浙江大學
ZHEJIANG UNIVERSITY

Undergraduate Project Report

Mini SQL Design

	Course	Database System		
	Academic Advisor	Professor Bo Zhou		
	Teaching Assistants	Qilian Li		
	Member	Leming Shen	Ruixiao Lin	Xieshi Zheng
	ID Number	3180103654	3180105874	3180103810
	College	College of Computer Science and Technology		
	Major	Software Engineering		

29 June, 2020

ABSTRACT

In 1970, IBM researcher, is regarded as the "father" of the Relational database of Edgar Frank cape (Edgar Frank Codd or e. f. Codd) Dr In journals published "the Communication of the ACM was entitled" A Relational Model of Data for Large Shared Data Banks (Large Shared database Relational Model) "papers, this paper puts forward the concept of relation database Model for the first time. It lays the theoretical foundation of the relational model. At the end of 1970s, great achievements were made in the theoretical research of relational methods and the development of software systems. The relational database experimental System R developed by IBM San Jose laboratory on the IBM370 series computer lasted for 6 years and achieved success. In 1981, IBM announced SQL/DS, a new database product with all the features of System R. Because the relational model is simple and clear, and has a solid mathematical theory foundation, it has been highly valued and widely responded to by the academic circle and the industry, and has quickly become the mainstream of the database market. Since the 1980s, almost all database management systems introduced by computer manufacturers support relational model, and most of the current research work in the field of database is based on relational model. Therefore, our team developed a set of small database system based on Microsoft Visual C# platform to simulate the internal storage of various queries and data.

摘要

1970 年，IBM 的研究员，有“关系数据库之父”之称的埃德加•弗兰克•科德（Edgar Frank Codd 或 E. F. Codd）博士在刊物《Communication of the ACM》上发表了题为“A Relational Model of Data for Large Shared Data banks（大型共享数据库的关系模型）”的论文，文中首次提出了数据库的关系模型的概念，奠定了关系模型的理论基础。20 世纪 70 年代末，关系方法的理论研究和软件系统的研制均取得了很大成果，IBM 公司的 San Jose 实验室在 IBM370 系列机上研制的关系数据库实验系统 System R 历时 6 年获得成功。1981 年 IBM 公司又宣布了具有 System R 全部特征的新的数据库产品 SQL/DS 问世。由于关系模型简单明了、具有坚实的数学理论基础，所以一经推出就受到了学术界和产业界的高度重视和广泛响应，并很快成为数据库市场的主流。20 世纪 80 年代以来，计算机厂商推出的数据库管理系统几乎都支持关系模型，数据库领域当前的研究工作大都以关系模型为基础。因此我们小组在 Microsoft Visual C#平台的基础上开发了一套小型的数据库系统以模拟各种查询及数据的内部存储。

CONTENTS

1.简介

1.1. 目标

设计并实现一个精简型单用户 SQL 引擎(DBMS) Mini SQL，允许用户通过字符界面输入 SQL 语句实现表的建立/删除；索引的建立/删除以及表记录的插入/删除/查找。通过对 Mini SQL 的设计与实现，提高我们的系统编程能力，加深对数据库系统原理的理解。能够较熟练运用 MySQL 和 SQLServer 进行简单数据库的建立。对数据库的储存方式和操作机理有较清晰的理解。能熟练运用 C#语言进行编程，并希望利用 C#对已理解的 SQL 操作和储存方式进行复现和优化。

1.2. 背景

本项目由沈乐明、林瑞潇、郑燮石受数据库老师周波教授任命，以课堂大程的形式来独立完成一个 MiniSQL 的小型系统。

本组开发人员具有数据库应用与实现的知识，并且有较好的 C++/C#基础，而且有良好的团队精神。

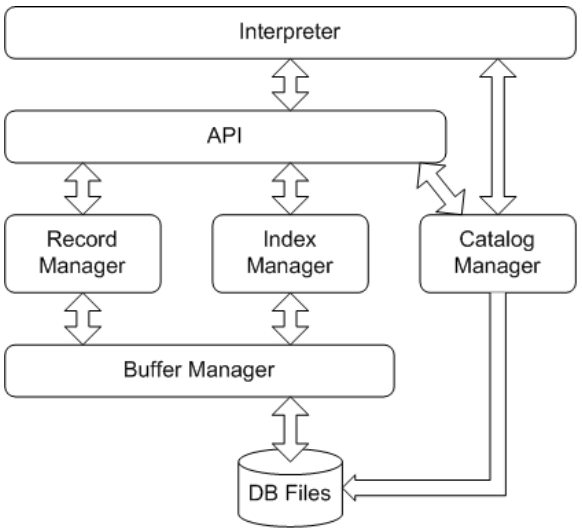
1.3. 参考资料

1.3.1. Microsoft Visual C# Step by Step, 9th Edition.

1.3.2. Database System Concepts, 6th Edition.

2.系统架构

2.1. 模块



2.2. 程序文件清单

实现功能	程序文件名	语言	描述
主页导航及导入数据库文件	MainPage.xaml	Xaml	主页的视图层
	MainPage.xaml.cs	C#	导航栏响应及数据库文件导入响应
数据表插入、删除	Table_Operations.xaml	Xaml	为数据表插入与删除提供可视化界面
	Table_Operations.xaml.cs	C#	实现数据表的插入删除和导出数据库信息文件
索引的插入、删除	Index_Operations.xaml	Xaml	为索引的插入与删除提供可视化界面
	Index_Operations.xaml.cs	C#	实现索引的插入删除
表内记录的插入	Insert_Operations.xaml	Xaml	为表内记录的插入提供可视化界面
	Insert_Operations.xaml.cs	C#	实现表内记录的插入
表内记录的删除	Delete_Operations.xaml	Xaml	为表内记录的删除提供可视化界面
	Delete_Operations.xaml.cs	C#	实现表内记录的删除
查询语句的实现	Querys.xaml	Xaml	为查询语句的实现提供可视化的界面
	Querys.xaml.cs	C#	实现查询并显示结果
B+树类的实现	BPlusTree.cs	C#	实现 B+树类
数据表类的实现	Table.cs	C#	实现数据表类
字段类的实现	Attribute.cs	C#	实现字段类

3.模块设计

3.1. fileManager 模块

3.1.1.模块描述

该模块主要实现的是文件的管理

3.1.2.主要功能

除了基本的控制文件流的开关以外, 还可以从文件中读取一个数据块或者将一个数据块写回文件。

3.1.3.具体实现

readIn 刚开始记录文件名和文件属性，防止被多次打开。定义一个文件指针用于打开文件流。用 fseek 函数从一个文件中找到这个块的具体位置后，通过 fread 函数从指针 fin 位置开始，将块大小为 BLOCK_SIZE 大小的数据块读一次到 blockNode 类型的数组 data 中 Writeback 和 readIn 相反，但过程中需要把已在缓冲区的数据通过 flush 函数写入文件里

3.2. bufferManager 模块

3.2.1.模块描述

负责管理需要把哪些数据块放入文件中，主要采用了 LRU 算法

3.2.2.主要功能

通过 getStrFromNode(blockNode &nd, int begin, int end)，setStrInNode(blockNode &nd, int begin, int end, string str) 和 setDataPackInNode(blockNode &nd, int begin, int len, char *data)函数实现从数据块中获取字符串，将字符串写入数据块和将数据打包写入数据块中。而模板中则实现了 int、float、double 不同的类型在数据块中的某一位置获取数据值。

3.2.3.接口描述

和 table 模块有接口，table 部分不断地使用此模块类中的函数管理其自身的数据。

3.2.4.具体实现

从数据块中获取字符串需要给定起始位置和终止位置，然后依次读取其中的数据。向数据块的某一位置写入字符串则相反，如果给出的长度大于需要写入的字符串的长度，则在最后补 0。当向数据块中某一位置写入数据包和向数据块中某一位置写入字符串类似。

popBlock()，pushBlock()、loadBlock()和 refreshBlock()都是针对数据块的操作，分别是将一个已在缓冲区中的数据块弹出，外部输入的数据块压入缓冲区，从文件中拿出一个块放入缓冲区准备加载和更新缓冲区中数据块的时间戳。

3.3. Catalog Manager 模块

3.3.1.模块描述

Catalog Manager 模块负责管理数据库中表的模式信息，是数据字典管理器，其主要功能是在建立表、删除表、创建或删除某个属性上的索引时，通过文件的形式读取或写入表的模式信息。

3.3.2.接口概述

主要存储表和索引的信息，供其他模块的操作读取。表、索引和数据字典信息放在不同的文件里。CatalogManager 向 API 提供若干操作的基本函数，如 reateTable, dropTable 等，同时向逻辑层的 IndexManager 和 RecordManager 提供 getTable, getAttrByIndex 等函数用于从数据字典加载表和索引的信息。

3.3.3.具体实现

- createTable 新建表，向数据字典写入表名和属性。记录的每个属性，都有名字、类型、是否允许空值、是否为主键、是否唯一和索引名称。
- getTable 在已有表中检查是否存在表，如果存在则返回一个 table 类对象（详见 table 类说明）。否则返回一个表名为*NULL*的表。
- dropTable 读取数据字典，并从中删除一个表的所有信息。
- getAttrByIndex 通过索引名，找到相应的表名和属性名。
- giveAttrIndex 向指定表的指定属性添加索引，并将此信息记入数据字典。
- discardAttrIndex 在数据字典中清除一个索引，（实现方式是用*NULL* 代替索引名）。

3.4. interpreter 模块

3.4.1.模块描述

Interpreter 模块负责 minisql 中外部输入的解释和语法分析，抓取输入语句中的关键词，并将结果传给相应的模块。

3.4.2.主要功能

每次读入一行 sql 语句，利用函数将 sql 语句解释成一组词，每次用一个函数判断是否已经读完，读完就进入下一行的读入。

解释 sql 语句时，如果在抓取词语过程中发现有语法错误，会显示错误信息。在最后的会显示所执行语句过程的耗时和其他提示信息，比如错误提示。

Interpreter Definition:

```
1.  class interpreter
2.  {
3.  private:
4.      selector sr;
5.      void printRecord(map<string, vector<string>> &res,
6.          vector<string> & Attrs, vector<int> & width, int x);  //每一行记录
7.      void printLine(vector<int> & width);  //打印每行边框
8.
9.  public:
10.     interpreter();
11.     ~interpreter();
12.     void flush();  //把所有内存中的东西写到文件里
13.     map<string,vector<string>> execute(string cmd, int printOrNot=-1); //属性及对应数据集
14.     void printAll(map<string, vector<string>> res); //打印所有信息
15. };
```

Spliter Definition:

```
class Spliter
{
    string str;
    int cur;
public:
    Spliter(string S);
    ~Spliter();
    string getWord();  //得到一个词语
    string getBrackets();  //得到一个括号里的字符串
    bool getSemicolon();  //能否找到一个封号，如果能找到返回 true
    string getOp();  //得到逻辑操作符
    string getUntilSemicolon();  //得到封号之前的字符串
    string getUntilWord();  //用于分隔单词
    bool end();  //判断是否已经到达终点
};
vector<string> splitComma(string S);
```

3.4.3.接口概述

Interpreter 模块部分只有主程序调用，负责处理用户的输入和输出。

用户使用时，调用 execute 函数。两个参数分别表示命令字符串和是否将结果输出到标准输出流（1 表示输出，0 表示不输出，-1 表示默认；默认情况下只有 select 的结果和错误信息或输出）。

Execute 函数返回一个类型为 map<string,vector<string>>的值。Map 的第一维 string 表示属性名，第二维的 vector 表示该属性的值。系统自动生成的状态属性名以#开头，例如#Error 表示错误信息，#time 表示运行时间，#Status 表示操作是否成功等。Select 得到的数据按照属性名按列返回。

3.4.4.具体实现

根据 sql 语句，首先读入操作，根据操作再分不同的情形。其中 Spliter 类中有负责读取的函数，getWord()函数从字符串中找到一个词语，vector<string> splitComma(string S)从一组带着逗号的字符串中将属性名和属性类型分开存入一个类型为 string 的 vector 容器中。

当读入的操作是 create 时，读取下一个词，此时可能是 table 或 index。如果是 table，取词看是否有 table name，如果有则得到括号里的字符串，然后用 splitComma 处理得到相应的属性名和属性值。最后处理像 private、key 这样的附加字句，在处理属性类型时，相对应的情况如下：

```
#define ATTR_TYPE_INT 0
#define ATTR_TYPE_FLOAT -1
#define ATTR_TYPE_DOUBLE -2
#define ATTR_TYPE_UNDEF -3
```

另外 char 类型的值均大于 0，不同的整数值代表不同的长度。在处理像 notnull，unique，primary 等属性的相关信息时，只需要将相应的属性值变成

true 就行。过程中还需要检查属性名是否和主键，其他属性名重合，如果有报错，最后寻找是否能找到一个封号表示该 sql 语句结束。

相比较 table，index 更为简单。只需要用 getWord 抓取第一个词，如果不是 on 就报错。否则继续取词和得到括号内的字符串。考虑是否唯一，如果唯一就可以建立索引值当读入的操作不是 create 而是 insert 时，类似地，检查下一个词是不是 into，然后依次获得表名，判断是不是 values，然后获取括号内的字符串，经 splitComma 函数提取后，依次检查类型是否一致，最后将属性值插入到文件中，最后寻找是否能找到一个封号表示该 sql 语句结束。

当读入的操作是 select 时，同样地，先找到表名和选择关键词是否是 where，然后获取判断逻辑，最后检查是否有一个封号表示 sql 语句结束。

当读入的操作是 delete 时，读入下一个词，如果不是 from 则报错，否则再次抓取表名。随后抓取一个词，判断是否为 where，如果是则执行类似 select 语句中的操作，寻找得出判断逻辑。

当读入的操作是 drop 时，读取下一个词，如果是 table，就直接拆分词，寻找封号判断 sql 语句是否已经结束；如果是 index 同理。当读入的操作是 execfile 时，判断文件路径是否正确，然后按字符串分别存值。

3.5. attribute

3.5.1.功能

记录一条属性的信息，包括名称、类型、索引、是否主键、是否非空、是否唯一。

3.5.2.实现细节

Attribute 类记录了表中一列的名字、类型和基本约束（包括是否主键、是否非空和是否唯一）。其中，类型用一个整数表示，0 表示 int，-1 表示 float，-2 表示 double，正数表示字符串类型的长度。在代码中，用类型代表的数字被定义成宏 ATTR_TYPE_INT, ATTR_TYPE_FLOAT, ATTR_TYPE_DOUBLE。

3.6. condition

3.6.1.功能

在 select 和 delete 语句中使用的 where 表达将转化成为一个 condition 类，以便操作。

3.6.2.实现细节

condition 实际上是类型 pair<vector<valueComparison>, vector<logicalComparison>>的缩写。其中 valueComparison 包含三个值, val1, val2，op，用于表示 val1 与 val2 之间通过符号 op 进行比较。而 logicalComparison 仅包含一个字符串，表示操作符。假如一个 condition 的值被

表示为：

```
( { { A,1,> } , { B,2,< } , { C,3,== }  } ,
   {  {"("} , {"O"} , {")A"} , {""}  } )
```

则源本的表达式可能是：

```
select * from tb where (A>1 or B<2) and C==3;
```

另外，between 表达式会被拆成两个比较表达

```
( {  { attr,1,>= } , { attr,10,<= }  } ,
  { {""}, {"A"}, {""} } )
```

```
select * from tb where attr between 1 and 10;
```

注意到：logicalComparison 的长度总比 valueComparison 的长度多 1，除非没有选择条件（此时两个 vector 都为空）。

3.7. conditionChecker

3.7.1.功能

用于检测一条记录是否符合 condition 条件的功能类。

3.7.2.实现细节

首先，对于 condition 的每一个 valueComparison 系统会检测 val1 和 val2 的类型：如果是表内的属性名，则替换成记录中的值，否则用正则表达式判断数据类型；在本组的设计中，所有字符类型的值总有单引号包围，例如'name123' 会被认为是字符串，而 name123 会被当做属性名处理，如果找不到这样的属性，则会报错；没有小数点的数字被认为是 int，有小数点的数字被认为是 double；系统不支持字符串和任何类型的数字比较，但允许按字典序比较字符串。
正则表达式 ：

```
stirng: regex("\'(\\s|\\S)*\'")
double: regex("[+-]?\\d+\\.\\d+")
int:    regex("[+-]?\\d+")
```

在处理完所有 valueComparison 的比较后，会把每个比较替换为一个 0/1 值，并与 logicalComparison 进行拼接。拼接后的字符串形如："(1O0)A1"。

这样的字符串会交给一个叫做 evaluate 的函数按照"括号 > 与 > 或"的优先级计算整个布尔表达式的值。

3.8. indexManager

3.8.1.功能

索引管理器，直接操作索引文件（由于用例中使用的数据较少，系统足以将索

引文件全部冲入主存，所以没有使用分块读取，使用 STL 红黑树实现顺序索引）。

3.8.2.实现细节

索引管理器以“val:blockNumber,offset”的格式保存在一个.index 文件中。由于索引文件较小，所以我们选择将整个 index 读进主存，进行处理。索引在内存中使用红黑树存储，可以实现 O(logN) 的点查询和区间查询。对于 float 和 double 类型，如果直接输出到文件可能造成严重的精度误差。所以我们使用 *(int*)&x 的方法将 float 强制转换为 int，存入文件，而 double 类型是 64 位数据，则强制转换为 long long 类型。

除非 drop index 或 exit，使用过的 index 都不会从内存中消失。尽管可以使用函数 writeToFile 将 index 强制保存到文件，但这并不意味着将 index 从内存中清除。

值得注意的是，对于小数的点查询，为了避免精度误差导致无法找到记录，我们会使用 [x EP S, x + EP S) 上的区间查询代替点查询，其中 EP S = 1e 6。

3.9. recordManager

3.9.1.功能

记录管理器，通过 table 类管理数据文件。

3.9.2.实现细节

记录管理器同时负责插入、删除和选择记录。事实上，插入和删除都需要选择记录，因为插入需要检查主键约束和 unique 键约束。表在文件中以两个链表表示，一个是记录的链表，一个是被删除的位置的链表，以用于将被删除的记录的位置用于新记录的插入。

在插入记录时，记录管理器使用 creteRecord 函数生成一个 char 数组来表示一条记录在二进制文件中的表达。然后传给 insetIntoBlock 函数插入文件，并维护链表形态。

在选择时，如果是对于同一个属性的单一比较或者区间比较并且属性上有索引，则会使用索引；如果没有条件，则直接选取所有记录；否则遍历所有记录检查条件。

注意：虽然我们支持在表中放置空值，但我们不支持选择 UNDEFINED 值，因为比较时 UNDEFINED 值会按照 0 和空字符串进行计算。

3.10. selector

API 类，提供直接且抽象的 Query 函数。

3.11. table

3.11.1. 功能

表，存储从数据字典加载的数据库模式，是各个类之间传递数据库模式的介质。同时作为连接逻辑层和物理层的交互纽带，管理数据文件结构，从缓冲区读取数据。

3.11.2. 11.2 实现细节

table 类记录了表的名字，文件名，属性，记录字节数，表头长度，每个属性在记录中的 offset，并提供了一个通过属性名查找属性的方括号运算符。每条属性默认包含 16+ 属性个数的额外数据，用五个内置属性表示，分别是 #nullList, #preBlock, #preOffset, #nextBlock, #nextOffset （# 开头的属性表示系统属性，在 interpreter 的返回值中也是如此），分别表示空值表，前一条记录的块，前一条记录的偏移，下一条记录的块，下一条记录的偏移。这些数据共同维护表中的两个双向链表。

4.异常处理

我们主要针对输入值类型不匹配、主键对应的值冲突、重复删除或重复插入等异常进行了判定和处理。

```
if ((Application.Current as App).tables[i].get_name() == table_name.Text)
{
    MessageDialog msg = new MessageDialog("You have already created this table!");
    _ = msg.ShowAsync();
    return;
}
```

```
if (flag)
{
    MessageDialog msg = new MessageDialog("Can not find the table.");
    _ = msg.ShowAsync();
    return;
}
```

```
if ((Application.Current as App).tables[i].get_name() == tablename)
{
    MessageDialog msg = new MessageDialog("You have already created this table!");
    _ = msg.ShowAsync();
    return;
}
```

```
if (type[i].IndexOf("int") == -1 && type[i].IndexOf("float") == -1 && type[i].IndexOf("char") == -1)
{
    MessageDialog msg = new MessageDialog("Wrong type!");
    _ = msg.ShowAsync();
    return;
}
```

```
if (name.Count != type.Count)
{
    MessageDialog msg = new MessageDialog("Unmatched attribute name and type");
    _ = msg.ShowAsync();
    return;
}
```

```
catch (ArgumentException)
{
    MessageDialog msg = new MessageDialog("Check your SQL code carefully!");
    _ = msg.ShowAsync();
}
catch (NullReferenceException)
{
    MessageDialog msg = new MessageDialog("You haven't input any attribute's name or type.");
    _ = msg.ShowAsync();
}
catch (Exception)
{
    MessageDialog msg = new MessageDialog("Something else wrong happened!");
    _ = msg.ShowAsync();
}
```

```
if (k == attributes.Count)
{
    MessageDialog msg = new MessageDialog("There is no condition " + attribute11.Text + " !");
    _ = msg.ShowAsync();
    return;
}
```

```
if (k == selattributes.Count)
{
    MessageDialog msg = new MessageDialog("There is no attribute " + name[i] + " !");
    _ = msg.ShowAsync();
    return;
}
```

```
if (single_values[i].Length - 2 > len)
{
    MessageDialog msge = new MessageDialog("char's length is too large!");
    _ = msge.ShowAsync();
    return;
}
```

```
if (i == (Application.Current as App).tables.Count)
{
    MessageDialog msg = new MessageDialog("There is no this table!");
    _ = msg.ShowAsync();
    return;
}
```

```
else
{
    MessageDialog msg = new MessageDialog("The table does not exist!");
    _ = msg.ShowAsync();
    return;
}
```

5.源代码

MainPage.xaml.cs

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.IO;
4.  using Windows.Storage;
5.  using Windows.Storage.Pickers;
6.  using Windows.UI.Popups;
7.  using Windows.UI.Xaml;
8.  using Windows.UI.Xaml.Controls;
9.  using Windows.UI.Xaml.Input;
10. using Windows.Storage.Streams;
11.
12. // The Blank Page item template is documented at https://go.microsoft.com/fw
    link/?LinkId=402352&clcid=0x409
13.
14. namespace MiniSQL
15. {
16.     /// <summary>
17.     /// An empty page that can be used on its own or navigated to within a F
    rame.
18.     /// </summary>
19.     public sealed partial class MainPage : Page
20.     {
21.         public MainPage()
22.         {
23.             this.InitializeComponent();
24.         }
25.
26.         private void main(object sender, TappedRoutedEventArgs e)
27.         {
28.             Frame.Navigate(typeof(MainPage), "");
29.         }
30.
31.         private void query(object sender, TappedRoutedEventArgs e)
32.         {
33.             Frame.Navigate(typeof(Querys), "");
34.         }
35.
36.         private void exit(object sender, TappedRoutedEventArgs e)
37.         {
```

