以 project为例，修改SPJ操作符

存在一个临时表的全局变量，我们把内存中用到的所有中间表以临时数据字典的形式存放，临时数据字典的fileID取负值即为中间表所在的bufferID：

relation \* temp\_data\_dict = new relation[MAX\_FILE\_NUM];

每一个操作符的的传入参数为旧表的临时数据字典的下标，新表属性个数，新表属性列表的数组，返回值为新表的临时数据字典的下标。

SELECT address, name, custkey

FROM customer;

在单表上做投影：

首先调用tablescan函数扫描customer表，把外存中的表读入内存，建立了一个临时数据字典，是外存中customer表的数据字典的一个副本，在temp\_datadic[0]中。Tablescan传入要扫描的表名，返回值为中间表的临时数据字典下标。调用方法如下。

int customer\_scan = -1;

customer\_scan = TableScan(&head, temp\_data\_dict, "customer");

if (customer\_scan>=0) {

printf("tablescan succeed!\n"); }

在这里tablescan结束后customer\_scan值为0。

然后做project 传入临时数据字典的下表号，即0，传入属性个数，即3，传入属性列表，是一个二维数组，利用这两个参数生成新的临时数据字典，返回值为新表的临时数据字典的下标，即1。

char attribute\_list[3][NAMELENGTH] = {"address", "name", "custkey"};

if(project(&head, temp\_data\_dict, 0, 3, attribute\_list)>=0)

printf("project succed!\n");

project中需要做输入检查和异常检查。检查传入的属性列表名字是否正确，检查是否有空余的临时数据字典可用。对于传入的属性列表中的属性顺序与原表中的属性顺序不同的这种情况，我采用的办法是按原表中的属性顺序建新表。

前三个实验没有实现的：

1. delete一行数据 scan判断 玮毓
2. 数据字典没有写到文件（class类无法像struct一样直接写到文件） 梦溪
3. 查询优化，加到数据字典，是否有索引，是否排序 梦溪
4. order by 梦溪
5. ~~indexscan实现范围查找 玮毓~~
6. sort merge join调试 梦溪
7. indexjoin 文慧
8. 三表连接在编译时确定
9. 优化器 chooseJoin(relation \* temp\_datadic, int a, int b, const char \* attribute\_name) 文慧
10. ~~当不用loaddata函数，用insert从空表开始插入，scan时有问题，原因在于找不到file结束的页号pagenum和偏移量offset 天贞~~
11. ~~实验四实验报告 小师~~