复模块** | 5月31日 |
| **最终提交** | 6月7日 |

1. 项目检查（机动安排）

评分依据（满分100分，完成额外功能可以加分，最高不超过110分）：以教材给出的大学数据库为测试的schema

基础要求完整、正确（90分）；

以HSQLDB为基准，评测性能（10分）；

进阶要求，酌情给分（不超过10分）。

1. 加分项：所有其他的扩展（支持更多的SQL、性能优化、权限认证、触发器、约束等）都可以作为加分项。
2. 课堂展示（第16周）：PPT（系统介绍）。