```
38.             (Application.Current as App).Exit();
39.         }
40.
41.         private void table(object sender, TappedRoutedEventArgs e)
42.         {
43.             Frame.Navigate(typeof(Table_Operations), "");
44.         }
45.
46.         private void index(object sender, TappedRoutedEventArgs e)
47.         {
48.             Frame.Navigate(typeof(Index_Operations), "");
49.         }
50.
51.         private void insert(object sender, TappedRoutedEventArgs e)
52.         {
53.             Frame.Navigate(typeof(Insert_Operations), "");
54.         }
55.
56.         private void delete(object sender, TappedRoutedEventArgs e)
57.         {
58.             Frame.Navigate(typeof(Delete_Operations), "");
59.         }
60.
61.         private async void pick(object sender, RoutedEventArgs e)
62.         {
63.             (Application.Current as App).tables.Clear();
64.
65.             FileOpenPicker fp = new FileOpenPicker();
66.             fp.SuggestedStartLocation = PickerLocationId.DocumentsLibrary;
67.             fp.ViewMode = PickerViewMode.List;
68.             fp.FileTypeFilter.Add("*");
69.
70.             StorageFile file = await fp.PickSingleFileAsync();
71.             if (file != null)
72.             {
73.                 var fileStream = await file.OpenAsync(FileAccessMode.Read);

74.                 var inputStream = fileStream.GetInputStreamAt(0);
75.                 TextReader reader = new StreamReader(inputStream.AsStreamFor
    Read());
76.                 List<string> table = new List<string>(reader.ReadToEnd().Spl
    it("\n\n"));
77.
78.                 if (table.Count == 0)
```

```
79.                 {
80.                     MessageDialog msg = new MessageDialog("The content is nu
    ll");
81.                     _ = msg.ShowAsync();
82.                     return;
83.                 }
84.
85.                 Table temp_table = new Table();
86.
87.                 for (int i = 0; i < table.Count; i++)
88.                 {
89.                     string tablename = "";
90.                     List<string> detail = new List<string>(table[i].Split('\
    n'));
91.
92.                     for (int j = 0; j < detail.Count; j++)
93.                     {
94.                         if (j == 0)
95.                         {
96.                             tablename = detail[j];
97.                             temp_table.set_name(tablename);
98.                         }
99.                         else if (detail[j].IndexOf("True") != -
    1 || detail[j].IndexOf("False") != -1)
100.                         {
101.                             string name = detail[j].Substring(0, detail[j].
    IndexOf(' '));
102.                             string type = detail[j].Substring(detail[j].Ind
    exOf(' ') + 1, detail[j].LastIndexOf(' ') - detail[j].IndexOf(' ') - 1);
103.                             bool primary = bool.Parse(detail[j].Substring(d
    etail[j].LastIndexOf(' ') + 1, detail[j].Length - detail[j].LastIndexOf(' ')
     - 1));
104.                             Attribute temp_attribute = new Attribute(name,
    type, primary);
105.                             temp_table.add_attr(temp_attribute);
106.                         }
107.                         else
108.                         {
109.                             if (detail[j].IndexOf(' ') == -1)
110.                             {
111.                                 temp_table.set_record_number(int.Parse(deta
    il[j]));
112.                             }
113.                             else
```

```
114.                                {
115.                                    List<string> values = new List<string>(deta
    il[j].Split(' '));
116.                                    for (int k = 0; k < temp_table.get_attribut
    e().Count; k++)
117.                                    {
118.                                        temp_table.get_attribute()[k].insert_va
    lue(values[k]);
119.                                    }
120.                                }
121.                            }
122.                        }
123.
124.                        (Application.Current as App).tables.Add(temp_table);
125.                        temp_table = new Table();
126.                    }
127.                }
128.                else
129.                {
130.                    MessageDialog msg = new MessageDialog("The file is null or
    you haven't chosen a file.");
131.                    _ = msg.ShowAsync();
132.                }
133.            }
134.
135.            private async void writedata(object sender, RoutedEventArgs e)
136.            {
137.                try
138.                {
139.                    string data = "";
140.
141.                    for (int i = 0; i < (Application.Current as App).tables.Cou
    nt; i++)
142.                    {
143.                        Table element = (Application.Current as App).tables[i];

144.                        data += element.get_name() + "\n";
145.                        foreach (Attribute temp in element.get_attribute())
146.                        {
147.                            data += temp.get_name() + " " + temp.get_type() + "
     " + temp.get_primary().ToString();
148.                            data += "\n";
149.                        }
150.
```

```
151.                     data += (Application.Current as App).tables[i].get_reco
    rd_number().ToString() + ((Application.Current as App).tables[i].get_record_
    number() == 0 ? "" : "\n");
152.
153.                     for (int j = 0; j < element.get_record_number(); j++)
154.                     {
155.                         for (int k = 0; k < element.get_attribute().Count;
    k++)
156.                         {
157.                             data += element.get_attribute()[k].get_values()
    [j] + (k == element.get_attribute().Count - 1 ? "" : " ");
158.                         }
159.                         data += ((i == (Application.Current as App).tables.
    Count - 1 && j == element.get_record_number() - 1) ? "" : "\n");
160.                     }
161.
162.                     data += (i == (Application.Current as App).tables.Count
     - 1 ? "" : "\n");
163.                 }
164.
165.
166.                 FileSavePicker fp = new FileSavePicker();
167.                 var filedb = new[] { ".txt" };
168.                 fp.FileTypeChoices.Add("Plain Text", new List<string>() { "
    .txt" });
169.                 fp.SuggestedFileName = "table information" + DateTime.Now.D
    ay + "-" + DateTime.Now.Month + "-" + DateTime.Now.Year;
170.                 StorageFile sf = await fp.PickSaveFileAsync();
171.
172.                 if (sf != null)
173.                 {
174.                     using (StorageStreamTransaction transaction = await sf.
    OpenTransactedWriteAsync())
175.                     {
176.                         using (DataWriter dataWriter = new DataWriter(trans
    action.Stream))
177.                         {
178.                             dataWriter.WriteString(data);
179.                             transaction.Stream.Size = await dataWriter.Stor
    eAsync();
180.                             await transaction.CommitAsync();
181.                         }
182.                     }
183.                 }
```

```
184.
185.                 MessageDialog msg = new MessageDialog("Operation succeed!")
    ;
186.                 _ = msg.ShowAsync();
187.             }
188.             catch (Exception)
189.             {
190.                 MessageDialog msg = new MessageDialog("Something wrong happ
    ened.");
191.                 _ = msg.ShowAsync();
192.             }
193.
194.         }
195.     }
196. }
```

MainPage.xaml

```
1.  <Page
2.      x:Class="MiniSQL.MainPage"
3.      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4.      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5.      xmlns:local="using:MiniSQL"
6.      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7.      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8.      mc:Ignorable="d"
9.      Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
10.
11.     <Grid>
12.         <NavigationView x:Name="Navigation" IsBackEnabled="True" IsSettingsV
    isible="True" IsTabStop="False">
13.             <NavigationView.MenuItems>
14.                 <NavigationViewItemHeader Content="Functions"/>
15.                 <NavigationViewItem Icon="Home" Content="Main Page" Tapped="
    main"/>
16.                 <NavigationViewItemSeparator/>
17.
18.                 <NavigationViewItem Icon="Import" Content="Table Operations"
     Tapped="table"></NavigationViewItem>
19.                 <NavigationViewItem Icon="List" Content="Index Operations" T
    apped="index"/>
20.                 <NavigationViewItem Icon="Find" Content="Query" Tapped="quer
    y"></NavigationViewItem>
21.                 <NavigationViewItem Icon="AllApps" Content="Insert" Tapped="
    insert"/>
```

```
22.                 <NavigationViewItem Icon="Delete" Content="Delete" Tapped="d
    elete"></NavigationViewItem>
23.                 <NavigationViewItem Icon="Next" Content="Export" Tapped="wri
    tedata"/>
24.                 <NavigationViewItem Icon="Undo" Content="Exit" Tapped="exit"
    />
25.             </NavigationView.MenuItems>
26.
27.             <Grid>
28.                 <Image HorizontalAlignment="Center" VerticalAlignment="Top"
    Source="/Images/zjulib.jpg" Width="843.5" Height="234.5" Margin="0, 50, 0, 0
    "></Image>
29.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Margin="0, 300, 0, 0" FontSize="60">Database System Concepts</TextBlock>

30.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Margin="0, 400, 0, 0" FontSize="50">Course Project --
    - Mini SQL</TextBlock>
31.
32.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Margin="0, 500, 0, 0" FontSize="30">College of Computer Science and Tech
    nology</TextBlock>
33.                 <Button HorizontalAlignment="Center" VerticalAlignment="Top"
     Margin="0, 650, 0, 0" FontSize="30" Height="60" Click="pick">Select Your Da
    tabase File Before Getting Started</Button>
34.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Margin="0, 750, 0, 0" FontSize="30" FontFamily="Consolas">Group Leader:
     Leming Shen    3180103654</TextBlock>
35.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Margin="0, 800, 0, 0" FontSize="30" FontFamily="Consolas">Group Member:
     Ruixiao Lin    3180105874</TextBlock>
36.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Margin="0, 850, 0, 0" FontSize="30" FontFamily="Consolas">Group Member:
     Xieshi Zheng   3180103810</TextBlock>
37.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Margin="0, 920, 0, 0" FontSize="40" FontWeight="Bold" FontFamily="Times
    New Roman" FocusVisualPrimaryBrush="#FFDE3636" Foreground="#FFFFE800">Academ
    ic Advisor:  Professor Bo Zhou</TextBlock>
38.
39.             </Grid>
40.
41.             <NavigationView.PaneFooter>
42.                 <HyperlinkButton x:Name="MoreInfoBtn" Content="Email Me" Mar
    gin="12,0" NavigateUri="mailto:ZJU.SLM@gmail.com"/>
```

```
            </NavigationView.PaneFooter>
        </NavigationView>
    </Grid>
</Page>
```

Attribute.cs

```
using System.Collections.Generic;

namespace MiniSQL
{
    public class Attribute
    {
        private string name;
        private string type;
        private bool is_primary;
        private List<string> values;

        public Attribute()
        {
            this.name = "";
            this.type = "";
            this.is_primary = false;
            this.values = new List<string>();
        }

        public Attribute(string name, string type, bool primary)
        {
            this.name = name;
            this.type = type;
            this.is_primary = primary;
            this.values = new List<string>();
        }

        public string get_name()
        {
            return this.name;
        }

        public string get_type()
        {
            return this.type;
        }

        public bool get_primary()
```

```
        {
            return this.is_primary;
        }

        public List<string> get_values()
        {
            return this.values;
        }

        public void set_name(string name)
        {
            this.name = name;
        }

        public void set_type(string type)
        {
            this.type = type;
        }

        public void set_primary(bool primary)
        {
            this.is_primary = primary;
        }

        public void insert_value(string value)
        {
            this.values.Add(value);
        }

        public void remove_value(int index)
        {
            this.values.RemoveAt(index);
        }
    }
}
```

Table.cs

```
using System.Collections.Generic;
using Windows.UI.Popups;

namespace MiniSQL
{
    public class Table
    {
```

```
8.          private List<Attribute> attributes;
9.          private string name;
10.         private int record_number;
11.
12.         public Table()
13.         {
14.             this.attributes = new List<Attribute>();
15.             this.name = "";
16.             this.record_number = 0;
17.         }
18.
19.         public Table(string name)
20.         {
21.             this.attributes = new List<Attribute>();
22.             this.name = name;
23.             this.record_number = 0;
24.         }
25.
26.         public Table(string name, List<Attribute> attribute)
27.         {
28.             this.name = name;
29.             this.record_number = 0;
30.             this.attributes = attribute;
31.         }
32.
33.         public string get_name()
34.         {
35.             return this.name;
36.         }
37.
38.         public List<Attribute> get_attribute()
39.         {
40.             return this.attributes;
41.         }
42.
43.         public void set_name(string name)
44.         {
45.             this.name = name;
46.         }
47.
48.         public void add_attr(Attribute attribute)
49.         {
50.             this.attributes.Add(attribute);
51.         }
```

```
52.
53.         public bool insert_record(List<string> input)
54.         {
55.             if (input.Count != this.attributes.Count)
56.             {
57.                 MessageDialog msg = new MessageDialog("Lack of values!");
58.                 _ = msg.ShowAsync();
59.                 return false;
60.             }
61.
62.             for (int i = 0; i < input.Count; i++)
63.             {
64.                 if (attributes[i].get_primary() && attributes[i].get_values(
    ).IndexOf(input[i]) != -1)
65.                 {
66.                     MessageDialog msg = new MessageDialog("Primary key distr
    ibuted! Constraints violated!");
67.                     _ = msg.ShowAsync();
68.                     return false;
69.                 }
70.                 else
71.                 {
72.                     attributes[i].insert_value(input[i]);
73.                 }
74.             }
75.
76.             this.record_number++;
77.             return true;
78.         }
79.
80.         public bool remove_record(List<int> index)
81.         {
82.             for (int i = 0; i < index.Count; i++)
83.             {
84.                 for (int j = 0; j < attributes.Count; j++)
85.                 {
86.                     attributes[j].remove_value(index[i]);
87.                 }
88.             }
89.             this.record_number -= index.Count;
90.             return true;
91.         }
92.
93.         public void clear()
```

```
94.         {
95.             this.name = "";
96.             this.attributes.Clear();
97.             this.record_number = 0;
98.         }
99.
100.         public int get_record_number()
101.         {
102.             return this.record_number;
103.         }
104.
105.         public void set_record_number(int num)
106.         {
107.             this.record_number = num;
108.             return;
109.         }
110.     }
111. }
```

BPlusTree.cs

```
1.  using System;
2.  using KEY_TYPE = System.String;
3.
4.  namespace BPlusTree
5.  {
6.      enum NODE_TYPE
7.      {
8.          NODE_TYPE_ROOT = 1,
9.          NODE_TYPE_INTERNAL = 2,
10.         NODE_TYPE_LEAF = 3
11.     }
12.
13.     class mypub
14.     {
15.         public const int NULL = 0;
16.         public const int INVALID = 0;
17.         public const int FLAG_LEFT = 1;
18.         public const int FLAG_RIGHT = 2;
19.         public const int ORDER_V = 2;
20.         public const int MAXNUM_KEY = (ORDER_V * 2);
21.         public const int MAXNUM_POINTER = (MAXNUM_KEY + 1);
22.         public const int MAXNUM_DATA = (ORDER_V * 2);
23.     }
24.
```

```
25.     class CNode
26.     {
27.         public CNode()
28.         {
29.             m_Type = NODE_TYPE.NODE_TYPE_LEAF;
30.             m_Count = 0;
31.             m_pFather = null;
32.         }
33.
34.         public CNode M_pFather
35.         {
36.             get { return m_pFather; }
37.             set { m_pFather = value; }
38.         }
39.
40.         ~CNode()
41.         {
42.             DeleteChildren();
43.         }
44.
45.         public new NODE_TYPE GetType()
46.         {
47.             return m_Type;
48.         }
49.
50.         public void SetType(NODE_TYPE type)
51.         {
52.             m_Type = type;
53.         }
54.
55.         public int GetCount()
56.         {
57.             return m_Count;
58.         }
59.
60.         public void SetCount(int i)
61.         {
62.             m_Count = i;
63.         }
64.
65.         public virtual KEY_TYPE GetElement(int i)
66.         {
67.             return null;
68.         }
```

```
69.
70.         public virtual void SetElement(int i, KEY_TYPE value) { }
71.
72.         public CNode GetFather()
73.         {
74.             return m_pFather;
75.         }
76.
77.         virtual public CNode GetPointer(int i)
78.         {
79.             return null;
80.         }
81.
82.         virtual public void SetPointer(int i, CNode pointer) { }
83.
84.         public void SetFather(CNode father)
85.         {
86.             m_pFather = father;
87.         }
88.
89.         public CNode GetBrother(ref int flag)
90.         {
91.             if (m_pFather == null)
92.                 return null;
93.             CNode pBrother = null;
94.
95.             for (int i = 1; i <= m_pFather.GetCount() + 1; i++)
96.             {
97.                 if (m_pFather.GetPointer(i) == this)
98.                 {
99.                     if (i == m_pFather.GetCount() + 1)
100.                    {
101.                        pBrother = m_pFather.GetPointer(i - 1);
102.                        flag = mypub.FLAG_LEFT;
103.                    }
104.                    else
105.                    {
106.                        pBrother = m_pFather.GetPointer(i + 1);
107.                        flag = mypub.FLAG_RIGHT;
108.                    }
109.                }
110.            }
111.
112.            return pBrother;
```

```
113.         }
114.
115.         public void DeleteChildren()
116.         {
117.             for (int i = 1; i <= GetCount(); i++)
118.             {
119.                 CNode pNode = GetPointer(i);
120.
121.                 if (null != pNode)
122.                 {
123.                     pNode.DeleteChildren();
124.                 }
125.
126.                 pNode = null;
127.             }
128.         }
129.
130.         protected NODE_TYPE m_Type;
131.         protected int m_Count;
132.         protected CNode m_pFather;
133.     }
134.
135.     class CInternalNode : CNode
136.     {
137.         public CInternalNode()
138.         {
139.             m_Type = NODE_TYPE.NODE_TYPE_INTERNAL;
140.             int i = 0;
141.             m_Keys = new KEY_TYPE[mypub.MAXNUM_KEY];
142.             m_Pointers = new CNode[mypub.MAXNUM_POINTER];
143.
144.             for (i = 0; i < mypub.MAXNUM_KEY; i++)
145.                 m_Keys[i] = mypub.INVALID.ToString();
146.             for (i = 0; i < mypub.MAXNUM_POINTER; i++)
147.                 m_Pointers[i] = null;
148.         }
149.
150.         ~CInternalNode()
151.         {
152.             for (int i = 0; i < mypub.MAXNUM_POINTER; i++)
153.                 m_Pointers[i] = null;
154.         }
155.
156.         override public KEY_TYPE GetElement(int i)
```

```
157.         {
158.             if ((i > 0) && (i <= mypub.MAXNUM_KEY))
159.             {
160.                 return m_Keys[i - 1];
161.             }
162.             else
163.             {
164.                 return mypub.INVALID.ToString();
165.             }
166.         }
167.
168.         override public void SetElement(int i, KEY_TYPE key)
169.         {
170.             if ((i > 0) && (i <= mypub.MAXNUM_KEY))
171.             {
172.                 m_Keys[i - 1] = key;
173.             }
174.         }
175.
176.         override public CNode GetPointer(int i)
177.         {
178.             if ((i > 0) && (i <= mypub.MAXNUM_POINTER))
179.             {
180.                 return m_Pointers[i - 1];
181.             }
182.             else
183.             {
184.                 return null;
185.             }
186.         }
187.
188.         override public void SetPointer(int i, CNode pointer)
189.         {
190.             if ((i > 0) && (i <= mypub.MAXNUM_POINTER))
191.             {
192.                 m_Pointers[i - 1] = pointer;
193.             }
194.         }
195.
196.         public bool Insert(KEY_TYPE value, CNode pNode)
197.         {
198.             int i;
199.             if (GetCount() >= mypub.MAXNUM_KEY)
200.                 return false;
```

```
201.             int j = 0;
202.             for (i = 0; (String.Compare(value, m_Keys[i]) > 0) && (i < m_Co
    unt); i++) ;
203.             for (j = m_Count + 1; j > i + 1; j--)
204.             {
205.                 m_Pointers[j] = m_Pointers[j - 1];
206.             }
207.             m_Keys[i] = value;
208.             m_Pointers[i + 1] = pNode;
209.             pNode.SetFather(this);
210.             m_Count++;
211.             return true;
212.         }
213.
214.         public bool Delete(KEY_TYPE key)
215.         {
216.             int i, j, k;
217.             for (i = 0; (String.Compare(key, m_Keys[i]) >= 0) && (i < m_Cou
    nt); i++) ;
218.             for (j = i - 1; j < m_Count - 1; j++)
219.             {
220.                 m_Keys[j] = m_Keys[j + 1];
221.             }
222.             m_Keys[j] = mypub.INVALID.ToString();
223.             for (k = i; k < m_Count; k++)
224.             {
225.                 m_Pointers[k] = m_Pointers[k + 1];
226.             }
227.             m_Pointers[k] = null;
228.             m_Count--;
229.             return true;
230.         }
231.
232.         public KEY_TYPE Split(CInternalNode pNode, KEY_TYPE key)
233.         {
234.             int i = 0, j = 0;
235.
236.             // 如果要插入的键值在第 V 和 V+1 个键值中间，需要翻转一下，因此先处理
    此情况
237.             if ((String.Compare(key, this.GetElement(mypub.ORDER_V)) > 0) &
    & (String.Compare(key, this.GetElement(mypub.ORDER_V + 1)) < 0))
238.             {
239.                 for (i = mypub.ORDER_V + 1; i <= mypub.MAXNUM_KEY; i++)
240.                 {
```

```
241.                     j++;
242.                     pNode.SetElement(j, this.GetElement(i));
243.                     this.SetElement(i, mypub.INVALID.ToString());
244.                 }
245.                 // 把第 V+2 -- 2V+1 个指针移到指定的结点中
246.                 j = 0;
247.                 for (i = mypub.ORDER_V + 2; i <= mypub.MAXNUM_POINTER; i++)

248.                 {
249.                     j++;
250.                     this.GetPointer(i).SetFather(pNode);    // 重新设置子结
    点的父亲
251.                     pNode.SetPointer(j, this.GetPointer(i));
252.                     this.SetPointer(i, null);
253.                 }
254.                 this.SetCount(mypub.ORDER_V);
255.                 pNode.SetCount(mypub.ORDER_V);
256.                 return key;
257.             }
258.             // 以下处理 key 小于第 V 个键值或 key 大于第 V+1 个键值的情况
259.
260.             // 判断是提取第 V 还是 V+1 个键
261.             int position;
262.
263.             if (String.Compare(key, this.GetElement(mypub.ORDER_V)) < 0)
264.             {
265.                 position = mypub.ORDER_V;
266.             }
267.             else
268.             {
269.                 position = mypub.ORDER_V + 1;
270.             }
271.
272.             KEY_TYPE RetKey = this.GetElement(position);
273.             j = 0;
274.
275.             for (i = position + 1; i <= mypub.MAXNUM_KEY; i++)
276.             {
277.                 j++;
278.                 pNode.SetElement(j, this.GetElement(i));
279.                 this.SetElement(i, null);
280.             }
281.
282.             j = 0;
```

```
283.
284.                 for (i = position + 1; i <= mypub.MAXNUM_POINTER; i++)
285.                 {
286.                     j++;
287.                     this.GetPointer(i).SetFather(pNode);    // 重新设置子结点的父
    亲
288.                     pNode.SetPointer(j, this.GetPointer(i));
289.                     this.SetPointer(i, null);
290.                 }
291.
292.                 // 清除提取出的位置
293.                 this.SetElement(position, null);
294.                 this.SetCount(position - 1);
295.                 pNode.SetCount(mypub.MAXNUM_KEY - position);
296.
297.                 return RetKey;
298.             }
299.
300.             public bool Combine(CNode pNode)
301.             {
302.                 if (this.GetCount() + pNode.GetCount() + 1 > mypub.MAXNUM_DATA)
        // 预留一个新键的位置
303.                 {
304.                     return false;
305.                 }
306.
307.                 KEY_TYPE NewKey = pNode.GetPointer(1).GetElement(1);
308.                 m_Keys[m_Count] = NewKey;
309.                 m_Count++;
310.                 m_Pointers[m_Count] = pNode.GetPointer(1);
311.
312.                 for (int i = 1; i <= pNode.GetCount(); i++)
313.                 {
314.                     m_Keys[m_Count] = pNode.GetElement(i);
315.                     m_Count++;
316.                     m_Pointers[m_Count] = pNode.GetPointer(i + 1);
317.                 }
318.
319.                 return true;
320.             }
321.
322.             public bool MoveOneElement(CNode pNode)
323.             {
324.                 if (this.GetCount() >= mypub.MAXNUM_DATA)
```

