Record Manager 模块设计说明

作者：2016 级软件工程专业
傅诤哲 程浩然 叶慕祈 石磊 钱程

Record Manager 负责管理记录表中数据的数据文件。主要功能为实现记录的插入、删除与查找操作，并对外提供相应的接口。其中记录的查找操作要求能够支持不带条件的查找和带条件的查找（包括等值查找、不等值查找和区间查找）。

数据文件由一个或多个数据块组成，块大小应与缓冲区块大小相同。一个块中包含一条至多条记录，为简单起见，只要求支持定长记录的存储，且不要求支持记录的跨块存储。

其具体实现如下：

声明结构体或联合体：

```
struct p {
    string value;
    int n;
};
typedef struct p Pair;

union F {
    char byte[4];
    float f;
};

union I {
    char byte[4];
    int a;
};
```

struct p 是建立索引是，返回给 index manager 的数据结构。value 是键值，n 是表中第几个数据。

union F 是将浮点数与内码相互转化用的。用成员 f 接收浮点数，然后将成员 byte 拷贝进缓冲区，最后保存。读取时也需要这样的联合体结构。

union I 是将整型数据与内码相互转化用的。用成员 a 接收浮点数，然后将成员 byte 拷贝进缓冲区，最后保存。读取时也需要这样的联合体结构。

选择语句

SELECT * FROM 表名； SELECT 列 1，列 2，….列 n FROM 表名；

无 WHERE 条件，则须对表文件进行遍历，并对每条记录，根据选择属性名组，打印出对应的属性值。

SELECT * FROM 表名 WHERE 条件； SELECT 列 1，列 2，…列 n FROM 表名 WHERE 条件

若无可用的索引，则须对表文件进行遍历，并对每条记录，进行 WHERE 条件匹配，若符合，则打印相应属性的值。若有可用的索引，则可调用 Index Manater 模块的功能，

查找符合 WHERE 条件的记录，并打印相应属性的值。在最后打印出被选择的记录的条数。

插入记录语句

INSERT INTO 表名 VALUES (值 1，值 2，…….,值 n)；

根据 Catalog Manager 处理生成的初步内部数据形式，提取表名及记录，根据信息计算插入记录的块号，并调用 Buffer Manager 的功能，获取指定的内存块，并将记录插入到内存中，同时修改脏位及表信息即可。

删除记录语句

DELETE FORM 表名 WHERE 条件；

根据 Catalog Manager 处理生成的内部数据形式，提取表名及可用的索引。若有可用的索引，则须根据索引来查找符合条件的记录，删除该记录并更新该表在数据字典中的信息，循环往复，直至所有符合条件的记录都被删除为止。若无可用的索引，则须对表文件中的所有记录进行一次遍历，并对每一条记录都需要判别是否符合条件。若符合条件，则删除该记录并更新表在数据字典中的信息。在最后打印出被已被删除的记录的条数。

其主要函数的功能描述如下：

1.select all 所调用的函数，不需要索引直接遍历整个表文件，主要根据元组长度定长在缓冲区数组中寻找元组位置。

void Select_All(int N, string Table_Name, int Attr_N, int Attr_Type[], string Attr_Name[])

2.若没有索引可用，则需要对 where 中的条件进行验证处理，verify 就是这样一个函数，它可以根据 where 中的条件对每一个键值进行验证。

int verify(string temp, string Attr[], int op[], string Num[], int num, int Attr_Type[], string Attr_Name[], int n)

3.select 里判断有没有条件是对有索引的文件进行约束的，有的话执行 show 函数，没有的话执行 select All 函数

void Select(int N, string Table_Name, int Attr_N, int Attr_Type[], string Attr_Name[], int op_N, string Attr[], int op[], string Num[], string Index[])

4.show 函数针对某一索引得到全部元组，再对这部分元组验证剩下没有索引的属性。

void Show(string Table_Name, int length, int Attr_N, int Attr_Type[], string Attr_Name[], int op_N, string Attr[], int op[], string Num[], int s[], int n)

退出程序

5.对于现有数据 N，找到 N+1 在文件中应该存在的位置，将数据拷贝进去，最后直接回写。

Record

void RecordManager::Insert(int N, string Table_Name, int Attr_N, int Attr_Type[], string Attr_Name[], string value[])

6.delete all 删除全部内容

void Delete_All(int N, string Table_Name, int Attr_N, int Attr_Type[], string Attr_Name[], string Index[])

7.有索引则根据索引删除，无则遍历删除

void RecordManager::Delete(int N, string Table_Name, int Attr_N, int Attr_Type[], string Attr_Name[], int op_N, string Attr[], int op[], string Num[], string Index[])

8.返回某表某列的全部数据，建立索引时 index manager 调用。

Pair* Get_Attr(int N, string Table_Name, string Attr, int Attr_N, int Attr_Type[], string Attr_Name[])

8.API 验证插入是否 unique 的时候调用验证，返回 1 表示在表中已存在，0 则不存在。

int ExistValue(int N, string Table_Name, int Attr_N, int Attr_Type[], string Attr_Name[], string value[], int WhichOne)

通过 Record Manager 模块的功能，最终实现了对记录的插入，删除和选择的 SQL 语句。但实现该功能，则是在 Interpreter 模块，API 模块和 Catalog Manager 模块的功能之上，在调用 Buffer Manager 模块和 Index Manager 模块的功能，才得以实现。由此可见，这三条 SQL 语句通贯了所有模块的功能，是最能体现模块之间协调工作的测试用例。

Record