# MiniSQL报告

计算机科学 徐延成 3160101851

1. **功能概述**

miniSQL的整体设计要求数据库有建立，删除表和索引，插入、删除、搜索数据。表中支持int，float和char三种数据类型，表的属性可以定义为其中任意的数据类型，同时支持将属性设定为unique或是表的主键。对于定义为unique的属性可以生成B+树索引。

1. **程序分块**

程序分为表，索引，数据库管理和命令解析几个方面。

表类在其中存储了自身属性的具体信息（名称，数据类型，是否unique等）和内容数据，并提供了读写文件，和对表内部数据搜索，以及表输出的方法。

索引内部有指向表的指针，为其中的unique的属性建立b+树索引，和通过索引寻找数据的方法。通过b+树可以快速找到相应的数据，并打包返回。

数据库管理系统部分综合了表和索引两部分，提供了两部分沟通的功能。同时他也提供了整个数据库的API，可以通过它提供的函数来操作整个数据库。

命令解析可以解析输入的SQL命令，并把它转化为API可以识别的形式

1. **API设计**
2. 创建索引

dbCreateIndex(string indexName, string tableName, int attrNo);

2） 创建表

dbCreateTable(string tableattr);

3） 相等值查找

dbSearch(string tableName, int attrNo, string Value);

4） 不等查找

dbSearch(string tableName, int attrNo, float lowerValue, float higherValue, bool equal1, bool equal2);

5） 删除表

dbDropTable(string tableName);

6） 删除索引

dbDropIndex(string tableName, int attrNo);

dbDropIndex(string indexName);

1. **关键算法**
2. B+树插入
3. BPlusNode\* BPlusNode::recursiveInsert(BPlusNode\* T, KeyType keyValue, int tPos, BPlusNode\* Parent) {
4. while (j < T->keyNum && keyValue >= T->key[j]) {
5. if (是重复值)
6. return T;
7. j++;
8. }
9. if (j != 0 && T->children[0] != NULL)
10. j--;
11. if (T是叶节点) {
12. T中插入Key
13. if (T有父节点)
14. Parent->key[tPos] = T->key[0];
15. }
16. else
17. T->children[j] = recursiveInsert(T->children[j], keyValue, j, T);
18. Limit = M;
19. if (T中元素数目超过M) {//如果需要分裂节点
20. if (T没有父节点)
21. 分裂T节点；
22. else {
23. Sibling = T相邻的儿子没有满的节点；
24. if (Sibling != NULL) {
25. 将T一个元素移动到Sibling中；
26. }
27. else {
28. 将T分裂为两个节点；
29. }
30. }
31. }
32. if (T有父节点)
33. Parent->key[tPos] = T->key[0];
34. return T;
35. }

2）Table插入

1. void Table::tableInsert(string toAdd) {
2. for (int i = 0; i < MAX\_ATTR\_NUM; i++) {
3. if (attrName[i] == "") break;
4. if (unique[i] == true) {
5. 检查待插入的记录中所有的属性是否和表中unique属性没有重复
6. if (没有重复)
7. 将记录插入；
8. }
9. }
10. BufferNode\* focus = next;
11. if (buffer无内容) {
12. 创建新的bufferNode；
13. }
14. while (focus != NULL && focus->next != NULL) {
15. if (这个BufferNode没有满)
16. break;
17. focus = focus->next;
18. }
19. if (focus != NULL && focus->recordNum < BUFFER\_CAPACITY ) {
20. 将记录查到bufferNode的最后；
21. focus->recordNum++;
22. focus->length += toAdd.length();
23. }
24. else if (没有可以插入的位置) {
25. 创建新的节点并插入；
26. }
27. return;
28. }

3） Table搜索

1. Table Table::searchTable(int attrNo, double lower, double higher, bool equal1, bool equal2) {
2. 对应this table创建一张新的表
3. BufferNode\* focus = head.next;
4. while (focus != NULL) {
5. for (i = 0; i < focus中的记录数; i++) {
6. attrvalue = focus中的第i条记录中的相应属性；
7. 根据equal1，equal2的值确定是否符合条件
8. if(符合条件)
9. 将这一条记录插入result；
10. }
11. focus = focus->next;
12. }
13. return result;
14. }

4） Table选择

1. Table Table::selectTable(vector<int> attrNo) {
2. 根据选择的属性创建新的表result；
3. for (i = 0; i < 原表的总记录数; i++) {
4. newcontent = 提取原表中第i个记录中的需要属性；
5. 将newcontent插入result中；
6. }
7. return result
8. }