```
325.             {
326.                 return false;
327.             }
328.
329.             int i, j;
330.             if (String.Compare(pNode.GetElement(1), this.GetElement(1)) < 0
    )
331.             {
332.                 // 先腾出位置
333.                 for (i = m_Count; i > 0; i--)
334.                 {
335.                     m_Keys[i] = m_Keys[i - 1];
336.                 }
337.                 for (j = m_Count + 1; j > 0; j--)
338.                 {
339.                     m_Pointers[j] = m_Pointers[j - 1];
340.                 }
341.
342.                 // 赋值
343.                 // 第一个键值不是兄弟结点的最后一个键值，而是本结点第一个子结点的
    第一个元素的值
344.                 m_Keys[0] = GetPointer(1).GetElement(1);
345.                 // 第一个子结点为兄弟结点的最后一个子结点
346.                 m_Pointers[0] = pNode.GetPointer(pNode.GetCount() + 1);
347.
348.                 // 修改兄弟结点
349.                 pNode.SetElement(pNode.GetCount(), null);
350.                 pNode.SetPointer(pNode.GetCount() + 1, null);
351.             }
352.             else    // 兄弟结点在本结点右边
353.             {
354.                 // 赋值
355.                 // 最后一个键值不是兄弟结点的第一个键值，而是兄弟结点第一个子结点
    的第一个元素的值
356.                 m_Keys[m_Count] = pNode.GetPointer(1).GetElement(1);
357.                 // 最后一个子结点为兄弟结点的第一个子结点
358.                 m_Pointers[m_Count + 1] = pNode.GetPointer(1);
359.
360.                 // 修改兄弟结点
361.                 for (i = 1; i < pNode.GetCount() - 1; i++)
362.                 {
363.                     pNode.SetElement(i, pNode.GetElement(i + 1));
364.                 }
365.                 for (j = 1; j < pNode.GetCount(); j++)
```

```
366.                 {
367.                     pNode.SetPointer(j, pNode.GetPointer(j + 1));
368.                 }
369.             }
370.
371.             this.SetCount(this.GetCount() + 1);
372.             pNode.SetCount(pNode.GetCount() - 1);
373.
374.             return true;
375.         }
376.
377.         protected KEY_TYPE[] m_Keys;
378.         protected CNode[] m_Pointers;
379.     }
380.
381.     class CLeafNode : CNode
382.     {
383.         public CLeafNode()
384.         {
385.             m_Type = NODE_TYPE.NODE_TYPE_LEAF;
386.             m_Datas = new String[mypub.MAXNUM_DATA];
387.             for (int i = 0; i < mypub.MAXNUM_DATA; i++)
388.             {
389.                 m_Datas[i] = "\0";
390.             }
391.             m_pPrevNode = null;
392.             m_pNextNode = null;
393.         }
394.
395.         ~CLeafNode() { }
396.
397.         override public KEY_TYPE GetElement(int i)
398.         {
399.             if ((i > 0) && (i <= mypub.MAXNUM_DATA))
400.             {
401.                 return m_Datas[i - 1];
402.             }
403.             else
404.             {
405.                 return mypub.INVALID.ToString();
406.             }
407.         }
408.
409.         override public void SetElement(int i, KEY_TYPE data)
```

```
410.         {
411.             if ((i > 0) && (i <= mypub.MAXNUM_DATA))
412.             {
413.                 m_Datas[i - 1] = data;
414.             }
415.         }
416.
417.         public bool Insert(KEY_TYPE value)
418.         {
419.             int i, j;
420.             // 如果叶子结点已满，直接返回失败
421.             if (GetCount() >= mypub.MAXNUM_DATA)
422.             {
423.                 return false;
424.             }
425.
426.             // 找到要插入数据的位置
427.             for (i = 0; (String.Compare(value, m_Datas[i]) > 0) && (i < m_C
    ount); i++) ;
428.
429.             // 当前位置及其后面的数据依次后移，空出当前位置
430.             for (j = m_Count; j > i; j--)
431.             {
432.                 m_Datas[j] = m_Datas[j - 1];
433.             }
434.
435.             // 把数据存入当前位置
436.             m_Datas[i] = value;
437.
438.             m_Count++;
439.
440.             // 返回成功
441.             return true;
442.         }
443.
444.         public bool Delete(KEY_TYPE value)
445.         {
446.             int i, j;
447.             bool found = false;
448.             for (i = 0; i < m_Count; i++)
449.             {
450.                 if (value == m_Datas[i])
451.                 {
452.                     found = true;
```

```
453.                     break;
454.                 }
455.             }
456.             // 如果没有找到，返回失败
457.             if (false == found)
458.             {
459.                 return false;
460.             }
461.
462.             // 后面的数据依次前移
463.             for (j = i; j < m_Count - 1; j++)
464.             {
465.                 m_Datas[j] = m_Datas[j + 1];
466.             }
467.
468.             m_Datas[j] = null;
469.             m_Count--;
470.
471.             // 返回成功
472.             return true;
473.         }
474.
475.         public KEY_TYPE Split(CNode pNode)
476.         {
477.             int j = 0;
478.             for (int i = mypub.ORDER_V + 1; i <= mypub.MAXNUM_DATA; i++)
479.             {
480.                 j++;
481.                 pNode.SetElement(j, this.GetElement(i));
482.                 this.SetElement(i, null);
483.             }
484.             // 设置好 Count 个数
485.             this.SetCount(this.GetCount() - j);
486.             pNode.SetCount(pNode.GetCount() + j);
487.
488.             // 返回新结点的第一个元素作为键
489.             return pNode.GetElement(1);
490.         }
491.
492.         public bool Combine(CNode pNode)
493.         {
494.             if (this.GetCount() + pNode.GetCount() > mypub.MAXNUM_DATA)
495.             {
496.                 return false;
```

```
497.             }
498.
499.             for (int i = 1; i <= pNode.GetCount(); i++)
500.             {
501.                 this.Insert(pNode.GetElement(i));
502.             }
503.
504.             return true;
505.         }
506.
507.         public CLeafNode m_pPrevNode;
508.         public CLeafNode m_pNextNode;
509.         protected KEY_TYPE[] m_Datas;
510.     }
511.
512.     class BPlusTree
513.     {
514.         public BPlusTree()
515.         {
516.             m_Depth = 0;
517.             m_Root = null;
518.             m_pLeafHead = null;
519.             m_pLeafTail = null;
520.         }
521.         ~BPlusTree() { }
522.
523.         // 查找指定的数据
524.         public bool Search(KEY_TYPE data)
525.         {
526.             int i = 0;
527.
528.             CNode pNode = GetRoot();
529.             // 循环查找对应的叶子结点
530.             while (null != pNode)
531.             {
532.                 // 结点为叶子结点，循环结束
533.                 if (NODE_TYPE.NODE_TYPE_LEAF == pNode.GetType())
534.                 {
535.                     break;
536.                 }
537.
538.                 // 找到第一个键值大于等于 key 的位置
539.                 for (i = 1; (String.Compare(data, pNode.GetElement(i)) >= 0
    ) && (i <= pNode.GetCount()); i++) ;
```

```
540.
541.                 pNode = pNode.GetPointer(i);
542.             }
543.
544.             // 没找到
545.             if (null == pNode)
546.             {
547.                 return false;
548.             }
549.
550.             // 在叶子结点中继续找
551.             bool found = false;
552.             for (i = 1; (i <= pNode.GetCount()); i++)
553.             {
554.                 if (data == pNode.GetElement(i))
555.                 {
556.                     found = true;
557.                 }
558.             }
559.
560.             return found;
561.         }
562.
563.         // 插入指定的数据
564.         public bool Insert(KEY_TYPE data)
565.         {
566.             // 检查是否重复插入
567.             bool found = Search(data);
568.             if (true == found)
569.             {
570.                 return false;
571.             }
572.             // 查找理想的叶子结点
573.             CLeafNode pOldNode = new CLeafNode();
574.             pOldNode = SearchLeafNode(data);
575.             // 如果没有找到，说明整个树是空的，生成根结点
576.             if (null == pOldNode)
577.             {
578.                 pOldNode = new CLeafNode();
579.                 m_pLeafHead = pOldNode;
580.                 m_pLeafTail = pOldNode;
581.                 SetRoot(pOldNode);
582.             }
583.
```

```
584.            // 叶子结点未满，对应情况 1，直接插入
585.            if (pOldNode.GetCount() < mypub.MAXNUM_DATA)
586.            {
587.                return pOldNode.Insert(data);
588.            }
589.
590.            // 原叶子结点已满，新建叶子结点，并把原结点后一半数据剪切到新结点
591.            CLeafNode pNewNode = new CLeafNode();
592.            KEY_TYPE key = null;
593.            key = pOldNode.Split(pNewNode);
594.
595.            // 在双向链表中插入结点
596.            CLeafNode pOldNext = pOldNode.m_pNextNode;
597.            pOldNode.m_pNextNode = pNewNode;
598.            pNewNode.m_pNextNode = pOldNext;
599.            pNewNode.m_pPrevNode = pOldNode;
600.            if (null == pOldNext)
601.            {
602.                m_pLeafTail = pNewNode;
603.            }
604.            else
605.            {
606.                pOldNext.m_pPrevNode = pNewNode;
607.            }
608.
609.
610.            // 判断是插入到原结点还是新结点中，确保是按数据值排序的
611.            if (String.Compare(data, key) < 0)
612.            {
613.                pOldNode.Insert(data);    // 插入原结点
614.            }
615.            else
616.            {
617.                pNewNode.Insert(data);    // 插入新结点
618.            }
619.
620.            // 父结点
621.            CInternalNode pFather = (CInternalNode)(pOldNode.GetFather());
622.
623.            // 如果原结点是根节点，对应情况 2
624.            if (null == pFather)
625.            {
626.                CNode pNode1 = new CInternalNode();
```

```
627.                    pNode1.SetPointer(1, pOldNode);                         /
    / 指针 1 指向原结点
628.                    pNode1.SetElement(1, key);                              /
    / 设置键
629.                    pNode1.SetPointer(2, pNewNode);                         /
    / 指针 2 指向新结点
630.                    pOldNode.SetFather(pNode1);                             /
    / 指定父结点
631.                    pNewNode.SetFather(pNode1);                             /
    / 指定父结点
632.                    pNode1.SetCount(1);
633.
634.                    SetRoot(pNode1);
    // 指定新的根结点
635.                    return true;
636.                }
637.
638.                // 情况 3 和情况 4 在这里实现
639.                bool ret = InsertInternalNode(pFather, key, pNewNode);
640.                return ret;
641.            }
642.
643.            // 删除指定的数据
644.            public bool Delete(KEY_TYPE data)
645.            {
646.                CLeafNode pOldNode = SearchLeafNode(data);
647.                // 如果没有找到，返回失败
648.                if (null == pOldNode)
649.                {
650.                    return false;
651.                }
652.
653.                // 删除数据，如果失败一定是没有找到，直接返回失败
654.                bool success = pOldNode.Delete(data);
655.                if (false == success)
656.                {
657.                    return false;
658.                }
659.
660.                // 获取父结点
661.                CInternalNode pFather = (CInternalNode)(pOldNode.GetFather());

662.                if (null == pFather)
663.                {
```

```
664.                    // 如果一个数据都没有了，删除根结点(只有根节点可能出现此情况)
665.                    if (0 == pOldNode.GetCount())
666.                    {
667.                        pOldNode = null;
668.                        m_pLeafHead = null;
669.                        m_pLeafTail = null;
670.                        SetRoot(null);
671.                    }
672.
673.                    return true;
674.                }
675.
676.
677.                // 删除后叶子结点填充度仍>=50%，对应情况 1
678.                if (pOldNode.GetCount() >= mypub.ORDER_V)
679.                {
680.                    for (int i = 1; (String.Compare(data, pFather.GetElement(i)
    ) >= 0) && (i <= pFather.GetCount()); i++)
681.                    {
682.                        // 如果删除的是父结点的键值，需要更改该键
683.                        if (pFather.GetElement(i) == data)
684.                        {
685.                            pFather.SetElement(i, pOldNode.GetElement(1));    /
    / 更改为叶子结点新的第一个元素
686.                        }
687.                    }
688.
689.                    return true;
690.                }
691.
692.                // 找到一个最近的兄弟结点(根据 B+树的定义，除了叶子结点，总是能找到
    的)
693.                int flag = mypub.FLAG_LEFT;
694.                CLeafNode pBrother = (CLeafNode)(pOldNode.GetBrother(ref flag))
    ;
695.
696.                // 兄弟结点填充度>50%，对应情况 2A
697.                KEY_TYPE NewData = null;
698.                if (pBrother.GetCount() > mypub.ORDER_V)
699.                {
700.                    if (mypub.FLAG_LEFT == flag)    // 兄弟在左边，移最后一个数据
    过来
701.                    {
702.                        NewData = pBrother.GetElement(pBrother.GetCount());
```

```
703.                 }
704.                 else    // 兄弟在右边，移第一个数据过来
705.                 {
706.                     NewData = pBrother.GetElement(1);
707.                 }
708.
709.                 pOldNode.Insert(NewData);
710.                 pBrother.Delete(NewData);
711.
712.                 // 修改父结点的键值
713.                 if (mypub.FLAG_LEFT == flag)
714.                 {
715.                     for (int i = 1; i <= pFather.GetCount() + 1; i++)
716.                     {
717.                         if (pFather.GetPointer(i) == pOldNode && i > 1)
718.                         {
719.                             pFather.SetElement(i - 1, pOldNode.GetElement(1
    ));    // 更改本结点对应的键
720.                         }
721.                     }
722.                 }
723.                 else
724.                 {
725.                     for (int i = 1; i <= pFather.GetCount() + 1; i++)
726.                     {
727.                         if (pFather.GetPointer(i) == pOldNode && i > 1)
728.                         {
729.                             pFather.SetElement(i - 1, pOldNode.GetElement(1
    ));    // 更改本结点对应的键
730.                         }
731.                         if (pFather.GetPointer(i) == pBrother && i > 1)
732.                         {
733.                             pFather.SetElement(i - 1, pBrother.GetElement(1
    ));    // 更改兄弟结点对应的键
734.                         }
735.                     }
736.                 }
737.
738.
739.                 return true;
740.             }
741.
742.             // 情况 2B
743.
```

```
744.                // 父结点中要删除的键
745.                KEY_TYPE NewKey = null;
746.
747.                // 把本结点与兄弟结点合并，无论如何合并到数据较小的结点，这样父结点就无需修改指针
748.
749.                if (mypub.FLAG_LEFT == flag)
750.                {
751.                    pBrother.Combine(pOldNode);
752.                    NewKey = pOldNode.GetElement(1);
753.
754.                    CLeafNode pOldNext = pOldNode.m_pNextNode;
755.                    pBrother.m_pNextNode = pOldNext;
756.                    // 在双向链表中删除结点
757.                    if (null == pOldNext)
758.                    {
759.                        m_pLeafTail = pBrother;
760.                    }
761.                    else
762.                    {
763.                        pOldNext.m_pPrevNode = pBrother;
764.                    }
765.                    // 删除本结点
766.                    pOldNode = null;
767.                }
768.                else
769.                {
770.                    pOldNode.Combine(pBrother);
771.                    NewKey = pBrother.GetElement(1);
772.
773.                    CLeafNode pOldNext = pBrother.m_pNextNode;
774.                    pOldNode.m_pNextNode = pOldNext;
775.                    // 在双向链表中删除结点
776.                    if (null == pOldNext)
777.                    {
778.                        m_pLeafTail = pOldNode;
779.                    }
780.                    else
781.                    {
782.                        pOldNext.m_pPrevNode = pOldNode;
783.                    }
784.                    // 删除本结点
785.                    pBrother = null;
786.                }
```

```
787.
788.                 return DeleteInternalNode(pFather, NewKey);
789.             }
790.
791.             // 清除树
792.             public void ClearTree()
793.             {
794.                 CNode pNode = GetRoot();
795.                 if (null != pNode)
796.                 {
797.                     pNode.DeleteChildren();
798.
799.                     pNode = null;
800.                 }
801.
802.                 m_pLeafHead = null;
803.                 m_pLeafTail = null;
804.                 SetRoot(null);
805.             }
806.
807.             // 打印树
808.             public void PrintTree()
809.             {
810.                 CNode pRoot = GetRoot();
811.                 if (null == pRoot) return;
812.
813.                 CNode p1, p2, p3;
814.                 int i, j, k;
815.                 int total = 0;
816.
817.                 System.Console.Write("\n 第一层\n | ");
818.                 PrintNode(pRoot);
819.                 total = 0;
820.                 System.Console.Write("\n 第二层\n | ");
821.                 for (i = 1; i <= mypub.MAXNUM_POINTER; i++)
822.                 {
823.                     p1 = pRoot.GetPointer(i);
824.                     if (null == p1) continue;
825.                     PrintNode(p1);
826.                     total++;
827.                     if (total % 4 == 0) System.Console.Write("\n | ");
828.                 }
829.                 total = 0;
830.                 System.Console.Write("\n 第三层\n | ");
```

```
831.             for (i = 1; i <= mypub.MAXNUM_POINTER; i++)
832.             {
833.                 p1 = pRoot.GetPointer(i);
834.                 if (null == p1) continue;
835.                 for (j = 1; j <= mypub.MAXNUM_POINTER; j++)
836.                 {
837.                     p2 = p1.GetPointer(j);
838.                     if (null == p2) continue;
839.                     PrintNode(p2);
840.                     total++;
841.                     if (total % 4 == 0) System.Console.Write("\n | ");
842.                 }
843.             }
844.             total = 0;
845.             System.Console.Write("\n 第四层\n | ");
846.             for (i = 1; i <= mypub.MAXNUM_POINTER; i++)
847.             {
848.                 p1 = pRoot.GetPointer(i);
849.                 if (null == p1) continue;
850.                 for (j = 1; j <= mypub.MAXNUM_POINTER; j++)
851.                 {
852.                     p2 = p1.GetPointer(j);
853.                     if (null == p2) continue;
854.                     for (k = 1; k <= mypub.MAXNUM_POINTER; k++)
855.                     {
856.                         p3 = p2.GetPointer(k);
857.                         if (null == p3) continue;
858.                         PrintNode(p3);
859.                         total++;
860.                         if (total % 4 == 0) System.Console.Write("\n | ");
861.                     }
862.                 }
863.             }
864.         }
865.
866.         public void PrintNode(CNode pNode)
867.         {
868.             if (null == pNode)
869.             {
870.                 return;
871.             }
872.
873.             for (int i = 1; i <= mypub.MAXNUM_KEY; i++)
```

```
874.             {
875.                 System.Console.Write(pNode.GetElement(i) + " ");
876.                 if (i >= mypub.MAXNUM_KEY)
877.                 {
878.                     System.Console.Write(" | ");
879.                 }
880.             }
881.         }
882.
883.         // 递归检查结点及其子树是否满足 B+树的定义
884.
885.         // 获取和设置根结点
886.         public CNode GetRoot()
887.         {
888.             return m_Root;
889.         }
890.
891.         public void SetRoot(CNode root)
892.         {
893.             m_Root = root;
894.         }
895.
896.         // 获取和设置深度
897.         public int GetDepth()
898.         {
899.             return m_Depth;
900.         }
901.
902.         public void SetDepth(int depth)
903.         {
904.             m_Depth = depth;
905.         }
906.
907.         // 深度加一
908.         public void IncDepth()
909.         {
910.             m_Depth = m_Depth + 1;
911.         }
912.
913.         // 深度减一
914.         public void DecDepth()
915.         {
916.             if (m_Depth > 0)
917.             {
```

```
                m_Depth = m_Depth - 1;
            }
        }

        // 以下两个变量用于实现双向链表
        public CLeafNode m_pLeafHead;                   // 头结点
        public CLeafNode m_pLeafTail;                     // 尾结点

        // 为插入而查找叶子结点
        protected CLeafNode SearchLeafNode(KEY_TYPE data)
        {
            int i = 0;

            CNode pNode = new CNode();
            pNode = GetRoot();
            // 循环查找对应的叶子结点
            while (null != pNode)
            {
                // 结点为叶子结点，循环结束
                if (NODE_TYPE.NODE_TYPE_LEAF == pNode.GetType())
                {
                    break;
                }

                // 找到第一个键值大于等于 key 的位置
                for (i = 1; i <= pNode.GetCount(); i++)
                {
                    if (String.Compare(data, pNode.GetElement(i)) < 0)
                    {
                        break;
                    }
                }

                pNode = pNode.GetPointer(i);
            }

            return (CLeafNode)pNode;
        }

        //插入键到中间结点
        protected bool InsertInternalNode(CInternalNode pNode, KEY_TYPE key
    , CNode pRightSon)
        {
```

```
961.             if (null == pNode || NODE_TYPE.NODE_TYPE_LEAF == pNode.GetType(
    ))
962.             {
963.                 return false;
964.             }
965.
966.             // 结点未满，直接插入
967.             if (pNode.GetCount() < mypub.MAXNUM_KEY)
968.             {
969.                 return pNode.Insert(key, pRightSon);
970.             }
971.
972.             CInternalNode pBrother = new CInternalNode();  //C++中 new 类名
    表示分配一个类需要的内存空间，并返回其首地址；
973.             KEY_TYPE NewKey = null;
974.             // 分裂本结点
975.             NewKey = pNode.Split(pBrother, key);
976.
977.             if (pNode.GetCount() < pBrother.GetCount())
978.             {
979.                 pNode.Insert(key, pRightSon);
980.             }
981.             else if (pNode.GetCount() > pBrother.GetCount())
982.             {
983.                 pBrother.Insert(key, pRightSon);
984.             }
985.             else    // 两者相等，即键值在第 V 和 V+1 个键值中间的情况，把字节点挂
    到新结点的第一个指针上
986.             {
987.                 pBrother.SetPointer(1, pRightSon);
988.                 pRightSon.SetFather(pBrother);
989.             }
990.
991.             CInternalNode pFather = (CInternalNode)(pNode.GetFather());
992.             // 直到根结点都满了，新生成根结点
993.             if (null == pFather)
994.             {
995.                 pFather = new CInternalNode();
996.                 pFather.SetPointer(1, pNode);                          //
    指针 1 指向原结点
997.                 pFather.SetElement(1, NewKey);                         //
    设置键
998.                 pFather.SetPointer(2, pBrother);                       //
    指针 2 指向新结点
```

```
999.                    pNode.SetFather(pFather);                                   //
    指定父结点
1000.                       pBrother.SetFather(pFather);
    // 指定父结点
1001.                       pFather.SetCount(1);
1002.
1003.                       SetRoot(pFather);
     // 指定新的根结点
1004.                       return true;
1005.                   }
1006.
1007.                   // 递归
1008.                   return InsertInternalNode(pFather, NewKey, pBrother);
1009.               }
1010.
1011.               // 在中间结点中删除键
1012.               protected bool DeleteInternalNode(CInternalNode pNode, KEY_TYPE
    key)
1013.               {
1014.                   bool success = pNode.Delete(key);
1015.                   if (false == success)
1016.                   {
1017.                       return false;
1018.                   }
1019.
1020.                   // 获取父结点
1021.                   CInternalNode pFather = (CInternalNode)(pNode.GetFather());

1022.                   if (null == pFather)
1023.                   {
1024.                       // 如果一个数据都没有了，把根结点的第一个结点作为根结点
1025.                       if (0 == pNode.GetCount())
1026.                       {
1027.                           SetRoot(pNode.GetPointer(1));
1028.                           pNode = null;
1029.                       }
1030.
1031.                       return true;
1032.                   }
1033.
1034.                   // 删除后结点填充度仍>=50%
1035.                   if (pNode.GetCount() >= mypub.ORDER_V)
1036.                   {
```

