Interpreter 模块设计报告

3180101972_蔡灿宇

1. 任务概述

- **直接与用户交互，处理用户输入信息：**

  程序流程控制，即“启动并初始化——接收命令——处理命令——显示命令结果”的循环、退出流程。

- **进行语法分析与正确性判断：**

  接收并解释用户输入的命令，生成命令的内部数据结构表示，同时检查命令的语法正确性和语义正确性，对正确的命令调用API层提供的函数执行并显示执行结果，对不正确的命令显示错误信息。

- **解析函数回调，向用户输出结果：**

  通过调用的API函数返还的内部数据结构，进行解析与输出格式整理，向用户打印出整齐格式的数据。

2. 功能描述

- 数据类型
  - 要求支持三种基本数据类型：int，string，float
- 表定义与删除
  - 一个表最多可以定义32个属性。
  - 支持单属性的主键定义，对没定义主键的表格进行提示抛出。
  - 支持表格的删除。
- 索引的建立和删除
  - 对于表的主属性自动建立B+树索引，所有的B+树索引都是单属性单值的。
  - 索引支持为空，能够根据用户的需要删除索引。

- 查找记录
    - 可以通过指定用and或or连接的条件进行查询。
    - 支持where语句后带多个条件。
- 插入和删除记录
    - 支持每次一条或多条记录的插入操作。
    - 支持每次一条或多条记录的删除操作。
- 更新记录
    - 支持每次一条或多条记录的更新操作。

3. 语句说明

create table

```
create table student (
        sno string,
        sname string,
        sage int,
        sgender float,
        primary key (sno)
);
```

create index

```
create index stunameidx on student (sname);
```

drop table

```
drop table student;
```

drop index

```
drop index stunameidx on student;
```

select from

```
select sno from student where sno != '3180100000';
select * from student where sno = '3180100000';
select * from student where sage > 21 and sno != '3180100004';
select t1.name,t2.name from cip_temps t1 inner join cip_tmp t2 on
t1.ID=t2.id;
```

insert into

```
insert into student values ('12345678','y',22,'M');
```

delete from

```
delete from student;
delete from student where sno = '88888888';
```

update

```
update student set sno = '123' where sage = 21;
```

4. 接口说明与工作原理

4.1 接口说明

- 外部接口
    - 本系统的外部接口为控制台。
    - 使用者通过控制台的字符界面输入查询语句使用MiniSQL。输入方式为键盘输入。
- 内部接口
    - 本模块提供`Interpreter.initial()`接口供`ClientRunner`启动程序调用，作用是初始化与启动模块。
    - 本模块采用类创建与函数调用的方式。调用`RecordManager`、`CatalogManager`等提供的接口，作用是将语法分析后得到的数据结构进行传输并生效，并将返回值解析输出。

4.2 工作原理

- 通过`Interpreter.interpreter()`处理输入的指令，再通过初步分析指令类型并跳转到其他函数。
- 其他每个功能函数逐个解析指令，准确保存任何有用信息，对错误信息进行抛出。
- 将指令分别保存为attribute、condition等信息，传递到其他模块的接口完成操作。
- 接受函数的返回值，分析判断并输出。
- 最后通过quit退出程序

5. 出错处理分析

- Interpret的语法分析

  在Interpret模块中采取了相应的方式判断输入是否符合语法，若不符合会提示语法具体错误并重新输入。

- 查询结果异常

  若出现表格、索引已存在或不存在的问题，则会返回出错的具体信息，由本模块抛出，并输出具体错误信息，提供重新输入。

6. 开发体会

在开发过程中，体会到interpreter模块的不容易之处。

不仅仅要考虑到各个语句的语法分析，更要设想多个设计样例的范本，设想多个情况，考虑到所有的输入出错情况，并设定对应的提醒与抛出。而且由于自定义了许多数据结构，需要做好流程图，对数据结构进行层层包装与层层解析，避免出现错误。

而在小组合作开发过程中，更加体会到整体合作到时候分工、文档的重要性。对于数据结构的定义需要提前考虑周到并且保证不再更改，否则如果在开发过程中再更改部分数据结构会导致很多工作需要重新再来，非常麻烦。

而在大规模、分布式合作过程中，也学会了zookeeper等集群的使用方式，了解thrift的协议，对日后有更大的方便。