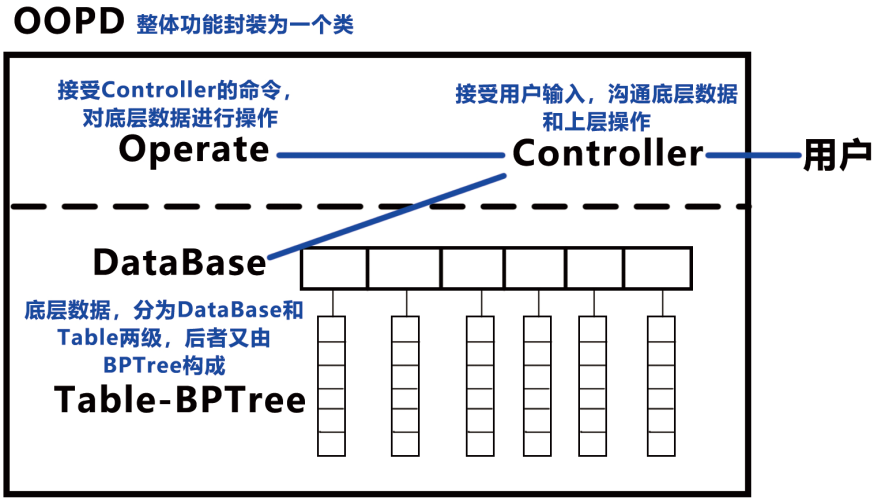
使用说明

1 总体架构

OOPDB类是对数据库的底层数据结构/底层接口、对数据的上层操作、总控的整体封装。

上述三个模块被划分为六个类：

- Controller类负责总控，以及与用户交互（读取指令）
- Operate类负责对数据进行一些上层操作
- BPTree类和BPTreeNode类是底层实现，同时BPTree类有一些底层接口，将被Operate类的方法调用
- Table类管理着整个Table的列信息，并存储着多个BPTree类对象（索引树）以供需要时使用
- DataBase类包含一个Table对象的数组，同时OOPDB类中有一个DataBase对象的数组



2 各模块简介

2.1 数据部分

数据部分包含多个类，具有多级关系，其关系为：OOPDB类中有一个DataBase容器(map)，每个DataBase对象中有一个Table容器(map)，每一个Table对象中有多个BPTree指针。BPTree对象中仅保存指向数据的指针，而实际的数据都建立在堆上，由Operate中的方法负责创建与销毁。

“每一行”数据存储为一个Data对象，其中包含三个数组(vector)，分别对应int, double, char三种数据类型。

底层数据结构为B+Tree，能够实现快速检索。

Table层级的管理方式是：在Table对象中通过map容器保存数据表中各列的信息，包括数据类型、元素在Data对象中的存储位置、NOT NULL属性等，可以通过读取Table对象中存储的信息来正确地访问Data对象中的各列数据。Table

中含有多个BPTree指针，构造BPTree对象的原则是对每个主键建一棵树用于索引，在一阶段仅有一个主键，但是支持同时存在多个主键。

Table和BPTree类提供了一些底层接口，可以实现一些基本的数据操作，Operate类的方法综合使用这些接口来实现对数据的上层操作。

2.2 Operate部分

Operate类是将数据库中对数据的全部上层操作（例如插入、删除、修改、检索等）抽象出来，作为一个子类进行封装。

在接口逻辑方面，Operate类为每种上层操作都提供了接口，但是它自己不负责判断应对哪些数据进行处理，而是从Controller类接受传入的参数：其中一类参数是指向执行操作的目标的引用，另一类参数则用于规范相应操作的具体行为。

在方法的实现中，Operate类通过进一步解析Controller类传入的参数，以及综合调用Table、BPTree等类提供的底层接口来实现对数据的上层操作。

2.3 Controller部分

Controller类负责数据库功能的总控，也负责与用户进行交互。

作为总控，Controller可以通过内部存储的一些指向底层数据的指针来访问底层数据，从而获取其地址或者提取一些信息以辅助输入解析；它也含有指向Operate对象的指针，用于调用其方法。

Controller读取用户的输入，将输入解析为应当执行的操作和对应操作的参数，然后从底层数据中找到操作的目标，将其引用和相关参数传递给Operate中的相应方法，从而实现对数据的操作。在这一过程中，Controller也间接沟通了Operate类与底层数据。

3 继续开发的优势

底层数据结构采用B+Tree，并且已经提供了必要的接口，尤其是用于批量搜索的接口。在后续开发中可以对这些底层接口善加利用以提高运行效率。目前执行Where子句的逻辑是仅当表达式中只涉及主键时才执行B+Tree的批量搜索，若二阶段可以设置多个主键，则可以进一步完善Where子句的执行逻辑。

底层的BPTree类是模板类，对数据类型有良好的适应性。

BPTree类已部分支持设置多个主键，若二阶段有相应要求则更容易进行开发。