```
1037.                     for (int i = 1; (String.Compare(key, pFather.GetElement
    (i)) >= 0) && (i <= pFather.GetCount()); i++)
1038.                     {
1039.                         // 如果删除的是父结点的键值，需要更改该键
1040.                         if (pFather.GetElement(i) == key)
1041.                         {
1042.                             pFather.SetElement(i, pNode.GetElement(1));
    // 更改为叶子结点新的第一个元素
1043.                         }
1044.                     }
1045.
1046.                     return true;
1047.                 }
1048.
1049.                 //找到一个最近的兄弟结点(根据 B+树的定义，除了根结点，总是能找到
    的)
1050.                 int flag = mypub.FLAG_LEFT;
1051.                 CInternalNode pBrother = (CInternalNode)(pNode.GetBrother(r
    ef flag));
1052.
1053.                 // 兄弟结点填充度>50%
1054.                 if (pBrother.GetCount() > mypub.ORDER_V)
1055.                 {
1056.                     pNode.MoveOneElement(pBrother);
1057.
1058.                     // 修改父结点的键值
1059.                     if (mypub.FLAG_LEFT == flag)
1060.                     {
1061.                         for (int i = 1; i <= pFather.GetCount() + 1; i++)
1062.                         {
1063.                             if (pFather.GetPointer(i) == pNode && i > 1)
1064.                             {
1065.                                 pFather.SetElement(i - 1, pNode.GetElement(
    1));    // 更改本结点对应的键
1066.                             }
1067.                         }
1068.                     }
1069.                     else
1070.                     {
1071.                         for (int i = 1; i <= pFather.GetCount() + 1; i++)
1072.                         {
1073.                             if (pFather.GetPointer(i) == pNode && i > 1)
1074.                             {
```

```
1075.                                 pFather.SetElement(i - 1, pNode.GetElement(
    1));    // 更改本结点对应的键
1076.                             }
1077.                             if (pFather.GetPointer(i) == pBrother && i > 1)

1078.                             {
1079.                                 pFather.SetElement(i - 1, pBrother.GetEleme
    nt(1));    // 更改兄弟结点对应的键
1080.                             }
1081.                         }
1082.                     }
1083.
1084.                     return true;
1085.                 }
1086.
1087.                 // 父结点中要删除的键：兄弟结点都不大于 50，则需要合并结点，此时
    父结点需要删除键
1088.                 KEY_TYPE NewKey = null;
1089.
1090.                 // 把本结点与兄弟结点合并，无论如何合并到数据较小的结点，这样父
    结点就无需修改指针
1091.                 if (mypub.FLAG_LEFT == flag)
1092.                 {
1093.                     pBrother.Combine(pNode);
1094.                     NewKey = pNode.GetElement(1);
1095.                     pNode = null;
1096.                 }
1097.                 else
1098.                 {
1099.                     pNode.Combine(pBrother);
1100.                     NewKey = pBrother.GetElement(1);
1101.                     pBrother = null;
1102.                 }
1103.
1104.                 // 递归
1105.                 return DeleteInternalNode(pFather, NewKey);
1106.             }
1107.
1108.             protected CNode m_Root;    // 根结点
1109.             protected int m_Depth;      // 树的深度
1110.         };
1111.     }
1112.
```

Table_Operations.xaml.cs

```
1.  using System;
2.  using System.Collections.Generic;
3.  using Windows.Storage;
4.  using Windows.Storage.Pickers;
5.  using Windows.Storage.Streams;
6.  using Windows.UI.Popups;
7.  using Windows.UI.Xaml;
8.  using Windows.UI.Xaml.Controls;
9.  using Windows.UI.Xaml.Input;
10.
11. // The Blank Page item template is documented at https://go.microsoft.com/fw
    link/?LinkId=234238
12.
13. namespace MiniSQL
14. {
15.     /// <summary>
16.     /// An empty page that can be used on its own or navigated to within a F
    rame.
17.     /// </summary>
18.     public sealed partial class Table_Operations : Page
19.     {
20.         public Table_Operations()
21.         {
22.             this.InitializeComponent();
23.
24.             foreach (Table element in (Application.Current as App).tables)
25.             {
26.                 ListViewItem item = new ListViewItem();
27.                 item.Content = element.get_name();
28.                 item.Tapped += new TappedEventHandler(Show_Detail);
29.                 table_lists.Items.Add(item);
30.             }
31.         }
32.
33.         private void main(object sender, TappedRoutedEventArgs e)
34.         {
35.             Frame.Navigate(typeof(MainPage), "");
36.         }
37.
38.         private void query(object sender, TappedRoutedEventArgs e)
39.         {
40.             Frame.Navigate(typeof(Querys), "");
41.         }
```

```
42.
43.         private void exit(object sender, TappedRoutedEventArgs e)
44.         {
45.             (Application.Current as App).Exit();
46.         }
47.
48.         private void table(object sender, TappedRoutedEventArgs e)
49.         {
50.             Frame.Navigate(typeof(Table_Operations), "");
51.         }
52.
53.         private void index(object sender, TappedRoutedEventArgs e)
54.         {
55.             Frame.Navigate(typeof(Index_Operations), "");
56.         }
57.
58.         private void insert(object sender, TappedRoutedEventArgs e)
59.         {
60.             Frame.Navigate(typeof(Insert_Operations), "");
61.         }
62.
63.         private void delete(object sender, TappedRoutedEventArgs e)
64.         {
65.             Frame.Navigate(typeof(Delete_Operations), "");
66.         }
67.
68.         private void goback(NavigationView sender, NavigationViewBackRequest
    edEventArgs args)
69.         {
70.             Frame.GoBack();
71.         }
72.
73.         private void execute(object sender, RoutedEventArgs e)
74.         {
75.             /* write your codes behind */
76.             try
77.             {
78.                 string sqlcode = source.Text.ToLower();
79.
80.                 if (source.Text == "")
81.                 {
82.                     if (attribute_name.Text.Length == 0 || attribute_type.Te
    xt.Length == 0)
83.                     {
```

```
84.                         throw new NullReferenceException();
85.                     }
86.
87.                     if (!(bool)create.IsChecked && !(bool)del.IsChecked)
88.                     {
89.                         MessageDialog msg = new MessageDialog("You haven't c
    hosen an operation yet!");
90.                         _ = msg.ShowAsync();
91.                         return;
92.                     }
93.
94.                     if ((bool)create.IsChecked)
95.                     {
96.                         List<Attribute> attributes = new List<Attribute>();

97.
98.                         for (int i = 0; i < (Application.Current as App).tab
    les.Count; i++)
99.                         {
100.                            if ((Application.Current as App).tables[i].get_
    name() == table_name.Text)
101.                            {
102.                                MessageDialog msg = new MessageDialog("You
    have already created this table!");
103.                                _ = msg.ShowAsync();
104.                                return;
105.                            }
106.                        }
107.
108.                        List<string> name = new List<string>(attribute_name
    .Text.Split("\r"));
109.                        List<string> type = new List<string>(attribute_type
    .Text.Split("\r"));
110.
111.                        if (name.Count != type.Count)
112.                        {
113.                            MessageDialog msg = new MessageDialog("Unmatche
    d attribute name and type");
114.                            _ = msg.ShowAsync();
115.                            return;
116.                        }
117.
118.                        for (int i = 0; i < name.Count; i++)
119.                        {
```

```
120.                             if (type[i].IndexOf("int") == -
    1 && type[i].IndexOf("float") == -1 && type[i].IndexOf("char") == -1)
121.                             {
122.                                 MessageDialog msg = new MessageDialog("Wron
    g type!");
123.                                 _ = msg.ShowAsync();
124.                                 return;
125.                             }
126.
127.                             Attribute temp = new Attribute(name[i], type[i]
    , primary_key.Text.IndexOf(name[i]) != -1);
128.                             attributes.Add(temp);
129.                         }
130.
131.                         Table table = new Table(table_name.Text, attributes
    );
132.                         (Application.Current as App).tables.Add(table);
133.
134.                         ListViewItem new_table = new ListViewItem();
135.                         new_table.Content = table_name.Text;
136.                         new_table.Tapped += new TappedEventHandler(Show_Det
    ail);
137.                         table_lists.Items.Add(new_table);
138.                     }
139.                     else
140.                     {
141.                         bool flag = true;
142.
143.                         for (int i = 0; i < (Application.Current as App).ta
    bles.Count; i++)
144.                         {
145.                             if ((Application.Current as App).tables[i].get_
    name() == table_name.Text)
146.                             {
147.                                 flag = false;
148.                                 (Application.Current as App).tables.RemoveA
    t(i);
149.                                 table_lists.Items.RemoveAt(i + 1);
150.                             }
151.                         }
152.
153.                         if (flag)
154.                         {
```

```
155.                             MessageDialog msg = new MessageDialog("Can not
    find the table.");
156.                             _ = msg.ShowAsync();
157.                             return;
158.                         }
159.                     }
160.                 }
161.                 else
162.                 {
163.                     if (sqlcode.IndexOf("create") != -
    1)        /* create */
164.                     {
165.                         List<string> code = new List<string>(sqlcode.Split(
    "\r"));
166.                         string tablename = code[0].Substring(13, code[0].Le
    ngth - 15);
167.
168.                         for (int i = 0; i < (Application.Current as App).ta
    bles.Count; i++)
169.                         {
170.                             if ((Application.Current as App).tables[i].get_
    name() == tablename)
171.                             {
172.                                 MessageDialog msg = new MessageDialog("You
    have already created this table!");
173.                                 _ = msg.ShowAsync();
174.                                 return;
175.                             }
176.                         }
177.
178.                         if (sqlcode.IndexOf("primary key") != -1)
179.                         {
180.                             List<Attribute> tableattribute = new List<Attri
    bute>();
181.
182.                             int primary_begin = code[code.Count - 2].IndexO
    f('(') + 2;
183.                             int primary_end = code[code.Count - 2].IndexOf(
    ')') - 2;
184.                             List<string> keys = new List<string>(code[code.
    Count - 2].Substring(primary_begin, primary_end - primary_begin + 1).Split("
    , "));
185.
186.                             for (int j = 1; j < code.Count - 2; j++)
```

```
187.                                {
188.                                    string element = code[j];
189.                                    int index_name = 0, index_type = 0;
190.
191.                                    for (int i = 0; i < element.Length; i++)
192.                                    {
193.                                        if (element[i] <= 'z' && element[i] >=
    'a')
194.                                        {
195.                                            index_name = i;
196.                                            break;
197.                                        }
198.                                    }
199.
200.                                    for (int i = index_name; i < element.Length
    ; i++)
201.                                    {
202.                                        if (element[i] == ' ')
203.                                        {
204.                                            index_type = i + 1;
205.                                            break;
206.                                        }
207.                                    }
208.
209.                                    string attr_name = element.Substring(index_
    name, index_type - index_name - 1);
210.                                    string attr_type = element.Substring(index_
    type, element.Length - index_type - 1);
211.
212.                                    if (attr_type.IndexOf("int") == -
    1 && attr_type.IndexOf("float") == -1 && attr_type.IndexOf("char") == -1)
213.                                    {
214.                                        MessageDialog msg = new MessageDialog("
    Wrong type!");
215.                                        _ = msg.ShowAsync();
216.                                        return;
217.                                    }
218.
219.                                    Attribute temp = new Attribute(attr_name, a
    ttr_type, (keys.IndexOf(attr_name) != -1));
220.                                    tableattribute.Add(temp);
221.                                }
222.
```

```
223.                             Table temp_table = new Table(tablename, tableat
    tribute);
224.                             (Application.Current as App).tables.Add(temp_ta
    ble);
225.
226.                             ListViewItem new_table = new ListViewItem();
227.                             new_table.Content = tablename;
228.                             new_table.Tapped += new TappedEventHandler(Show
    _Detail);
229.                             table_lists.Items.Add(new_table);
230.                         }
231.                         else
232.                         {
233.                             List<Attribute> tableattribute = new List<Attri
    bute>();
234.
235.                             for (int j = 1; j < code.Count - 1; j++)
236.                             {
237.                                 string element = code[j];
238.                                 int index_name = 0, index_type = 0;
239.
240.                                 for (int i = 0; i < element.Length; i++)
241.                                 {
242.                                     if (element[i] <= 'z' && element[i] >=
    'a')
243.                                     {
244.                                         index_name = i;
245.                                         break;
246.                                     }
247.                                 }
248.
249.                                 for (int i = index_name; i < element.Length
    ; i++)
250.                                 {
251.                                     if (element[i] == ' ')
252.                                     {
253.                                         index_type = i + 1;
254.                                         break;
255.                                     }
256.                                 }
257.
258.                                 string attr_name = element.Substring(index_
    name, index_type - index_name - 1);
```

```
259.                                string attr_type = element.Substring(index_
    type, element.Length - index_type - (j == code.Count - 2 ? 0 : 1));
260.
261.                                if (attr_type.IndexOf("int") == -
    1 && attr_type.IndexOf("float") == -1 && attr_type.IndexOf("char") == -1)
262.                                {
263.                                    MessageDialog msg = new MessageDialog("
    Wrong type!");
264.                                    _ = msg.ShowAsync();
265.                                    return;
266.                                }
267.
268.                                Attribute temp = new Attribute(attr_name, a
    ttr_type, false);
269.                                tableattribute.Add(temp);
270.                            }
271.
272.                            Table temp_table = new Table(tablename, tableat
    tribute);
273.                            (Application.Current as App).tables.Add(temp_ta
    ble);
274.
275.                            ListViewItem new_table = new ListViewItem();
276.                            new_table.Content = tablename;
277.                            new_table.Tapped += new TappedEventHandler(Show
    _Detail);
278.                            table_lists.Items.Add(new_table);
279.                        }
280.                    }
281.                    else if (sqlcode.IndexOf("drop") != -
    1)     /* drop table */
282.                    {
283.                        string tablename = sqlcode.Substring(11, sqlcode.Le
    ngth - 13);
284.                        bool flag = true;
285.
286.                        for (int i = 0; i < (Application.Current as App).ta
    bles.Count; i++)
287.                        {
288.                            if ((Application.Current as App).tables[i].get_
    name() == tablename)
289.                            {
290.                                flag = false;
```

```
291.                                 (Application.Current as App).tables.RemoveA
    t(i);
292.                                 table_lists.Items.RemoveAt(i + 1);
293.                             }
294.                         }
295.
296.                         if (flag)
297.                         {
298.                             MessageDialog msg = new MessageDialog("Can not
    find table.");
299.                             _ = msg.ShowAsync();
300.                         }
301.                     }
302.                     else                                        /* wrong in
    put */
303.                     {
304.                         throw new ArgumentException();
305.                     }
306.                 }
307.                 MessageDialog msge = new MessageDialog("The operation has s
    ucceeded! Now please choose a path to store your table information.");
308.                 _ = msge.ShowAsync();
309.             }
310.             catch (ArgumentException)
311.             {
312.                 MessageDialog msg = new MessageDialog("Check your SQL code
    carefully!");
313.                 _ = msg.ShowAsync();
314.             }
315.             catch (NullReferenceException)
316.             {
317.                 MessageDialog msg = new MessageDialog("You haven't input an
    y attribute's name or type.");
318.                 _ = msg.ShowAsync();
319.             }
320.             catch (Exception)
321.             {
322.                 MessageDialog msg = new MessageDialog("Something else wrong
     happened!");
323.                 _ = msg.ShowAsync();
324.             }
325.         }
326.
327.         private async void writedata(object sender, RoutedEventArgs e)
```

```
328.         {
329.             try
330.             {
331.                 string data = "";
332.
333.                 for (int i = 0; i < (Application.Current as App).tables.Cou
    nt; i++)
334.                 {
335.                     Table element = (Application.Current as App).tables[i];

336.                     data += element.get_name() + "\n";
337.                     foreach (Attribute temp in element.get_attribute())
338.                     {
339.                         data += temp.get_name() + " " + temp.get_type() + "
     " + temp.get_primary().ToString();
340.                         data += "\n";
341.                     }
342.
343.                     data += (Application.Current as App).tables[i].get_reco
    rd_number().ToString() + ((Application.Current as App).tables[i].get_record_
    number() == 0 ? "" : "\n");
344.
345.                     for (int j = 0; j < element.get_record_number(); j++)
346.                     {
347.                         for (int k = 0; k < element.get_attribute().Count;
    k++)
348.                         {
349.                             data += element.get_attribute()[k].get_values()
    [j] + (k == element.get_attribute().Count - 1 ? "" : " ");
350.                         }
351.                         data += ((i == (Application.Current as App).tables.
    Count - 1 && j == element.get_record_number() - 1) ? "" : "\n");
352.                     }
353.
354.                     data += (i == (Application.Current as App).tables.Count
     - 1 ? "" : "\n");
355.                 }
356.
357.
358.                 FileSavePicker fp = new FileSavePicker();
359.                 var filedb = new[] { ".txt" };
360.                 fp.FileTypeChoices.Add("Plain Text", new List<string>() { "
    .txt" });
```

```
361.                 fp.SuggestedFileName = "table information" + DateTime.Now.D
    ay + "-" + DateTime.Now.Month + "-" + DateTime.Now.Year;
362.                 StorageFile sf = await fp.PickSaveFileAsync();
363.
364.                 if (sf != null)
365.                 {
366.                     using (StorageStreamTransaction transaction = await sf.
    OpenTransactedWriteAsync())
367.                     {
368.                         using (DataWriter dataWriter = new DataWriter(trans
    action.Stream))
369.                         {
370.                             dataWriter.WriteString(data);
371.                             transaction.Stream.Size = await dataWriter.Stor
    eAsync();
372.                             await transaction.CommitAsync();
373.                         }
374.                     }
375.                 }
376.
377.                 MessageDialog msg = new MessageDialog("Operation succeed!")
    ;
378.                 _ = msg.ShowAsync();
379.             }
380.             catch (Exception)
381.             {
382.                 MessageDialog msg = new MessageDialog("Something wrong happ
    ened.");
383.                 _ = msg.ShowAsync();
384.             }
385.
386.         }
387.
388.         private void Show_Detail(object sender, TappedRoutedEventArgs e)
389.         {
390.             try
391.             {
392.                 ListViewItem item = sender as ListViewItem;
393.                 string output = "The table ";
394.
395.                 foreach (Table temp in (Application.Current as App).tables)

396.                 {
397.                     if (temp.get_name() == item.Content.ToString())
```

```
398.                     {
399.                         output += temp.get_name();
400.                         output += " has these attributes:\n";
401.                         List<Attribute> attributes = temp.get_attribute();
402.
403.                         for (int i = 0; i < attributes.Count; i++)
404.                         {
405.                             output += "        ";
406.                             output += attributes[i].get_name();
407.                             output += ":        ";
408.                             output += attributes[i].get_type();
409.                             output += "        and it is" + (attributes[i].
    get_primary() ? " a primary key." : " not a primary key.");
410.                             output += "\n";
411.                         }
412.
413.                         MessageDialog msg = new MessageDialog(output);
414.                         _ = msg.ShowAsync();
415.                     }
416.                 }
417.             }
418.             catch (Exception)
419.             {
420.                 MessageDialog msg = new MessageDialog("Something Wrong happ
    ened! Check again.");
421.                 _ = msg.ShowAsync();
422.             }
423.         }
424.     }
425. }
```

Table_Operations.xaml

```
1.  <Page
2.      x:Class="MiniSQL.Table_Operations"
3.      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4.      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5.      xmlns:local="using:MiniSQL"
6.      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7.      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8.      mc:Ignorable="d"
9.      Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
10.
11.     <Grid>
```

```
12.         <NavigationView x:Name="Navigation" IsBackEnabled="True" IsSettingsV
    isible="True" IsTabStop="False" BackRequested="goback">
13.             <NavigationView.MenuItems>
14.                 <NavigationViewItemHeader Content="Functions"/>
15.                 <NavigationViewItem Icon="Home" Content="Main Page" Tapped="
    main"/>
16.                 <NavigationViewItemSeparator/>
17.
18.                 <NavigationViewItem Icon="Import" Content="Table Operations"
     Tapped="table"></NavigationViewItem>
19.                 <NavigationViewItem Icon="List" Content="Index Operations" T
    apped="index"/>
20.                 <NavigationViewItem Icon="Find" Content="Query" Tapped="quer
    y"></NavigationViewItem>
21.                 <NavigationViewItem Icon="AllApps" Content="Insert" Tapped="
    insert"/>
22.                 <NavigationViewItem Icon="Delete" Content="Delete" Tapped="d
    elete"></NavigationViewItem>
23.                 <NavigationViewItem Icon="Next" Content="Export" Tapped="wri
    tedata"/>
24.                 <NavigationViewItem Icon="Undo" Content="Exit" Tapped="exit"
    />
25.             </NavigationView.MenuItems>
26.
27.             <Grid>
28.                 <ListView x:Name="table_lists" Height="1080" Width="400" Hor
    izontalAlignment="Right" BorderBrush="Red" Background="#FF323232">
29.                     <ListViewHeaderItem Content="Table Lists
     "></ListViewHeaderItem>
30.                 </ListView>
31.
32.                 <Grid Width="1200" HorizontalAlignment="Left">
33.                     <Image Source="/Images/zjulib.jpg" Width="421.75" Height
    ="117.25" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0, 50
    , 0, 0"></Image>
34.                     <TextBlock Width="400" Height="40" TextAlignment="Center
    " FontSize="30" HorizontalAlignment="Center" VerticalAlignment="Top" Margin=
    "-600, 200, 0,0">Input your SQL Code Here</TextBlock>
35.                     <TextBox x:Name="source" Width="400" Height="400" Horizo
    ntalAlignment="Center" VerticalAlignment="Top" Margin="-
    600, 300, 0, 0" FontFamily="Consolas" FontSize="20" AcceptsReturn="True"/>
36.                     <Button Width="150" Height="60" FontSize="30" VerticalAl
    ignment="Top" HorizontalAlignment="Center" Margin="-
    800, 750, 0, 0" Click="execute">Execute</Button>
```

```
37.                     <Button Width="150" Height="60" FontSize="30" VerticalAl
    ignment="Top" HorizontalAlignment="Center" Margin="-
    400, 750, 0, 0" Click="writedata">Export</Button>
38.                     <Image Source="/Images/under.png" Width="557" Height="11
    2" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0, 900, 0, 0
    "></Image>
39.
40.                     <TextBlock HorizontalAlignment="Center" VerticalAlignmen
    t="Center" Margin="0, 0, 0, 0" FontSize="50">OR</TextBlock>
41.
42.                     <TextBlock Width="500" Height="40" TextAlignment="Center
    " FontSize="30" HorizontalAlignment="Center" VerticalAlignment="Top" Margin=
    "600, 200, 0,0">Quick Operation Using Templates</TextBlock>
43.                     <TextBlock HorizontalAlignment="Center" VerticalAlignmen
    t="Top" FontSize="20" Width="150" TextAlignment="Center" Margin="300, 300, 0
    , 0">Table Name:</TextBlock>
44.                     <TextBox x:Name="table_name" PlaceholderText="Enter your
     table's name" HorizontalAlignment="Center" VerticalAlignment="Top" Width="2
    50" Height="40" Margin="800, 300, 0, 0" FontSize="20"></TextBox>
45.                     <TextBlock HorizontalAlignment="Center" VerticalAlignmen
    t="Top" FontSize="20" Width="150" TextAlignment="Center" Margin="300, 350, 0
    , 0">Operation:</TextBlock>
46.                     <RadioButton x:Name="create" HorizontalAlignment="Center
    " VerticalAlignment="Top" Margin="700, 350, 0, 0">Create</RadioButton>
47.                     <RadioButton x:Name="del" HorizontalAlignment="Center" V
    erticalAlignment="Top" Margin="900, 350, 0, 0">Delete</RadioButton>
48.                     <TextBlock HorizontalAlignment="Center" VerticalAlignmen
    t="Top" FontSize="20" Width="150" TextAlignment="Center" Margin="300, 400, 0
    , 0">Attribute Name:</TextBlock>
49.                     <TextBox x:Name="attribute_name" PlaceholderText="Enter
    attributes' name one line with single name" HorizontalAlignment="Center" Ver
    ticalAlignment="Top" Width="250" Height="100" Margin="800, 400, 0, 0" Accept
    sReturn="True" FontSize="20"></TextBox>
50.                     <TextBox x:Name="attribute_type" PlaceholderText="Enter
    attributes' name one line with single name" HorizontalAlignment="Center" Ver
    ticalAlignment="Top" Width="250" Height="100" Margin="800, 550, 0, 0" Accept
    sReturn="True" FontSize="20"></TextBox>
51.                     <TextBlock HorizontalAlignment="Center" VerticalAlignmen
    t="Top" FontSize="20" Width="150" TextAlignment="Center" Margin="300, 550, 0
    , 0">Attribute Type:</TextBlock>
52.                     <TextBlock HorizontalAlignment="Center" VerticalAlignmen
    t="Top" FontSize="20" Width="150" TextAlignment="Center" Margin="300, 700, 0
    , 0">Primary Keys:</TextBlock>
```

```
                    <TextBox x:Name="primary_key" PlaceholderText="Enter your table's primary keys" HorizontalAlignment="Center" VerticalAlignment="Top" Width="250" Height="100" Margin="800, 700, 0, 0" FontSize="20" AcceptsReturn="True"></TextBox>
                </Grid>

            </Grid>

            <NavigationView.PaneFooter>
                <HyperlinkButton x:Name="MoreInfoBtn" Content="Email Me" Margin="12,0" NavigateUri="mailto:ZJU.SLM@gmail.com"/>
            </NavigationView.PaneFooter>
        </NavigationView>
    </Grid>
</Page>
```

Querys.xaml.cs

```
using System;
using System.Collections.Generic;
using Windows.UI.Popups;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Input;
using Windows.Storage.Streams;
using Windows.Storage;
using Windows.Storage.Pickers;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace MiniSQL
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class Querys : Page
    {
        public Querys()
        {
            this.InitializeComponent();
        }

```

```
25.         private void goback(NavigationView sender, NavigationViewBackRequest
    edEventArgs args)
26.         {
27.             Frame.GoBack();
28.         }
29.
30.         private void main(object sender, TappedRoutedEventArgs e)
31.         {
32.             Frame.Navigate(typeof(MainPage), "");
33.         }
34.
35.         private void query(object sender, TappedRoutedEventArgs e)
36.         {
37.             Frame.Navigate(typeof(Querys), "");
38.         }
39.
40.         private void exit(object sender, TappedRoutedEventArgs e)
41.         {
42.             (Application.Current as App).Exit();
43.         }
44.
45.         private void table(object sender, TappedRoutedEventArgs e)
46.         {
47.             Frame.Navigate(typeof(Table_Operations), "");
48.         }
49.
50.         private void index(object sender, TappedRoutedEventArgs e)
51.         {
52.             Frame.Navigate(typeof(Index_Operations), "");
53.         }
54.
55.         private void insert(object sender, TappedRoutedEventArgs e)
56.         {
57.             Frame.Navigate(typeof(Insert_Operations), "");
58.         }
59.
60.         private void delete(object sender, TappedRoutedEventArgs e)
61.         {
62.             Frame.Navigate(typeof(Delete_Operations), "");
63.         }
64.
65.         private void execute(object sender, RoutedEventArgs e)
66.         {
67.             result.Visibility = Visibility.Visible;
```

```
68.
69.             try
70.             {
71.                 string sqlcode = source.Text.ToLower();
72.
73.                 if (source.Text == "")
74.                 {
75.                     if (attribute_name.Text.Length == 0)
76.                     {
77.                         throw new NullReferenceException();
78.                     }
79.
80.                     List<Attribute> attributes = new List<Attribute>();
81.
82.                     for (int i = 0; i <= (Application.Current as App).tables
    .Count; i++)
83.                     {
84.                         if (i == (Application.Current as App).tables.Count)

85.                         {
86.                             MessageDialog msg = new MessageDialog("There is
    no this table!");
87.                             _ = msg.ShowAsync();
88.                             return;
89.                         }
90.                         if ((Application.Current as App).tables[i].get_name(
    ) == table_name.Text)
91.                         {
92.                             attributes = (Application.Current as App).tables
    [i].get_attribute();
93.                             break;
94.                         }
95.                     }
96.
97.                     List<string> name = new List<string>(attribute_name.Text
    .Split("\r"));
98.                     List<Attribute> selattributes = new List<Attribute>();
99.
100.                     for (int k = 0; k < attributes.Count; k++)
101.                     {
102.                         Attribute temp = new Attribute(attributes[k].get_na
    me(), attributes[k].get_type(), attributes[k].get_primary());
103.                         selattributes.Add(temp);
104.                     }
```

```
105.
106.                     if (!(bool)and.IsChecked && !(bool)or.IsChecked)
107.                     {
108.                         var item = (ComboBoxItem)operator1.SelectedItem;
109.                         select(item.Content.ToString(), ref selattributes,
    attributes, attribute11, attribute12);
110.                     }
111.                     else if (!(bool)and.IsChecked && (bool)or.IsChecked)
112.                     {
113.                         var item1 = (ComboBoxItem)operator1.SelectedItem;
114.                         var item2 = (ComboBoxItem)operator2.SelectedItem;
115.                         List<Attribute> templist1 = new List<Attribute>();

116.
117.                         for (int k = 0; k < attributes.Count; k++)
118.                         {
119.                             Attribute temp = new Attribute(attributes[k].ge
    t_name(), attributes[k].get_type(), attributes[k].get_primary());
120.                             templist1.Add(temp);
121.                         }
122.
123.                         List<Attribute> templist2 = new List<Attribute>();

124.
125.                         for (int k = 0; k < attributes.Count; k++)
126.                         {
127.                             Attribute temp = new Attribute(attributes[k].ge
    t_name(), attributes[k].get_type(), attributes[k].get_primary());
128.                             templist2.Add(temp);
129.                         }
130.
131.                         select(item1.Content.ToString(), ref templist1, att
    ributes, attribute11, attribute12);
132.                         select(item2.Content.ToString(), ref templist2, att
    ributes, attribute21, attribute22);
133.
134.                         //merge
135.                         for (int i = 0; i < templist1[0].get_values().Count
    ; i++)
136.                         {
137.                             for (int k = 0; k < templist2[0].get_values().C
    ount; k++)
138.                             {
139.                                 int flag = 1;
```

```
140.
141.                                 for (int j = 0; j < templist1.Count; j++)
142.                                 {
143.                                     if (templist1[j].get_values()[i] != tem
    plist2[j].get_values()[k])
144.                                     {
145.                                         flag = 0;
146.                                         break;
147.                                     }
148.                                 }
149.                                 if (flag == 1)
150.                                     break;
151.                                 if (k == templist2[0].get_values().Count -
    1)
152.                                 {
153.                                     for (int l = 0; l < selattributes.Count
    ; l++)
154.                                         selattributes[l].insert_value(templ
    ist1[l].get_values()[i]);
155.                                 }
156.                             }
157.                         }
158.                         for (int k = 0; k < templist2[0].get_values().Count
    ; k++)
159.                         {
160.                             for (int l = 0; l < selattributes.Count; l++)
161.                                 selattributes[l].insert_value(templist2[l].
    get_values()[k]);
162.                         }
163.                     }
164.                     else
165.                     {
166.                         var item1 = (ComboBoxItem)operator1.SelectedItem;
167.                         var item2 = (ComboBoxItem)operator2.SelectedItem;
168.                         List<Attribute> templist = new List<Attribute>();
169.
170.                         for (int k = 0; k < attributes.Count; k++)
171.                         {
172.                             Attribute temp = new Attribute(attributes[k].ge
    t_name(), attributes[k].get_type(), attributes[k].get_primary());
173.                             templist.Add(temp);
174.                         }
175.
```

```
176.                         select(item1.Content.ToString(), ref templist, attr
    ibutes, attribute11, attribute12);
177.                         select(item2.Content.ToString(), ref selattributes,
     templist, attribute21, attribute22);
178.                     }
179.
180.                     List<Attribute> outputlist = new List<Attribute>();
181.
182.                     for (int i = 0; i < name.Count; i++)
183.                     {
184.                         for (int k = 0; k < selattributes.Count; k++)
185.                         {
186.                             if (name[i] == selattributes[k].get_name())
187.                             {
188.                                 Attribute temp = new Attribute();
189.                                 temp = selattributes[k];
190.                                 outputlist.Add(temp);
191.                                 break;
192.                             }
193.                             if (k == selattributes.Count)
194.                             {
195.                                 MessageDialog msg = new MessageDialog("Ther
    e is no attribute " + name[i] + " !");
196.                                 _ = msg.ShowAsync();
197.                                 return;
198.                             }
199.                         }
200.                     }
201.
202.                     Show_Detail(outputlist);
203.                 }
204.                 else
205.                 {
206.                     List<string> code = new List<string>(sqlcode.Split("\r"
    ));
207.                     List<Attribute> attributes = new List<Attribute>();
208.                     List<string> name = new List<string>();
209.                     string tablename = code[1].Substring(5, code[1].Length
    - 5);
210.
211.                     for (int i = 0; i <= (Application.Current as App).table
    s.Count; i++)
212.                     {
```

```
213.                         if ((Application.Current as App).tables[i].get_name
    () == tablename)
214.                         {
215.                             attributes = (Application.Current as App).table
    s[i].get_attribute();
216.                             break;
217.                         }
218.                     }
219.
220.                     int temp = 7;
221.                     List<Attribute> selattributes = new List<Attribute>();

222.
223.                     for (int k = 0; k < attributes.Count; k++)
224.                     {
225.                         Attribute temp1 = new Attribute(attributes[k].get_n
    ame(), attributes[k].get_type(), attributes[k].get_primary());
226.                         selattributes.Add(temp1);
227.                     }
228.
229.                     while (code[0].IndexOf(",", temp) != -1)
230.                     {
231.                         int temp2 = temp;
232.                         temp = code[0].IndexOf(",", temp) + 1;
233.                         name.Add(code[0].Substring(temp2, temp - temp2 - 1)
    );
234.                     }
235.
236.                     name.Add(code[0].Substring(temp, code[0].Length - temp)
    );
237.
238.                     if (code.Count == 2)
239.                     {
240.                         selattributes = attributes;
241.                     }
242.                     else
243.                     {
244.                         if (code[2].IndexOf("and") != -1)
245.                         {
246.                             List<string> temp1 = new List<string>(code[2].S
    plit("and"));
247.                             string content1, content2;
248.                             string attribute11, attribute12, attribute21, a
    ttribute22;
```

```
249.                             List<Attribute> templist = new List<Attribute>(
    );
250.
251.                             for (int k = 0; k < attributes.Count; k++)
252.                             {
253.                                 Attribute temp3 = new Attribute(attributes[
    k].get_name(), attributes[k].get_type(), attributes[k].get_primary());
254.                                 templist.Add(temp3);
255.                             }
256.
257.                             content1 = findcontent(temp1[0]);
258.                             content2 = findcontent(temp1[1]);
259.                             attribute11 = temp1[0].Substring(6, temp1[0].In
    dexOf(content1) - 6);
260.                             attribute12 = temp1[0].Substring(temp1[0].Index
    Of(content1) + content1.Length, temp1[0].Length - temp1[0].IndexOf(content1)
     - content1.Length - 1);
261.                             attribute21 = temp1[1].Substring(1, temp1[1].In
    dexOf(content2) - 1);
262.                             attribute22 = temp1[1].Substring(temp1[1].Index
    Of(content2) + content2.Length, temp1[1].Length - temp1[1].IndexOf(content2)
     - content2.Length);
263.                             select(content1, ref templist, attributes, attr
    ibute11, attribute12);
264.                             select(content2, ref selattributes, templist, a
    ttribute21, attribute22);
265.                         }
266.                         else if (code[2].IndexOf("or") != -1)
267.                         {
268.                             List<string> temp1 = new List<string>(code[2].S
    plit("or"));
269.                             string content1, content2;
270.                             string attribute11, attribute12, attribute21, a
    ttribute22;
271.                             content1 = findcontent(temp1[0]);
272.                             content2 = findcontent(temp1[1]);
273.                             attribute11 = temp1[0].Substring(6, temp1[0].In
    dexOf(content1) - 6);
274.                             attribute12 = temp1[0].Substring(temp1[0].Index
    Of(content1) + content1.Length, temp1[0].Length - temp1[0].IndexOf(content1)
     - content1.Length - 1);
275.                             attribute21 = temp1[1].Substring(1, temp1[1].In
    dexOf(content2) - 1);
```

```
276.                                attribute22 = temp1[1].Substring(temp1[1].Index
    Of(content2) + content2.Length, temp1[1].Length - temp1[1].IndexOf(content2)
     - content2.Length);
277.                                List<Attribute> templist1 = new List<Attribute>
    ();
278.
279.                                for (int k = 0; k < attributes.Count; k++)
280.                                {
281.                                    Attribute temp3 = new Attribute(attributes[
    k].get_name(), attributes[k].get_type(), attributes[k].get_primary());
282.                                    templist1.Add(temp3);
283.                                }
284.
285.                                List<Attribute> templist2 = new List<Attribute>
    ();
286.
287.                                for (int k = 0; k < attributes.Count; k++)
288.                                {
289.                                    Attribute temp3 = new Attribute(attributes[
    k].get_name(), attributes[k].get_type(), attributes[k].get_primary());
290.                                    templist2.Add(temp3);
291.                                }
292.
293.                                select(content1, ref templist1, attributes, att
    ribute11, attribute12);
294.                                select(content2, ref templist2, attributes, att
    ribute21, attribute22);
295.
296.                                //merge
297.                                for (int i = 0; i < templist1[0].get_values().C
    ount; i++)
298.                                {
299.                                    for (int k = 0; k < templist2[0].get_values
    ().Count; k++)
300.                                    {
301.                                        int flag = 1;
302.                                        for (int j = 0; j < templist1.Count; j+
    +)
303.                                        {
304.                                            if (templist1[j].get_values()[i] !=
     templist2[j].get_values()[k])
305.                                            {
306.                                                flag = 0;
307.                                                break;
```

```
308.                                        }
309.                                    }
310.                                    if (flag == 1)
311.                                        break;
312.                                    if (k == templist2[0].get_values().Coun
    t - 1)
313.                                    {
314.                                        for (int l = 0; l < selattributes.C
    ount; l++)
315.                                            selattributes[l].insert_value(t
    emplist1[l].get_values()[i]);
316.                                    }
317.                                }
318.                            }
319.                            for (int k = 0; k < templist2[0].get_values().C
    ount; k++)
320.                            {
321.                                for (int l = 0; l < selattributes.Count; l+
    +)
322.                                    selattributes[l].insert_value(templist2
    [l].get_values()[k]);
323.                            }
324.                        }
325.                        else
326.                        {
327.                            string temp1 = code[2];
328.                            string content1;
329.                            string attribute11, attribute12;
330.                            content1 = findcontent(temp1);
331.                            attribute11 = temp1.Substring(6, temp1.IndexOf(
    content1) - 6);
332.                            attribute12 = temp1.Substring(temp1.IndexOf(con
    tent1) + content1.Length, temp1.Length - temp1.IndexOf(content1) - content1.
    Length);
333.                            select(content1, ref selattributes, attributes,
     attribute11, attribute12);
334.                        }
335.                    }
336.                    List<Attribute> outputlist = new List<Attribute>();
337.                    for (int i = 0; i < name.Count; i++)
338.                    {
339.                        for (int k = 0; k < selattributes.Count; k++)
340.                        {
341.                            if (name[i] == selattributes[k].get_name())
```

```
342.                                {
343.                                    Attribute temp3 = new Attribute();
344.                                    temp3 = selattributes[k];
345.                                    outputlist.Add(temp3);
346.                                    break;
347.                                }
348.                                if (k == selattributes.Count)
349.                                {
350.                                    MessageDialog msg = new MessageDialog("Ther
    e is no attribute " + name[i] + " !");
351.                                    _ = msg.ShowAsync();
352.                                    return;
353.                                }
354.                            }
355.                        }
356.
357.                        Show_Detail(outputlist);
358.                    }
359.                }
360.                catch (ArgumentException)
361.                {
362.                    MessageDialog msg = new MessageDialog("Check your SQL code
    carefully!");
363.                    _ = msg.ShowAsync();
364.                }
365.                catch (NullReferenceException)
366.                {
367.                    MessageDialog msg = new MessageDialog("You haven't input an
    y attribute's name or type.");
368.                    _ = msg.ShowAsync();
369.                }
370.                catch (Exception)
371.                {
372.                    MessageDialog msg = new MessageDialog("Something else wrong
     happened!");
373.                    _ = msg.ShowAsync();
374.                }
375.            }
376.
377.            public void select(string content, ref List<Attribute> selattribute
    s, List<Attribute> attributes, TextBox attribute11, TextBox attribute12)
378.            {
379.                int conditionnum1 = 0;
380.
```

```
381.             for (int k = 0; k < attributes.Count; k++)
382.             {
383.                 if (attribute11.Text == attributes[k].get_name())
384.                 {
385.                     conditionnum1 = k;
386.                     break;
387.                 }
388.                 if (k == attributes.Count)
389.                 {
390.                     MessageDialog msg = new MessageDialog("There is no cond
    ition " + attribute11.Text + " !");
391.                     _ = msg.ShowAsync();
392.                     return;
393.                 }
394.             }
395.             if (content == ">")
396.             {
397.                 for (int i = 0; i < attributes[conditionnum1].get_values().
    Count; i++)
398.                 {
399.                     if (attributes[conditionnum1].get_type() == "char")
400.                     {
401.                         if (String.Compare(attributes[conditionnum1].get_va
    lues()[i], attribute12.Text) > 0)
402.                         {
403.                             for (int k = 0; k < selattributes.Count; k++)
404.                                 for (int j = 0; j < attributes.Count; j++)

405.                                     if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
406.                                     {
407.                                         string temp;
408.                                         temp = attributes[j].get_values()[i
    ];
409.                                         selattributes[k].insert_value(temp)
    ;
410.                                     }
411.
412.                         }
413.                     }
414.                     else
415.                     {
416.                         if (Convert.ToDouble(attributes[conditionnum1].get_
    values()[i]) > Convert.ToDouble(attribute12.Text))
```

```
417.                                 for (int k = 0; k < selattributes.Count; k++)
418.                                     for (int j = 0; j < attributes.Count; j++)

419.                                         if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
420.                                         {
421.                                             string temp;
422.                                             temp = attributes[j].get_values()[i
    ];
423.                                             selattributes[k].insert_value(temp)
    ;
424.                                         }
425.                         }
426.                     }
427.                 }
428.                 else if (content == "<")
429.                 {
430.                     for (int i = 0; i < attributes[conditionnum1].get_values().
    Count; i++)
431.                     {
432.                         if (attributes[conditionnum1].get_type() == "char")
433.                         {
434.                             if (String.Compare(attributes[conditionnum1].get_va
    lues()[i], attribute12.Text) < 0)
435.                             {
436.                                 for (int k = 0; k < selattributes.Count; k++)
437.                                     for (int j = 0; j < attributes.Count; j++)

438.                                         if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
439.                                         {
440.                                             string temp;
441.                                             temp = attributes[j].get_values()[i
    ];
442.                                             selattributes[k].insert_value(temp)
    ;
443.                                         }
444.
445.                             }
446.                         }
447.                         else
448.                             if (Convert.ToDouble(attributes[conditionnum1].get_
    values()[i]) < Convert.ToDouble(attribute12.Text))
449.                             for (int k = 0; k < selattributes.Count; k++)
```

```
450.                                 for (int j = 0; j < attributes.Count; j++)
451.                                     if (selattributes[k].get_name() == attribut
    es[j].get_name())
452.                                     {
453.                                         string temp;
454.                                         temp = attributes[j].get_values()[i];
455.                                         selattributes[k].insert_value(temp);
456.                                     }
457.                 }
458.             }
459.             else if (content == "<>")
460.             {
461.                 for (int i = 0; i < attributes[conditionnum1].get_values().
    Count; i++)
462.                 {
463.                     if (attributes[conditionnum1].get_type() == "char")
464.                         if (String.Compare(attributes[conditionnum1].get_va
    lues()[i], attribute12.Text) != 0)
465.                         {
466.                             for (int k = 0; k < selattributes.Count; k++)
467.                                 for (int j = 0; j < attributes.Count; j++)

468.                                     if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
469.                                     {
470.                                         string temp;
471.                                         temp = attributes[j].get_values()[i
    ];
472.                                         selattributes[k].insert_value(temp)
    ;
473.                                     }
474.
475.                         }
476.                         else;
477.                     else
478.                         if (Convert.ToDouble(attributes[conditionnum1].get_
    values()[i]) != Convert.ToDouble(attribute12.Text))
479.                         for (int k = 0; k < selattributes.Count; k++)
480.                             for (int j = 0; j < attributes.Count; j++)
481.                                 if (selattributes[k].get_name() == attribut
    es[j].get_name())
482.                                 {
483.                                     string temp;
484.                                     temp = attributes[j].get_values()[i];
```

```
485.                                     selattributes[k].insert_value(temp);
486.                                 }
487.                 }
488.             }
489.             else if (content == "=")
490.             {
491.                 for (int i = 0; i < attributes[conditionnum1].get_values().
    Count; i++)
492.                 {
493.                     if (attributes[conditionnum1].get_type() == "char")
494.                         if (String.Compare(attributes[conditionnum1].get_va
    lues()[i], attribute12.Text) == 0)
495.                         {
496.                             for (int k = 0; k < selattributes.Count; k++)
497.                                 for (int j = 0; j < attributes.Count; j++)

498.                                     if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
499.                                     {
500.                                         string temp;
501.                                         temp = attributes[j].get_values()[i
    ];
502.                                         selattributes[k].insert_value(temp)
    ;
503.                                     }
504.
505.                         }
506.                         else;
507.                     else
508.                         if (Convert.ToDouble(attributes[conditionnum1].get_
    values()[i]) == Convert.ToDouble(attribute12.Text))
509.                         for (int k = 0; k < selattributes.Count; k++)
510.                             for (int j = 0; j < attributes.Count; j++)
511.                                 if (selattributes[k].get_name() == attribut
    es[j].get_name())
512.                                 {
513.                                     string temp;
514.                                     temp = attributes[j].get_values()[i];
515.                                     selattributes[k].insert_value(temp);
516.                                 }
517.                 }
518.             }
519.             else if (content == ">=")
520.             {
```

```
521.                 for (int i = 0; i < attributes[conditionnum1].get_values().
    Count; i++)
522.                 {
523.                     if (attributes[conditionnum1].get_type() == "char")
524.                         if (String.Compare(attributes[conditionnum1].get_va
    lues()[i], attribute12.Text) >= 0)
525.                         {
526.                             for (int k = 0; k < selattributes.Count; k++)
527.                                 for (int j = 0; j < attributes.Count; j++)

528.                                     if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
529.                                     {
530.                                         string temp;
531.                                         temp = attributes[j].get_values()[i
    ];
532.                                         selattributes[k].insert_value(temp)
    ;
533.                                     }
534.
535.                         }
536.                         else;
537.                     else
538.                         if (Convert.ToDouble(attributes[conditionnum1].get_
    values()[i]) >= Convert.ToDouble(attribute12.Text))
539.                         for (int k = 0; k < selattributes.Count; k++)
540.                             for (int j = 0; j < attributes.Count; j++)
541.                                 if (selattributes[k].get_name() == attribut
    es[j].get_name())
542.                                 {
543.                                     string temp;
544.                                     temp = attributes[j].get_values()[i];
545.                                     selattributes[k].insert_value(temp);
546.                                 }
547.                 }
548.             }
549.             else if (content == "<=")
550.             {
551.                 for (int i = 0; i < attributes[conditionnum1].get_values().
    Count; i++)
552.                 {
553.                     if (attributes[conditionnum1].get_type() == "char")
554.                         if (String.Compare(attributes[conditionnum1].get_va
    lues()[i], attribute12.Text) <= 0)
```

```
555.                         {
556.                             for (int k = 0; k < selattributes.Count; k++)
557.                                 for (int j = 0; j < attributes.Count; j++)

558.                                     if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
559.                                     {
560.                                         string temp;
561.                                         temp = attributes[j].get_values()[i
    ];
562.                                         selattributes[k].insert_value(temp)
    ;
563.                                     }
564.
565.                         }
566.                         else;
567.                     else
568.                         if (Convert.ToDouble(attributes[conditionnum1].get_
    values()[i]) <= Convert.ToDouble(attribute12.Text))
569.                         for (int k = 0; k < selattributes.Count; k++)
570.                             for (int j = 0; j < attributes.Count; j++)
571.                                 if (selattributes[k].get_name() == attribut
    es[j].get_name())
572.                                 {
573.                                     string temp;
574.                                     temp = attributes[j].get_values()[i];
575.                                     selattributes[k].insert_value(temp);
576.                                 }
577.                 }
578.             }
579.             else;
580.
581.         }
582.
583.         public void select(string content, ref List<Attribute> selattribute
    s, List<Attribute> attributes, string attribute11, string attribute12)
584.         {
585.             int conditionnum1 = 0;
586.
587.             for (int k = 0; k < attributes.Count; k++)
588.             {
589.                 if (attribute11 == attributes[k].get_name())
590.                 {
591.                     conditionnum1 = k;
```

```
592.                     break;
593.                 }
594.                 if (k == attributes.Count)
595.                 {
596.                     MessageDialog msg = new MessageDialog("There is no cond
    ition " + attribute11 + " !");
597.                     _ = msg.ShowAsync();
598.                     return;
599.                 }
600.             }
601.
602.             if (content == ">")
603.             {
604.                 for (int i = 0; i < attributes[conditionnum1].get_values().
    Count; i++)
605.                 {
606.                     if (attributes[conditionnum1].get_type() == "char")
607.                         if (String.Compare(attributes[conditionnum1].get_va
    lues()[i], attribute12) > 0)
608.                         {
609.                             for (int k = 0; k < selattributes.Count; k++)
610.                                 for (int j = 0; j < attributes.Count; j++)

611.                                     if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
612.                                     {
613.                                         string temp;
614.                                         temp = attributes[j].get_values()[i
    ];
615.                                         selattributes[k].insert_value(temp)
    ;
616.                                     }
617.
618.                         }
619.                         else;
620.                     else
621.                         if (Convert.ToDouble(attributes[conditionnum1].get_
    values()[i]) > Convert.ToDouble(attribute12))
622.                         for (int k = 0; k < selattributes.Count; k++)
623.                             for (int j = 0; j < attributes.Count; j++)
624.                                 if (selattributes[k].get_name() == attribut
    es[j].get_name())
625.                                 {
626.                                     string temp;
```

```
627.                                     temp = attributes[j].get_values()[i];
628.                                     selattributes[k].insert_value(temp);
629.                                 }
630.                                 else;
631.
632.                 }
633.             }
634.             else if (content == "<")
635.             {
636.                 for (int i = 0; i < attributes[conditionnum1].get_values().
    Count; i++)
637.                 {
638.                     if (attributes[conditionnum1].get_type() == "char")
639.                         if (String.Compare(attributes[conditionnum1].get_va
    lues()[i], attribute12) < 0)
640.                         {
641.                             for (int k = 0; k < selattributes.Count; k++)
642.                                 for (int j = 0; j < attributes.Count; j++)

643.                                     if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
644.                                     {
645.                                         string temp;
646.                                         temp = attributes[j].get_values()[i
    ];
647.                                         selattributes[k].insert_value(temp)
    ;
648.                                     }
649.
650.                         }
651.                         else;
652.                     else
653.                         if (Convert.ToDouble(attributes[conditionnum1].get_
    values()[i]) < Convert.ToDouble(attribute12))
654.                         for (int k = 0; k < selattributes.Count; k++)
655.                             for (int j = 0; j < attributes.Count; j++)
656.                                 if (selattributes[k].get_name() == attribut
    es[j].get_name())
657.                                 {
658.                                     string temp;
659.                                     temp = attributes[j].get_values()[i];
660.                                     selattributes[k].insert_value(temp);
661.                                 }
662.                 }
```

```
663.             }
664.             else if (content == "<>")
665.             {
666.                 for (int i = 0; i < attributes[conditionnum1].get_values().
    Count; i++)
667.                 {
668.                     if (attributes[conditionnum1].get_type() == "char")
669.                         if (String.Compare(attributes[conditionnum1].get_va
    lues()[i], attribute12) != 0)
670.                         {
671.                             for (int k = 0; k < selattributes.Count; k++)
672.                                 for (int j = 0; j < attributes.Count; j++)

673.                                     if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
674.                                     {
675.                                         string temp;
676.                                         temp = attributes[j].get_values()[i
    ];
677.                                         selattributes[k].insert_value(temp)
    ;
678.                                     }
679.
680.                         }
681.                         else;
682.                     else
683.                         if (Convert.ToDouble(attributes[conditionnum1].get_
    values()[i]) != Convert.ToDouble(attribute12))
684.                         for (int k = 0; k < selattributes.Count; k++)
685.                             for (int j = 0; j < attributes.Count; j++)
686.                                 if (selattributes[k].get_name() == attribut
    es[j].get_name())
687.                                 {
688.                                     string temp;
689.                                     temp = attributes[j].get_values()[i];
690.                                     selattributes[k].insert_value(temp);
691.                                 }
692.                 }
693.             }
694.             else if (content == "=")
695.             {
696.                 for (int i = 0; i < attributes[conditionnum1].get_values().
    Count; i++)
697.                 {
```

```
698.                     if (attributes[conditionnum1].get_type() == "char")
699.                         if (String.Compare(attributes[conditionnum1].get_va
    lues()[i], attribute12) == 0)
700.                         {
701.                             for (int k = 0; k < selattributes.Count; k++)
702.                                 for (int j = 0; j < attributes.Count; j++)

703.                                     if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
704.                                     {
705.                                         string temp;
706.                                         temp = attributes[j].get_values()[i
    ];
707.                                         selattributes[k].insert_value(temp)
    ;
708.                                     }
709.
710.                         }
711.                         else;
712.                     else
713.                         if (Convert.ToDouble(attributes[conditionnum1].get_
    values()[i]) == Convert.ToDouble(attribute12))
714.                         for (int k = 0; k < selattributes.Count; k++)
715.                             for (int j = 0; j < attributes.Count; j++)
716.                                 if (selattributes[k].get_name() == attribut
    es[j].get_name())
717.                                 {
718.                                     string temp;
719.                                     temp = attributes[j].get_values()[i];
720.                                     selattributes[k].insert_value(temp);
721.                                 }
722.                 }
723.             }
724.             else if (content == ">=")
725.             {
726.                 for (int i = 0; i < attributes[conditionnum1].get_values().
    Count; i++)
727.                 {
728.                     if (attributes[conditionnum1].get_type() == "char")
729.                         if (String.Compare(attributes[conditionnum1].get_va
    lues()[i], attribute12) >= 0)
730.                         {
731.                             for (int k = 0; k < selattributes.Count; k++)
```

```
732.                                 for (int j = 0; j < attributes.Count; j++)
733.                                     if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
734.                                     {
735.                                         string temp;
736.                                         temp = attributes[j].get_values()[i
    ];
737.                                         selattributes[k].insert_value(temp)
    ;
738.                                     }
739.
740.                             }
741.                             else;
742.                         else
743.                             if (Convert.ToDouble(attributes[conditionnum1].get_
    values()[i]) >= Convert.ToDouble(attribute12))
744.                             for (int k = 0; k < selattributes.Count; k++)
745.                                 for (int j = 0; j < attributes.Count; j++)
746.                                     if (selattributes[k].get_name() == attribut
    es[j].get_name())
747.                                     {
748.                                         string temp;
749.                                         temp = attributes[j].get_values()[i];
750.                                         selattributes[k].insert_value(temp);
751.                                     }
752.                     }
753.                 }
754.                 else if (content == "<=")
755.                 {
756.                     for (int i = 0; i < attributes[conditionnum1].get_values().
    Count; i++)
757.                     {
758.                         if (attributes[conditionnum1].get_type() == "char")
759.                             if (String.Compare(attributes[conditionnum1].get_va
    lues()[i], attribute12) <= 0)
760.                             {
761.                                 for (int k = 0; k < selattributes.Count; k++)
762.                                     for (int j = 0; j < attributes.Count; j++)
763.                                         if (selattributes[k].get_name() == attr
    ibutes[j].get_name())
764.                                         {
765.                                             string temp;
```

```
766.                                        temp = attributes[j].get_values()[i
    ];
767.                                        selattributes[k].insert_value(temp)
    ;
768.                                    }
769.
770.                            }
771.                            else;
772.                        else
773.                            if (Convert.ToDouble(attributes[conditionnum1].get_
    values()[i]) <= Convert.ToDouble(attribute12))
774.                            for (int k = 0; k < selattributes.Count; k++)
775.                                for (int j = 0; j < attributes.Count; j++)
776.                                    if (selattributes[k].get_name() == attribut
    es[j].get_name())
777.                                    {
778.                                        string temp;
779.                                        temp = attributes[j].get_values()[i];
780.                                        selattributes[k].insert_value(temp);
781.                                    }
782.                    }
783.                }
784.                else;
785.
786.            }
787.
788.            private void back(object sender, RoutedEventArgs e)
789.            {
790.                result.Visibility = Visibility.Collapsed;
791.                /* write your codes behind */
792.                this.InitializeComponent();
793.            }
794.
795.            private void Show_Detail(List<Attribute> selattributes)
796.            {
797.                try
798.                {
799.                    float width = 1200 / selattributes.Count - 10;
800.
801.                    for (int i = 0; i < selattributes.Count; i++)
802.                    {
803.                        GridViewHeaderItem temp = new GridViewHeaderItem();
804.                        temp.Content = selattributes[i].get_name();
805.                        temp.Width = width;
```

```
806.
807.                     query_result.Items.Add(temp);
808.                 }
809.
810.                 for (int i = 0; i < selattributes[0].get_values().Count; i+
    +)
811.                 {
812.                     for (int k = 0; k < selattributes.Count; k++)
813.                     {
814.                         GridViewItem temp = new GridViewItem();
815.                         temp.Content = selattributes[k].get_values()[i];
816.                         temp.Width = width;
817.
818.                         query_result.Items.Add(temp);
819.                     }
820.                 }
821.
822.
823.                 MessageDialog msg = new MessageDialog("Query succeed!");
824.                 _ = msg.ShowAsync();
825.             }
826.             catch (Exception)
827.             {
828.                 MessageDialog msg = new MessageDialog("Something Wrong happ
    ened! Check again.");
829.                 _ = msg.ShowAsync();
830.             }
831.         }
832.
833.         public string findcontent(string code)
834.         {
835.             if (code.IndexOf(">=") != -1)
836.             {
837.                 return ">=";
838.             }
839.             else if (code.IndexOf("<=") != -1)
840.             {
841.                 return "<=";
842.             }
843.             else if (code.IndexOf("<>") != -1)
844.             {
845.                 return "<>";
846.             }
847.             else if (code.IndexOf("=") != -1)
```

```
848.             {
849.                 return "=";
850.             }
851.             else if (code.IndexOf(">") != -1)
852.             {
853.                 return ">";
854.             }
855.             else if (code.IndexOf("<") != -1)
856.             {
857.                 return "<";
858.             }
859.             else
860.                 return null;
861.         }
862.
863.         private async void writedata(object sender, RoutedEventArgs e)
864.         {
865.             try
866.             {
867.                 string data = "";
868.
869.                 for (int i = 0; i < (Application.Current as App).tables.Cou
    nt; i++)
870.                 {
871.                     Table element = (Application.Current as App).tables[i];

872.                     data += element.get_name() + "\n";
873.                     foreach (Attribute temp in element.get_attribute())
874.                     {
875.                         data += temp.get_name() + " " + temp.get_type() + "
     " + temp.get_primary().ToString();
876.                         data += "\n";
877.                     }
878.
879.                     data += (Application.Current as App).tables[i].get_reco
    rd_number().ToString() + ((Application.Current as App).tables[i].get_record_
    number() == 0 ? "" : "\n");
880.
881.                     for (int j = 0; j < element.get_record_number(); j++)
882.                     {
883.                         for (int k = 0; k < element.get_attribute().Count;
    k++)
884.                         {
```

```
885.                             data += element.get_attribute()[k].get_values()
    [j] + (k == element.get_attribute().Count - 1 ? "" : " ");
886.                         }
887.                         data += ((i == (Application.Current as App).tables.
    Count - 1 && j == element.get_record_number() - 1) ? "" : "\n");
888.                     }
889.
890.                     data += (i == (Application.Current as App).tables.Count
     - 1 ? "" : "\n");
891.                 }
892.
893.
894.                 FileSavePicker fp = new FileSavePicker();
895.                 var filedb = new[] { ".txt" };
896.                 fp.FileTypeChoices.Add("Plain Text", new List<string>() { "
    .txt" });
897.                 fp.SuggestedFileName = "table information" + DateTime.Now.D
    ay + "-" + DateTime.Now.Month + "-" + DateTime.Now.Year;
898.                 StorageFile sf = await fp.PickSaveFileAsync();
899.
900.                 if (sf != null)
901.                 {
902.                     using (StorageStreamTransaction transaction = await sf.
    OpenTransactedWriteAsync())
903.                     {
904.                         using (DataWriter dataWriter = new DataWriter(trans
    action.Stream))
905.                         {
906.                             dataWriter.WriteString(data);
907.                             transaction.Stream.Size = await dataWriter.Stor
    eAsync();
908.                             await transaction.CommitAsync();
909.                         }
910.                     }
911.                 }
912.
913.                 MessageDialog msg = new MessageDialog("Operation succeed!")
    ;
914.                 _ = msg.ShowAsync();
915.             }
916.             catch (Exception)
917.             {
918.                 MessageDialog msg = new MessageDialog("Something wrong happ
    ened.");
```

```
                _ = msg.ShowAsync();
            }

        }
    }
}
```

Querys.xaml

```
<Page
    x:Class="MiniSQL.Querys"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:MiniSQL"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>
        <NavigationView x:Name="Navigation" IsBackEnabled="True" IsSettingsVisible="True" IsTabStop="False" BackRequested="goback">
            <NavigationView.MenuItems>
                <NavigationViewItemHeader Content="Functions"/>
                <NavigationViewItem Icon="Home" Content="Main Page" Tapped="main"/>
                <NavigationViewItemSeparator/>

                <NavigationViewItem Icon="Import" Content="Table Operations" Tapped="table"></NavigationViewItem>
                <NavigationViewItem Icon="List" Content="Index Operations" Tapped="index"/>
                <NavigationViewItem Icon="Find" Content="Query" Tapped="query"></NavigationViewItem>
                <NavigationViewItem Icon="AllApps" Content="Insert" Tapped="insert"/>
                <NavigationViewItem Icon="Delete" Content="Delete" Tapped="delete"></NavigationViewItem>
                <NavigationViewItem Icon="Next" Content="Export" Tapped="writedata"/>
                <NavigationViewItem Icon="Undo" Content="Exit" Tapped="exit" />
            </NavigationView.MenuItems>

            <Grid>
```

```
28.                 <Image Source="/Images/zjulib.jpg" Width="421.75" Height="11
    7.25" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0, 50, 0,
     0"></Image>
29.                 <Image Source="/Images/under.png" Width="557" Height="112" H
    orizontalAlignment="Center" VerticalAlignment="Top" Margin="0, 900, 0, 0"></
    Image>
30.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Margin="0, 200, 0, 0" FontSize="50" Width="500" Height="70" TextAlignmen
    t="Center">Query Operations</TextBlock>
31.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="C
    enter" Margin="0, 0, 0, 0" FontSize="50">OR</TextBlock>
32.
33.                 <TextBlock Width="400" Height="40" TextAlignment="Center" Fo
    ntSize="30" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="-
    900, 300, 0,0">Input your SQL Code Here</TextBlock>
34.                 <TextBox x:Name="source" Width="400" Height="400" VerticalAl
    ignment="Top" HorizontalAlignment="Center" Margin="-
    900, 350, 0, 0" FontSize="20" AcceptsReturn="True" FontFamily="Consolas"></T
    extBox>
35.                 <Button Width="200" Height="60" FontSize="30" VerticalAlignm
    ent="Top" HorizontalAlignment="Center" Margin="-
    900, 800, 0, 0" Click="execute">Execute</Button>
36.
37.                 <TextBlock Width="500" Height="40" TextAlignment="Center" Fo
    ntSize="30" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="900
    , 300, 0,0">Quick Operations Using Templates</TextBlock>
38.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 400, 0, 0"
    >Table Name:</TextBlock>
39.                 <TextBox x:Name="table_name" PlaceholderText="Enter table na
    me you want to select from" HorizontalAlignment="Center" VerticalAlignment="
    Top" Margin="1000, 400, 0, 0" Width="400" Height="40" FontSize="20"></TextBo
    x>
40.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 500, 0, 0"
    >Attribute Name:</TextBlock>
41.                 <TextBox x:Name="attribute_name" PlaceholderText="Enter attr
    ibutes'name one with sigle line" HorizontalAlignment="Center" VerticalAlignm
    ent="Top" Margin="1000, 500, 0, 0" Width="400" Height="120" FontSize="20" Ac
    ceptsReturn="True"></TextBox>
42.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 650, 0, 0"
    >Filter Condition:</TextBlock>
```

```
43.                 <TextBox x:Name="attribute11" PlaceholderText="attribute" Fo
    ntSize="20" Width="150" Height="40" HorizontalAlignment="Center" VerticalAli
    gnment="Top" Margin="750, 650, 0, 0"></TextBox>
44.                 <ComboBox x:Name="operator1" Header="Operator" Width="100" H
    orizontalAlignment="Center" VerticalAlignment="Top" Margin="1050, 630, 0, 0"
    >
45.                     <ComboBoxItem>=</ComboBoxItem>
46.                     <ComboBoxItem><></ComboBoxItem>
47.                     <ComboBoxItem><</ComboBoxItem>
48.                     <ComboBoxItem>></ComboBoxItem>
49.                     <ComboBoxItem><=</ComboBoxItem>
50.                     <ComboBoxItem>>=</ComboBoxItem>
51.                 </ComboBox>
52.                 <TextBox x:Name="attribute12" PlaceholderText="attribute" Fo
    ntSize="20" Width="150" Height="40" HorizontalAlignment="Center" VerticalAli
    gnment="Top" Margin="1350, 650, 0, 0"></TextBox>
53.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 700, 0, 0"
    >Logic Condition:</TextBlock>
54.                 <RadioButton x:Name="and" HorizontalAlignment="Center" Verti
    calAlignment="Top" Margin="850, 700, 0, 0">AND</RadioButton>
55.                 <RadioButton x:Name="or" HorizontalAlignment="Center" Vertic
    alAlignment="Top" Margin="1050, 700, 0, 0">OR</RadioButton>
56.
57.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 750, 0, 0"
    >Filter Condition:</TextBlock>
58.                 <TextBox x:Name="attribute21" PlaceholderText="attribute" Fo
    ntSize="20" Width="150" Height="40" HorizontalAlignment="Center" VerticalAli
    gnment="Top" Margin="750, 750, 0, 0"></TextBox>
59.                 <ComboBox x:Name="operator2" Header="Operator" Width="100" H
    orizontalAlignment="Center" VerticalAlignment="Top" Margin="1050, 730, 0, 0"
    >
60.                     <ComboBoxItem>=</ComboBoxItem>
61.                     <ComboBoxItem><></ComboBoxItem>
62.                     <ComboBoxItem><</ComboBoxItem>
63.                     <ComboBoxItem>></ComboBoxItem>
64.                     <ComboBoxItem><=</ComboBoxItem>
65.                     <ComboBoxItem>>=</ComboBoxItem>
66.                 </ComboBox>
67.                 <TextBox x:Name="attribute22" PlaceholderText="attribute" Fo
    ntSize="20" Width="150" Height="40" HorizontalAlignment="Center" VerticalAli
    gnment="Top" Margin="1350, 750, 0, 0"></TextBox>
68.
```

```
69.                 <Grid x:Name="result" Background="{ThemeResource Application
    PageBackgroundThemeBrush}" Visibility="Collapsed">
70.                     <Image Source="/Images/zjulib.jpg" Width="421.75" Height
    ="117.25" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0, 50
    , 0, 0"></Image>
71.                     <Image Source="/Images/under.png" Width="557" Height="11
    2" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0, 900, 0, 0
    "></Image>
72.                     <TextBlock HorizontalAlignment="Center" VerticalAlignmen
    t="Top" Margin="0, 200, 0, 0" FontSize="50" Width="500" Height="70" TextAlig
    nment="Center">Query Result</TextBlock>
73.
74.                     <GridView x:Name="query_result" HorizontalAlignment="Cen
    ter" VerticalAlignment="Top" Width="1200" Height="450" Margin="0, 300, 0, 0"
     BorderBrush="Red" Foreground="#FFE62525" Background="#FF5D5D5D">
75.
76.                     </GridView>
77.
78.                     <Button HorizontalAlignment="Center" VerticalAlignment="
    Top" Margin="0, 800, 0, 0" Width="100" Height="40" FontSize="20" Click="back
    ">Return</Button>
79.
80.                 </Grid>
81.
82.             </Grid>
83.
84.             <NavigationView.PaneFooter>
85.                 <HyperlinkButton x:Name="MoreInfoBtn" Content="Email Me" Mar
    gin="12,0" NavigateUri="mailto:ZJU.SLM@gmail.com"/>
86.             </NavigationView.PaneFooter>
87.         </NavigationView>
88.     </Grid>
89. </Page>
```

Insert_Operations.xaml.cs

```
1. using System;
2. using System.Collections.Generic;
3. using Windows.UI.Popups;
4. using Windows.UI.Xaml;
5. using Windows.UI.Xaml.Controls;
6. using Windows.UI.Xaml.Input;
7. using Windows.Storage.Streams;
8. using Windows.Storage;
9. using Windows.Storage.Pickers;
```

```
10.
11. // The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238
12.
13. namespace MiniSQL
14. {
15.     /// <summary>
16.     /// An empty page that can be used on its own or navigated to within a Frame.
17.     /// </summary>
18.     public sealed partial class Insert_Operations : Page
19.     {
20.         public Insert_Operations()
21.         {
22.             this.InitializeComponent();
23.         }
24.
25.         private void goback(NavigationView sender, NavigationViewBackRequestedEventArgs args)
26.         {
27.             Frame.GoBack();
28.         }
29.
30.         private void main(object sender, TappedRoutedEventArgs e)
31.         {
32.             Frame.Navigate(typeof(MainPage), "");
33.         }
34.
35.         private void query(object sender, TappedRoutedEventArgs e)
36.         {
37.             Frame.Navigate(typeof(Querys), "");
38.         }
39.
40.         private void exit(object sender, TappedRoutedEventArgs e)
41.         {
42.             (Application.Current as App).Exit();
43.         }
44.
45.         private void table(object sender, TappedRoutedEventArgs e)
46.         {
47.             Frame.Navigate(typeof(Table_Operations), "");
48.         }
49.
50.         private void index(object sender, TappedRoutedEventArgs e)
```

```
51.         {
52.             Frame.Navigate(typeof(Index_Operations), "");
53.         }
54.
55.         private void insert(object sender, TappedRoutedEventArgs e)
56.         {
57.             Frame.Navigate(typeof(Insert_Operations), "");
58.         }
59.
60.         private void delete(object sender, TappedRoutedEventArgs e)
61.         {
62.             Frame.Navigate(typeof(Delete_Operations), "");
63.         }
64.
65.         private void execute(object sender, RoutedEventArgs e)
66.         {
67.             /* write your codes behind */
68.             try
69.             {
70.                 if (table_name.Text == "")
71.                 {
72.                     string name = source.Text.Substring(12, source.Text.Inde
    xOf('(') - 13);
73.                     List<string> single_values = new List<string>(source.Tex
    t.Substring(source.Text.IndexOf('(') + 2, source.Text.IndexOf(')') - 20).Spl
    it(", "));
74.                     int table_index = -1;
75.
76.                     for (int i = 0; i < (Application.Current as App).tables.
    Count; i++)
77.                     {
78.                         if ((Application.Current as App).tables[i].get_name(
    ) == name)
79.                         {
80.                             table_index = i;
81.                             break;
82.                         }
83.                     }
84.
85.                     if (table_index >= 0)
86.                     {
87.                         for (int i = 0; i < single_values.Count; i++)
88.                         {
```

```
89.                             if (single_values[i].IndexOf('.') != -
    1 && (Application.Current as App).tables[table_index].get_attribute()[i].get
    _type() == "int")
90.                             {
91.                                 MessageDialog msge = new MessageDialog("Inpu
    t type does not match!");
92.                                 _ = msge.ShowAsync();
93.                                 return;
94.                             }
95.
96.
97.                             if ((Application.Current as App).tables[table_in
    dex].get_attribute()[i].get_type().IndexOf("char") != -1)
98.                             {
99.                                 string type = (Application.Current as App).t
    ables[table_index].get_attribute()[i].get_type();
100.                                int len = int.Parse(type.Substring(5, type.
    IndexOf(')') - type.IndexOf('(') - 1));
101.                                if (single_values[i].Length - 2 > len)
102.                                {
103.                                    MessageDialog msge = new MessageDialog(
    "char's length is too large!");
104.                                    _ = msge.ShowAsync();
105.                                    return;
106.                                }
107.
108.                                single_values[i] = single_values[i].Substri
    ng(1, single_values[i].Length - 2);
109.                            }
110.                        }
111.
112.                        if ((Application.Current as App).tables[table_index
    ].insert_record(single_values))
113.                        {
114.                            MessageDialog msg = new MessageDialog("Operatio
    n Succeed!");
115.                            _ = msg.ShowAsync();
116.                        }
117.                    }
118.                    else
119.                    {
120.                        MessageDialog msg = new MessageDialog("The table do
    es not exist!");
121.                        _ = msg.ShowAsync();
```

```
122.                         return;
123.                     }
124.                 }
125.                 else
126.                 {
127.                     int table_index = -1;
128.
129.                     for (int i = 0; i < (Application.Current as App).tables
    .Count; i++)
130.                     {
131.                         if ((Application.Current as App).tables[i].get_name
    () == table_name.Text)
132.                         {
133.                             table_index = i;
134.                             break;
135.                         }
136.                     }
137.
138.                     if (table_index >= 0)
139.                     {
140.                         List<string> single_values = new List<string>(value
    s.Text.Split('\r'));
141.
142.                         for (int i = 0; i < single_values.Count; i++)
143.                         {
144.                             if (single_values[i].IndexOf('.') != -
    1 && (Application.Current as App).tables[table_index].get_attribute()[i].get
    _type() == "int")
145.                             {
146.                                 MessageDialog msge = new MessageDialog("Inp
    ut type does not match!");
147.                                 _ = msge.ShowAsync();
148.                                 return;
149.                             }
150.
151.
152.                             if ((Application.Current as App).tables[table_i
    ndex].get_attribute()[i].get_type().IndexOf("char") != -1)
153.                             {
154.                                 string type = (Application.Current as App).
    tables[table_index].get_attribute()[i].get_type();
155.                                 int len = int.Parse(type.Substring(5, type.
    IndexOf(')') - type.IndexOf('(') - 1));
156.                                 if (single_values[i].Length > len)
```

```
157.                                 {
158.                                     MessageDialog msge = new MessageDialog(
    "char's length is too large!");
159.                                     _ = msge.ShowAsync();
160.                                     return;
161.                                 }
162.                             }
163.                         }
164.
165.                         if ((Application.Current as App).tables[table_index
    ].insert_record(single_values))
166.                         {
167.                             MessageDialog msg = new MessageDialog("Operatio
    n Succeed!");
168.                             _ = msg.ShowAsync();
169.                         }
170.                     }
171.                     else
172.                     {
173.                         MessageDialog msg = new MessageDialog("The table do
    es not exist!");
174.                         _ = msg.ShowAsync();
175.                         return;
176.                     }
177.                 }
178.             }
179.             catch (ArgumentException)
180.             {
181.                 MessageDialog msg = new MessageDialog("Check your SQL code
    carefully!");
182.                 _ = msg.ShowAsync();
183.             }
184.             catch (NullReferenceException)
185.             {
186.                 MessageDialog msg = new MessageDialog("You haven't input an
    y attribute's name or type.");
187.                 _ = msg.ShowAsync();
188.             }
189.             catch (Exception)
190.             {
191.                 MessageDialog msg = new MessageDialog("Something else wrong
     happened!");
192.                 _ = msg.ShowAsync();
193.             }
```

```
194.         }
195.
196.         private async void writedata(object sender, RoutedEventArgs e)
197.         {
198.             try
199.             {
200.                 string data = "";
201.
202.                 for (int i = 0; i < (Application.Current as App).tables.Cou
    nt; i++)
203.                 {
204.                     Table element = (Application.Current as App).tables[i];
205.                     data += element.get_name() + "\n";
206.                     foreach (Attribute temp in element.get_attribute())
207.                     {
208.                         data += temp.get_name() + " " + temp.get_type() + "
     " + temp.get_primary().ToString();
209.                         data += "\n";
210.                     }
211.
212.                     data += (Application.Current as App).tables[i].get_reco
    rd_number().ToString() + ((Application.Current as App).tables[i].get_record_
    number() == 0 ? "" : "\n");
213.
214.                     for (int j = 0; j < element.get_record_number(); j++)
215.                     {
216.                         for (int k = 0; k < element.get_attribute().Count;
    k++)
217.                         {
218.                             data += element.get_attribute()[k].get_values()
    [j] + (k == element.get_attribute().Count - 1 ? "" : " ");
219.                         }
220.                         data += ((i == (Application.Current as App).tables.
    Count - 1 && j == element.get_record_number() - 1) ? "" : "\n");
221.                     }
222.
223.                     data += (i == (Application.Current as App).tables.Count
     - 1 ? "" : "\n");
224.                 }
225.
226.
227.                 FileSavePicker fp = new FileSavePicker();
228.                 var filedb = new[] { ".txt" };
```

```
229.                 fp.FileTypeChoices.Add("Plain Text", new List<string>() { "
    .txt" });
230.                 fp.SuggestedFileName = "table information" + DateTime.Now.D
    ay + "-" + DateTime.Now.Month + "-" + DateTime.Now.Year;
231.                 StorageFile sf = await fp.PickSaveFileAsync();
232.
233.                 if (sf != null)
234.                 {
235.                     using (StorageStreamTransaction transaction = await sf.
    OpenTransactedWriteAsync())
236.                     {
237.                         using (DataWriter dataWriter = new DataWriter(trans
    action.Stream))
238.                         {
239.                             dataWriter.WriteString(data);
240.                             transaction.Stream.Size = await dataWriter.Stor
    eAsync();
241.                             await transaction.CommitAsync();
242.                         }
243.                     }
244.                 }
245.
246.                 MessageDialog msg = new MessageDialog("Operation succeed!")
    ;
247.                 _ = msg.ShowAsync();
248.             }
249.             catch (Exception)
250.             {
251.                 MessageDialog msg = new MessageDialog("Something wrong happ
    ened.");
252.                 _ = msg.ShowAsync();
253.             }
254.
255.         }
256.     }
257. }
```

Insert_Operations.xaml

```
1.  <Page
2.      x:Class="MiniSQL.Insert_Operations"
3.      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4.      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5.      xmlns:local="using:MiniSQL"
6.      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

```
7.      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8.      mc:Ignorable="d"
9.      Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
10.
11.     <Grid>
12.         <NavigationView x:Name="Navigation" IsBackEnabled="True" IsSettingsV
    isible="True" IsTabStop="False" BackRequested="goback">
13.             <NavigationView.MenuItems>
14.                 <NavigationViewItemHeader Content="Functions"/>
15.                 <NavigationViewItem Icon="Home" Content="Main Page" Tapped="
    main"/>
16.                 <NavigationViewItemSeparator/>
17.
18.                 <NavigationViewItem Icon="Import" Content="Table Operations"
     Tapped="table"></NavigationViewItem>
19.                 <NavigationViewItem Icon="List" Content="Index Operations" T
    apped="index"/>
20.                 <NavigationViewItem Icon="Find" Content="Query" Tapped="quer
    y"></NavigationViewItem>
21.                 <NavigationViewItem Icon="AllApps" Content="Insert" Tapped="
    insert"/>
22.                 <NavigationViewItem Icon="Delete" Content="Delete" Tapped="d
    elete"></NavigationViewItem>
23.                 <NavigationViewItem Icon="Next" Content="Export" Tapped="wri
    tedata"/>
24.                 <NavigationViewItem Icon="Undo" Content="Exit" Tapped="exit"
    />
25.             </NavigationView.MenuItems>
26.
27.             <Grid>
28.                 <Image Source="/Images/zjulib.jpg" Width="421.75" Height="11
    7.25" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0, 50, 0,
     0"></Image>
29.                 <Image Source="/Images/under.png" Width="557" Height="112" H
    orizontalAlignment="Center" VerticalAlignment="Top" Margin="0, 900, 0, 0"></
    Image>
30.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Margin="0, 200, 0, 0" FontSize="50" Width="500" Height="70" TextAlignmen
    t="Center">Insert Operations</TextBlock>
31.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="C
    enter" Margin="0, 0, 0, 0" FontSize="50">OR</TextBlock>
32.
```

```
33.                 <TextBlock Width="400" Height="40" TextAlignment="Center" Fo
    ntSize="30" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="-
    900, 300, 0,0">Input your SQL Code Here</TextBlock>
34.                 <TextBox x:Name="source" Width="400" Height="400" VerticalAl
    ignment="Top" HorizontalAlignment="Center" Margin="-
    900, 350, 0, 0" FontSize="20" FontFamily="Consolas" AcceptsReturn="True"></T
    extBox>
35.                 <Button Width="200" Height="60" FontSize="30" VerticalAlignm
    ent="Top" HorizontalAlignment="Center" Margin="-
    900, 800, 0, 0" Click="execute">Execute</Button>
36.
37.                 <TextBlock Width="500" Height="40" TextAlignment="Center" Fo
    ntSize="30" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="900
    , 300, 0,0">Quick Operations Using Templates</TextBlock>
38.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" FontSize="20" Width="150" TextAlignment="Center" Margin="300, 350, 0, 0"
    >Table Name:</TextBlock>
39.                 <TextBox x:Name="table_name" PlaceholderText="Enter your tab
    le name" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="900, 3
    50, 0, 0" Width="400" Height="40" FontSize="20"></TextBox>
40.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" FontSize="20" Width="150" TextAlignment="Center" Margin="300, 450, 0, 0"
    >Values:</TextBlock>
41.                 <TextBox x:Name="values" PlaceholderText="Enter the correspo
    nding values line by line" HorizontalAlignment="Center" VerticalAlignment="T
    op" Width="400" Height="400" Margin="900, 450, 0, 0" FontSize="20" FontFamil
    y="Consolas" AcceptsReturn="True"></TextBox>
42.
43.             </Grid>
44.
45.             <NavigationView.PaneFooter>
46.                 <HyperlinkButton x:Name="MoreInfoBtn" Content="Email Me" Mar
    gin="12,0" NavigateUri="mailto:ZJU.SLM@gmail.com"/>
47.             </NavigationView.PaneFooter>
48.         </NavigationView>
49.     </Grid>
50. </Page>
```

Index_Operations.xaml.cs

```
1.  using BPlusTree;
2.  using System;
3.  using System.Collections.Generic;
4.  using Windows.UI.Popups;
5.  using Windows.UI.Xaml;
```

```
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Input;
using Windows.Storage.Streams;
using Windows.Storage;
using Windows.Storage.Pickers;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace MiniSQL
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class Index_Operations : Page
    {
        public Index_Operations()
        {
            this.InitializeComponent();
        }

        private void goback(NavigationView sender, NavigationViewBackRequestedEventArgs args)
        {
            Frame.GoBack();
        }

        private void main(object sender, TappedRoutedEventArgs e)
        {
            Frame.Navigate(typeof(MainPage), "");
        }

        private void query(object sender, TappedRoutedEventArgs e)
        {
            Frame.Navigate(typeof(Querys), "");
        }

        private void exit(object sender, TappedRoutedEventArgs e)
        {
            (Application.Current as App).Exit();
        }

        private void table(object sender, TappedRoutedEventArgs e)
```

```
47.         {
48.             Frame.Navigate(typeof(Table_Operations), "");
49.         }
50.
51.         private void index(object sender, TappedRoutedEventArgs e)
52.         {
53.             Frame.Navigate(typeof(Index_Operations), "");
54.         }
55.
56.         private void insert(object sender, TappedRoutedEventArgs e)
57.         {
58.             Frame.Navigate(typeof(Insert_Operations), "");
59.         }
60.
61.         private void delete(object sender, TappedRoutedEventArgs e)
62.         {
63.             Frame.Navigate(typeof(Delete_Operations), "");
64.         }
65.
66.         private void execute(object sender, RoutedEventArgs e)
67.         {
68.             /* write your codes behind */
69.             try
70.             {
71.                 string sqlcode = source.Text.ToLower();
72.
73.                 if (source.Text == "")
74.                 {
75.                     if (attribute_name.Text.Length == 0)
76.                     {
77.                         throw new NullReferenceException();
78.                     }
79.
80.                     List<Attribute> attributes = new List<Attribute>();
81.
82.                     for (int i = 0; i <= (Application.Current as App).tables
    .Count; i++)
83.                     {
84.                         if (i == (Application.Current as App).tables.Count)

85.                         {
86.                             MessageDialog msg = new MessageDialog("There is
    no this table!");
87.                             _ = msg.ShowAsync();
```

```
88.                             return;
89.                         }
90.                         if ((Application.Current as App).tables[i].get_name(
    ) == table_name.Text)
91.                         {
92.                             attributes = (Application.Current as App).tables
    [i].get_attribute();
93.                             break;
94.                         }
95.                     }
96.
97.                     string indexname = index_name.Text;
98.                     Attribute indexitem = new Attribute();
99.
100.                     for (int i = 0; i < attributes.Count; i++)
101.                     {
102.                         if (attribute_name.Text == attributes[i].get_name()
    )
103.                             indexitem = attributes[i];
104.                     }
105.
106.                     BPlusTree.BPlusTree pTree = new BPlusTree.BPlusTree();

107.                     pTree.ClearTree();
108.
109.                     for (int i = 0; i < indexitem.get_values().Count; i++)

110.                     {
111.                         pTree.Insert(indexitem.get_values()[i]);
112.                     }
113.
114.                     pTree.PrintTree();
115.
116.
117.                     Show_Detail(pTree);
118.
119.                 }
120.                 else
121.                 {
122.                     //create index 索引名 on 表名 ( 列名 );
123.                     if (sqlcode.IndexOf("create") != -1)
124.                     {
125.                         string listname = sqlcode.Substring(sqlcode.IndexOf
    ("on") + 3, sqlcode.IndexOf("(") - sqlcode.IndexOf("on") - 4);
```

```
126.                         string attributename = sqlcode.Substring(sqlcode.In
    dexOf("(") + 2, sqlcode.IndexOf(")") - sqlcode.IndexOf("(") - 3);
127.                         List<Attribute> attributes = new List<Attribute>();

128.
129.                         for (int i = 0; i <= (Application.Current as App).t
    ables.Count; i++)
130.                         {
131.                             if (i == (Application.Current as App).tables.Co
    unt)
132.                             {
133.                                 MessageDialog msg = new MessageDialog("Ther
    e is no this table!");
134.                                 _ = msg.ShowAsync();
135.                                 return;
136.                             }
137.                             if ((Application.Current as App).tables[i].get_
    name() == listname)
138.                             {
139.                                 attributes = (Application.Current as App).t
    ables[i].get_attribute();
140.                                 break;
141.                             }
142.                         }
143.
144.                         Attribute indexitem = new Attribute();
145.
146.                         for (int i = 0; i < attributes.Count; i++)
147.                         {
148.                             if (attributename == attributes[i].get_name())

149.                                 indexitem = attributes[i];
150.                         }
151.
152.                         BPlusTree.BPlusTree pTree = new BPlusTree.BPlusTree
    ();
153.                         pTree.ClearTree();
154.
155.                         for (int i = 0; i < indexitem.get_values().Count; i
    ++)
156.                         {
157.                             pTree.Insert(indexitem.get_values()[i]);
158.                         }
159.                         pTree.PrintTree();
```

```
160.
161.                         Show_Detail(pTree);
162.                     }
163.                     else
164.                     {
165.                         MessageDialog msg = new MessageDialog("Check your S
    QL code carefully!");
166.                         _ = msg.ShowAsync();
167.                         return;
168.                     }
169.                 }
170.             }
171.             catch (ArgumentException)
172.             {
173.                 MessageDialog msg = new MessageDialog("Check your SQL code
    carefully!");
174.                 _ = msg.ShowAsync();
175.             }
176.             catch (NullReferenceException)
177.             {
178.                 MessageDialog msg = new MessageDialog("You haven't input an
    y attribute's name or type.");
179.                 _ = msg.ShowAsync();
180.             }
181.             catch (Exception)
182.             {
183.                 MessageDialog msg = new MessageDialog("Something else wrong
     happened!");
184.                 _ = msg.ShowAsync();
185.             }
186.         }
187.         private void Show_Detail(BPlusTree.BPlusTree pTree)
188.         {
189.             if (null == pTree.GetRoot())
190.                 return;
191.
192.             BPlusTree.CNode p1, p2, p3;
193.             int i, j, k;
194.             int total = 0;
195.             string output = null;
196.             output += "\n 第一层\n | ";
197.             PrintNode(pTree.GetRoot(), ref output);
198.             total = 0;
199.             output += "\n 第二层\n | ";
```

```
200.
201.            for (i = 1; i <= mypub.MAXNUM_POINTER; i++)
202.            {
203.                p1 = pTree.GetRoot().GetPointer(i);
204.
205.                if (null == p1) continue;
206.                PrintNode(p1, ref output);
207.                total++;
208.
209.                if (total % 4 == 0) output += "\n | ";
210.            }
211.
212.            total = 0;
213.            output += "\n 第三层\n | ";
214.
215.            for (i = 1; i <= mypub.MAXNUM_POINTER; i++)
216.            {
217.                p1 = pTree.GetRoot().GetPointer(i);
218.
219.                if (null == p1) continue;
220.                for (j = 1; j <= mypub.MAXNUM_POINTER; j++)
221.                {
222.                    p2 = p1.GetPointer(j);
223.                    if (null == p2) continue;
224.                    PrintNode(p2, ref output);
225.                    total++;
226.                    if (total % 4 == 0) output += "\n | ";
227.                }
228.            }
229.
230.            total = 0;
231.            output += "\n 第四层\n | ";
232.
233.            for (i = 1; i <= mypub.MAXNUM_POINTER; i++)
234.            {
235.                p1 = pTree.GetRoot().GetPointer(i);
236.                if (null == p1) continue;
237.                for (j = 1; j <= mypub.MAXNUM_POINTER; j++)
238.                {
239.                    p2 = p1.GetPointer(j);
240.                    if (null == p2) continue;
241.                    for (k = 1; k <= mypub.MAXNUM_POINTER; k++)
242.                    {
243.                        p3 = p2.GetPointer(k);
```

```
                        if (null == p3) continue;
                        PrintNode(p3, ref output);
                        total++;
                        if (total % 4 == 0) output += "\n | ";
                    }
                }
            }

            MessageDialog msg = new MessageDialog(output);
            _ = msg.ShowAsync();
        }
        private void PrintNode(CNode pNode, ref string output)
        {
            if (null == pNode)
            {
                return;
            }

            for (int i = 1; i <= mypub.MAXNUM_KEY; i++)
            {
                string temp;
                if ((temp = pNode.GetElement(i)) != "\0")
                    output += temp;
                output += " ";
                if (i >= mypub.MAXNUM_KEY)
                {
                    output += " | ";
                }
            }
        }

        private async void writedata(object sender, RoutedEventArgs e)
        {
            try
            {
                string data = "";

                for (int i = 0; i < (Application.Current as App).tables.Cou
    nt; i++)
                {
                    Table element = (Application.Current as App).tables[i];

                    data += element.get_name() + "\n";
                    foreach (Attribute temp in element.get_attribute())
```

```
286.                     {
287.                         data += temp.get_name() + " " + temp.get_type() + "
     " + temp.get_primary().ToString();
288.                         data += "\n";
289.                     }
290.
291.                     data += (Application.Current as App).tables[i].get_reco
    rd_number().ToString() + ((Application.Current as App).tables[i].get_record_
    number() == 0 ? "" : "\n");
292.
293.                     for (int j = 0; j < element.get_record_number(); j++)
294.                     {
295.                         for (int k = 0; k < element.get_attribute().Count;
    k++)
296.                         {
297.                             data += element.get_attribute()[k].get_values()
    [j] + (k == element.get_attribute().Count - 1 ? "" : " ");
298.                         }
299.                         data += ((i == (Application.Current as App).tables.
    Count - 1 && j == element.get_record_number() - 1) ? "" : "\n");
300.                     }
301.
302.                     data += (i == (Application.Current as App).tables.Count
     - 1 ? "" : "\n");
303.                 }
304.
305.
306.                 FileSavePicker fp = new FileSavePicker();
307.                 var filedb = new[] { ".txt" };
308.                 fp.FileTypeChoices.Add("Plain Text", new List<string>() { "
    .txt" });
309.                 fp.SuggestedFileName = "table information" + DateTime.Now.D
    ay + "-" + DateTime.Now.Month + "-" + DateTime.Now.Year;
310.                 StorageFile sf = await fp.PickSaveFileAsync();
311.
312.                 if (sf != null)
313.                 {
314.                     using (StorageStreamTransaction transaction = await sf.
    OpenTransactedWriteAsync())
315.                     {
316.                         using (DataWriter dataWriter = new DataWriter(trans
    action.Stream))
317.                         {
318.                             dataWriter.WriteString(data);
```

```
                            transaction.Stream.Size = await dataWriter.StoreAsync();
                            await transaction.CommitAsync();
                        }
                    }
                }

                MessageDialog msg = new MessageDialog("Operation succeed!");
                _ = msg.ShowAsync();
            }
            catch (Exception)
            {
                MessageDialog msg = new MessageDialog("Something wrong happened.");
                _ = msg.ShowAsync();
            }

        }
    }
}
```

Index_Operations.xaml

```
<Page
    x:Class="MiniSQL.Index_Operations"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:MiniSQL"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>
        <NavigationView x:Name="Navigation" IsBackEnabled="True" IsSettingsVisible="True" IsTabStop="False" BackRequested="goback">
            <NavigationView.MenuItems>
                <NavigationViewItemHeader Content="Functions"/>
                <NavigationViewItem Icon="Home" Content="Main Page" Tapped="main"/>
                <NavigationViewItemSeparator/>

                <NavigationViewItem Icon="Import" Content="Table Operations" Tapped="table"></NavigationViewItem>
```

```
19.                 <NavigationViewItem Icon="List" Content="Index Operations" T
    apped="index"/>
20.                 <NavigationViewItem Icon="Find" Content="Query" Tapped="quer
    y"></NavigationViewItem>
21.                 <NavigationViewItem Icon="AllApps" Content="Insert" Tapped="
    insert"/>
22.                 <NavigationViewItem Icon="Delete" Content="Delete" Tapped="d
    elete"></NavigationViewItem>
23.                 <NavigationViewItem Icon="Next" Content="Export" Tapped="wri
    tedata"/>
24.                 <NavigationViewItem Icon="Undo" Content="Exit" Tapped="exit"
    />
25.             </NavigationView.MenuItems>
26.
27.             <Grid>
28.                 <Image Source="/Images/zjulib.jpg" Width="421.75" Height="11
    7.25" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0, 50, 0,
     0"></Image>
29.                 <Image Source="/Images/under.png" Width="557" Height="112" H
    orizontalAlignment="Center" VerticalAlignment="Top" Margin="0, 900, 0, 0"></
    Image>
30.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Margin="0, 200, 0, 0" FontSize="50" Width="500" Height="70" TextAlignmen
    t="Center">Index Operations</TextBlock>
31.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="C
    enter" Margin="0, 0, 0, 0" FontSize="50">OR</TextBlock>
32.
33.                 <TextBlock Width="400" Height="40" TextAlignment="Center" Fo
    ntSize="30" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="-
    900, 300, 0,0">Input your SQL Code Here</TextBlock>
34.                 <TextBox x:Name="source" Width="400" Height="400" VerticalAl
    ignment="Top" HorizontalAlignment="Center" Margin="-
    900, 350, 0, 0" FontSize="20" FontFamily="Consolas" AcceptsReturn="True"></T
    extBox>
35.                 <Button Width="200" Height="60" FontSize="30" VerticalAlignm
    ent="Top" HorizontalAlignment="Center" Margin="-
    900, 800, 0, 0" Click="execute">Execute</Button>
36.
37.                 <TextBlock Width="500" Height="40" TextAlignment="Center" Fo
    ntSize="30" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="900
    , 300, 0,0">Quick Operations Using Templates</TextBlock>
38.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 400, 0, 0"
    >Index Name:</TextBlock>
```

```
39.                 <TextBox x:Name="index_name" PlaceholderText="Enter index na
    me" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="1000, 400,
    0, 0" Width="400" Height="40" FontSize="20"></TextBox>
40.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 500, 0, 0"
    >Operation:</TextBlock>
41.                 <RadioButton x:Name="create" HorizontalAlignment="Center" Ve
    rticalAlignment="Top" Margin="800, 500, 0, 0" FontSize="20">Create</RadioBut
    ton>
42.                 <RadioButton x:Name="del" HorizontalAlignment="Center" Verti
    calAlignment="Top" Margin="1100, 500, 0, 0" FontSize="20">Delete</RadioButto
    n>
43.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 600, 0, 0"
    >Table Name:</TextBlock>
44.                 <TextBox x:Name="table_name" PlaceholderText="Enter index ta
    ble" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="1000, 600,
     0, 0" Width="400" Height="40" FontSize="20"></TextBox>
45.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 700, 0, 0"
    >Attribute Name:</TextBlock>
46.                 <TextBox x:Name="attribute_name" PlaceholderText="Enter inde
    x attribute" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="10
    00, 700, 0, 0" Width="400" Height="40" FontSize="20"></TextBox>
47.
48.             </Grid>
49.
50.             <NavigationView.PaneFooter>
51.                 <HyperlinkButton x:Name="MoreInfoBtn" Content="Email Me" Mar
    gin="12,0" NavigateUri="mailto:ZJU.SLM@gmail.com"/>
52.             </NavigationView.PaneFooter>
53.         </NavigationView>
54.     </Grid>
55. </Page>
```

Delete_Operations.xaml.cs

```
1. using System;
2. using System.Collections.Generic;
3. using Windows.UI.Popups;
4. using Windows.UI.Xaml;
5. using Windows.UI.Xaml.Controls;
6. using Windows.UI.Xaml.Input;
7. using Windows.Storage.Streams;
8. using Windows.Storage;
```

```
using Windows.Storage.Pickers;

// The Blank Page item template is documented at https://go.microsoft.com/fwlink/?LinkId=234238

namespace MiniSQL
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class Delete_Operations : Page
    {
        public Delete_Operations()
        {
            this.InitializeComponent();
        }

        private void goback(NavigationView sender, NavigationViewBackRequestedEventArgs args)
        {
            Frame.GoBack();
        }

        private void main(object sender, TappedRoutedEventArgs e)
        {
            Frame.Navigate(typeof(MainPage), "");
        }

        private void query(object sender, TappedRoutedEventArgs e)
        {
            Frame.Navigate(typeof(Querys), "");
        }

        private void exit(object sender, TappedRoutedEventArgs e)
        {
            (Application.Current as App).Exit();
        }

        private void table(object sender, TappedRoutedEventArgs e)
        {
            Frame.Navigate(typeof(Table_Operations), "");
        }

```

```
50.         private void index(object sender, TappedRoutedEventArgs e)
51.         {
52.             Frame.Navigate(typeof(Index_Operations), "");
53.         }
54.
55.         private void insert(object sender, TappedRoutedEventArgs e)
56.         {
57.             Frame.Navigate(typeof(Insert_Operations), "");
58.         }
59.
60.         private void delete(object sender, TappedRoutedEventArgs e)
61.         {
62.             Frame.Navigate(typeof(Delete_Operations), "");
63.         }
64.
65.         private void execute(object sender, RoutedEventArgs e)
66.         {
67.             /* write your codes behind */
68.             try
69.             {
70.                 if (table_name.Text == "")
71.                 {
72.                     if (source.Text.IndexOf("where") == -1)
73.                     {
74.                         string name = source.Text.Substring(12, source.Text.
    Length - 13);
75.                         int table_index = -1;
76.
77.                         for (int i = 0; i < (Application.Current as App).tab
    les.Count; i++)
78.                         {
79.                             if ((Application.Current as App).tables[i].get_n
    ame() == table_name.Text)
80.                             {
81.                                 table_index = i;
82.                                 break;
83.                             }
84.                         }
85.
86.                         if (table_index >= 0)
87.                         {
88.                             (Application.Current as App).tables[table_index]
    .clear();
89.                         }
```

```
90.                         else
91.                         {
92.                             MessageDialog msg = new MessageDialog("The table
    does not exist!");
93.                             _ = msg.ShowAsync();
94.                             return;
95.                         }
96.                     }
97.                     else
98.                     {
99.                         string name = source.Text.Substring(12, source.Text.
    IndexOf("where") - 13);
100.                         int table_index = -1;
101.
102.                         for (int i = 0; i < (Application.Current as App).ta
    bles.Count; i++)
103.                         {
104.                             if ((Application.Current as App).tables[i].get_
    name() == name)
105.                             {
106.                                 table_index = i;
107.                                 break;
108.                             }
109.                         }
110.
111.                         if (table_index >= 0)
112.                         {
113.                             List<string> conditions = new List<string>(sour
    ce.Text.Substring(source.Text.IndexOf("where") + 6, source.Text.Length - 7 -
     source.Text.IndexOf("where")).Split(" and "));
114.                             List<int> index = new List<int>();
115.
116.                             for (int i = 0; i < conditions.Count; i++)
117.                             {
118.                                 List<string> expression = new List<string>(
    conditions[i].Split(' '));
119.
120.                                 for (int j = 0; j < (Application.Current as
     App).tables[table_index].get_attribute().Count; j++)
121.                                 {
122.                                     if (expression[0] == (Application.Curre
    nt as App).tables[table_index].get_attribute()[j].get_name() && (Application
    .Current as App).tables[table_index].get_attribute()[j].get_type().IndexOf("
    char") != -1)
```

```
123.                                     {
124.                                         expression[2] = expression[2].Subst
    ring(1, expression[2].Length - 2);
125.                                         conditions[i] = expression[0] + ' '
     + expression[1] + ' ' + expression[2];
126.                                     }
127.                                 }
128.                             }
129.
130.                             List<int> attribute_index = new List<int>();
131.
132.                             for (int i = 0; i < conditions.Count; i++)
133.                             {
134.                                 List<string> expression = new List<string>(
    conditions[i].Split(' '));
135.
136.                                 for (int j = 0; j < (Application.Current as
     App).tables[table_index].get_attribute().Count; j++)
137.                                 {
138.                                     if (expression[0] == (Application.Curre
    nt as App).tables[table_index].get_attribute()[j].get_name())
139.                                     {
140.                                         attribute_index.Add(j);
141.                                     }
142.                                 }
143.                             }
144.
145.                             for (int i = 0; i < (Application.Current as App
    ).tables[table_index].get_record_number(); i++)
146.                             {
147.                                 index.Add(i);
148.                             }
149.
150.                             for (int i = 0; i < conditions.Count; i++)
151.                             {
152.                                 List<string> expression = new List<string>(
    conditions[i].Split(' '));
153.
154.                                 if (expression[1] == "=")
155.                                 {
156.                                     for (int j = 0; j < (Application.Curren
    t as App).tables[table_index].get_record_number(); j++)
157.                                     {
```

```
158.                                         if (!(expression[2] == (Application
    .Current as App).tables[table_index].get_attribute()[attribute_index[i]].get
    _values()[j]))
159.                                         {
160.                                             index.Remove(j);
161.                                         }
162.                                     }
163.                                 }
164.                                 else if (expression[1] == "<>")
165.                                 {
166.                                     for (int j = 0; j < (Application.Curren
    t as App).tables[table_index].get_record_number(); j++)
167.                                     {
168.                                         if (!(expression[2] != (Application
    .Current as App).tables[table_index].get_attribute()[attribute_index[i]].get
    _values()[j]))
169.                                         {
170.                                             index.Remove(j);
171.                                         }
172.                                     }
173.                                 }
174.                                 else if (expression[1] == "<")
175.                                 {
176.                                     for (int j = 0; j < (Application.Curren
    t as App).tables[table_index].get_record_number(); j++)
177.                                     {
178.                                         if (!(string.Compare(expression[2],
     (Application.Current as App).tables[table_index].get_attribute()[attribute_
    index[i]].get_values()[j]) < 0))
179.                                         {
180.                                             index.Remove(j);
181.                                         }
182.                                     }
183.                                 }
184.                                 else if (expression[1] == ">")
185.                                 {
186.                                     for (int j = 0; j < (Application.Curren
    t as App).tables[table_index].get_record_number(); j++)
187.                                     {
188.                                         if (!(string.Compare(expression[2],
     (Application.Current as App).tables[table_index].get_attribute()[attribute_
    index[i]].get_values()[j]) > 0))
189.                                         {
190.                                             index.Remove(j);
```

```
                                        }
                                    }
                                }
                                else if (expression[1] == "<=")
                                {
                                    for (int j = 0; j < (Application.Curren
    t as App).tables[table_index].get_record_number(); j++)
                                    {
                                        if (!(string.Compare(expression[2],
     (Application.Current as App).tables[table_index].get_attribute()[attribute_
    index[i]].get_values()[j]) <= 0))
                                        {
                                            index.Remove(j);
                                        }
                                    }
                                }
                                else if (expression[1] == ">=")
                                {
                                    for (int j = 0; j < (Application.Curren
    t as App).tables[table_index].get_record_number(); j++)
                                    {
                                        if (!(string.Compare(expression[2],
     (Application.Current as App).tables[table_index].get_attribute()[attribute_
    index[i]].get_values()[j]) >= 0))
                                        {
                                            index.Remove(j);
                                        }
                                    }
                                }
                            }

                            if ((Application.Current as App).tables[table_i
    ndex].remove_record(index))
                            {
                                MessageDialog msg = new MessageDialog("Oper
    ation Succeed!");
                                _ = msg.ShowAsync();
                                return;
                            }
                        }
                        else
                        {
                            MessageDialog msg = new MessageDialog("The tabl
    e does not exist!");
```

```
                                _ = msg.ShowAsync();
                                return;
                            }
                        }
                    }
                    else
                    {
                        int table_index = -1;

                        for (int i = 0; i < (Application.Current as App).tables
    .Count; i++)
                        {
                            if ((Application.Current as App).tables[i].get_name
    () == table_name.Text)
                            {
                                table_index = i;
                                break;
                            }
                        }

                        if (table_index >= 0)
                        {
                            List<int> index = new List<int>();

                            if (!(bool)and.IsChecked && !(bool)or.IsChecked)
                            {
                                MessageDialog msg = new MessageDialog("You have
    n't choose an opertion yet!");
                                _ = msg.ShowAsync();
                                return;
                            }
                            else
                            {
                                int index1 = -1, index2 = -1;
                                for (int i = 0; i < (Application.Current as App
    ).tables[table_index].get_attribute().Count; i++)
                                {
                                    if (attribute1.Text == (Application.Current
     as App).tables[table_index].get_attribute()[i].get_name())
                                    {
                                        index1 = i;
                                    }
                                    if (attribute2.Text == (Application.Current
     as App).tables[table_index].get_attribute()[i].get_name())
```

```
264.                                 {
265.                                     index2 = i;
266.                                 }
267.                             }
268.
269.                             for (int i = 0; i < (Application.Current as App
    ).tables[table_index].get_record_number(); i++)
270.                             {
271.                                 Attribute attributes1 = (Application.Curren
    t as App).tables[table_index].get_attribute()[index1];
272.                                 Attribute attributes2 = (Application.Curren
    t as App).tables[table_index].get_attribute()[index2];
273.                                 bool flag1 = false, flag2 = false;
274.
275.                                 if ((bool)and.IsChecked)
276.                                 {
277.                                     var item = (ComboBoxItem)operator1.Sele
    ctedItem;
278.                                     if (item.Name == "equal1" && attributes
    1.get_values()[i] == value1.Text && attributes2.get_values()[i] == value2.Te
    xt)
279.                                     {
280.                                         flag1 = true;
281.                                     }
282.                                     if (item.Name == "notequal1" && attribu
    tes1.get_values()[i] == value1.Text && attributes2.get_values()[i] != value2
    .Text)
283.                                     {
284.                                         flag1 = true;
285.                                     }
286.                                     if (item.Name == "less1" && attributes1
    .get_values()[i] == value1.Text && string.Compare(attributes2.get_values()[i
    ], value2.Text) < 0)
287.                                     {
288.                                         flag1 = true;
289.                                     }
290.                                     if (item.Name == "large1" && attributes
    1.get_values()[i] == value1.Text && string.Compare(attributes2.get_values()[
    i], value2.Text) > 0)
291.                                     {
292.                                         flag1 = true;
293.                                     }
```

```
294.                                    if (item.Name == "lessequal1" && attrib
    utes1.get_values()[i] == value1.Text && string.Compare(attributes2.get_value
    s()[i], value2.Text) <= 0)
295.                                    {
296.                                        flag1 = true;
297.                                    }
298.                                    if (item.Name == "largeequal1" && attri
    butes1.get_values()[i] == value1.Text && string.Compare(attributes2.get_valu
    es()[i], value2.Text) >= 0)
299.                                    {
300.                                        flag1 = true;
301.                                    }
302.                                }
303.                                else if ((bool)or.IsChecked)
304.                                {
305.                                    var item = (ComboBoxItem)operator2.Sele
    ctedItem;
306.                                    if (item.Name == "equal1" && (attribute
    s1.get_values()[i] == value1.Text || attributes2.get_values()[i] == value2.T
    ext))
307.                                    {
308.                                        flag2 = true;
309.                                    }
310.                                    if (item.Name == "notequal1" && (attrib
    utes1.get_values()[i] == value1.Text || attributes2.get_values()[i] != value
    2.Text))
311.                                    {
312.                                        flag2 = true;
313.                                    }
314.                                    if (item.Name == "less1" && (attributes
    1.get_values()[i] == value1.Text || string.Compare(attributes2.get_values()[
    i], value2.Text) < 0))
315.                                    {
316.                                        flag2 = true;
317.                                    }
318.                                    if (item.Name == "large1" && (attribute
    s1.get_values()[i] == value1.Text || string.Compare(attributes2.get_values()
    [i], value2.Text) > 0))
319.                                    {
320.                                        flag2 = true;
321.                                    }
322.                                    if (item.Name == "lessequal1" && (attri
    butes1.get_values()[i] == value1.Text || string.Compare(attributes2.get_valu
    es()[i], value2.Text) <= 0))
```

```
323.                                     {
324.                                         flag2 = true;
325.                                     }
326.                                     if (item.Name == "largeequal1" && (attr
    ibutes1.get_values()[i] == value1.Text || string.Compare(attributes2.get_val
    ues()[i], value2.Text) >= 0))
327.                                     {
328.                                         flag2 = true;
329.                                     }
330.                                 }
331.
332.                                 if (flag1 || flag2)
333.                                 {
334.                                     index.Add(i);
335.                                 }
336.
337.                             }
338.
339.                             if ((Application.Current as App).tables[table_i
    ndex].remove_record(index))
340.                             {
341.                                 MessageDialog msg = new MessageDialog("Oper
    ation Succeed!");
342.                                 _ = msg.ShowAsync();
343.                                 return;
344.                             }
345.                         }
346.                     }
347.                     else
348.                     {
349.                         MessageDialog msg = new MessageDialog("The table do
    es not exist!");
350.                         _ = msg.ShowAsync();
351.                         return;
352.                     }
353.                 }
354.             }
355.             catch (ArgumentException)
356.             {
357.                 MessageDialog msg = new MessageDialog("Check your SQL code
    carefully!");
358.                 _ = msg.ShowAsync();
359.             }
360.             catch (NullReferenceException)
```

```
361.             {
362.                 MessageDialog msg = new MessageDialog("You haven't input an
    y attribute's name or type.");
363.                 _ = msg.ShowAsync();
364.             }
365.             catch (Exception)
366.             {
367.                 MessageDialog msg = new MessageDialog("Something else wrong
     happened!");
368.                 _ = msg.ShowAsync();
369.             }
370.         }
371.
372.         private async void writedata(object sender, RoutedEventArgs e)
373.         {
374.             try
375.             {
376.                 string data = "";
377.
378.                 for (int i = 0; i < (Application.Current as App).tables.Cou
    nt; i++)
379.                 {
380.                     Table element = (Application.Current as App).tables[i];

381.                     data += element.get_name() + "\n";
382.                     foreach (Attribute temp in element.get_attribute())
383.                     {
384.                         data += temp.get_name() + " " + temp.get_type() + "
     " + temp.get_primary().ToString();
385.                         data += "\n";
386.                     }
387.
388.                     data += (Application.Current as App).tables[i].get_reco
    rd_number().ToString() + ((Application.Current as App).tables[i].get_record_
    number() == 0 ? "" : "\n");
389.
390.                     for (int j = 0; j < element.get_record_number(); j++)
391.                     {
392.                         for (int k = 0; k < element.get_attribute().Count;
    k++)
393.                         {
394.                             data += element.get_attribute()[k].get_values()
    [j] + (k == element.get_attribute().Count - 1 ? "" : " ");
395.                         }
```

```
396.                         data += ((i == (Application.Current as App).tables.
    Count - 1 && j == element.get_record_number() - 1) ? "" : "\n");
397.                     }
398.
399.                     data += (i == (Application.Current as App).tables.Count
     - 1 ? "" : "\n");
400.                 }
401.
402.
403.                 FileSavePicker fp = new FileSavePicker();
404.                 var filedb = new[] { ".txt" };
405.                 fp.FileTypeChoices.Add("Plain Text", new List<string>() { "
    .txt" });
406.                 fp.SuggestedFileName = "table information" + DateTime.Now.D
    ay + "-" + DateTime.Now.Month + "-" + DateTime.Now.Year;
407.                 StorageFile sf = await fp.PickSaveFileAsync();
408.
409.                 if (sf != null)
410.                 {
411.                     using (StorageStreamTransaction transaction = await sf.
    OpenTransactedWriteAsync())
412.                     {
413.                         using (DataWriter dataWriter = new DataWriter(trans
    action.Stream))
414.                         {
415.                             dataWriter.WriteString(data);
416.                             transaction.Stream.Size = await dataWriter.Stor
    eAsync();
417.                             await transaction.CommitAsync();
418.                         }
419.                     }
420.                 }
421.
422.                 MessageDialog msg = new MessageDialog("Operation succeed!")
    ;
423.                 _ = msg.ShowAsync();
424.             }
425.             catch (Exception)
426.             {
427.                 MessageDialog msg = new MessageDialog("Something wrong happ
    ened.");
428.                 _ = msg.ShowAsync();
429.             }
430.
```

```
        }
    }
}
```

Delete_Operations.xaml

```
<Page
    x:Class="MiniSQL.Delete_Operations"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:MiniSQL"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <Grid>
        <NavigationView x:Name="Navigation" IsBackEnabled="True" IsSettingsVisible="True" IsTabStop="False" BackRequested="goback">
            <NavigationView.MenuItems>
                <NavigationViewItemHeader Content="Functions"/>
                <NavigationViewItem Icon="Home" Content="Main Page" Tapped="main"/>
                <NavigationViewItemSeparator/>

                <NavigationViewItem Icon="Import" Content="Table Operations" Tapped="table"></NavigationViewItem>
                <NavigationViewItem Icon="List" Content="Index Operations" Tapped="index"/>
                <NavigationViewItem Icon="Find" Content="Query" Tapped="query"></NavigationViewItem>
                <NavigationViewItem Icon="AllApps" Content="Insert" Tapped="insert"/>
                <NavigationViewItem Icon="Delete" Content="Delete" Tapped="delete"></NavigationViewItem>
                <NavigationViewItem Icon="Next" Content="Export" Tapped="writedata"/>
                <NavigationViewItem Icon="Undo" Content="Exit" Tapped="exit" />
            </NavigationView.MenuItems>

            <Grid>
                <Image Source="/Images/zjulib.jpg" Width="421.75" Height="117.25" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0, 50, 0, 0"></Image>
```

```
28.                 <Image Source="/Images/under.png" Width="557" Height="112" H
    orizontalAlignment="Center" VerticalAlignment="Top" Margin="0, 900, 0, 0"></
    Image>
29.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Margin="0, 200, 0, 0" FontSize="50" Width="500" Height="70" TextAlignmen
    t="Center">Delete Operations</TextBlock>
30.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="C
    enter" Margin="0, 0, 0, 0" FontSize="50">OR</TextBlock>
31.
32.                 <TextBlock Width="400" Height="40" TextAlignment="Center" Fo
    ntSize="30" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="-
    900, 300, 0,0">Input your SQL Code Here</TextBlock>
33.                 <TextBox PlaceholderText="Only AND supported" x:Name="source
    " Width="400" Height="400" VerticalAlignment="Top" HorizontalAlignment="Cent
    er" Margin="-
    900, 350, 0, 0" AcceptsReturn="True" FontSize="20" FontFamily="Consolas"></T
    extBox>
34.                 <Button Width="200" Height="60" FontSize="30" VerticalAlignm
    ent="Top" HorizontalAlignment="Center" Margin="-
    900, 800, 0, 0" Click="execute">Execute</Button>
35.
36.                 <TextBlock Width="500" Height="40" TextAlignment="Center" Fo
    ntSize="30" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="900
    , 300, 0,0">Quick Operations Using Templates</TextBlock>
37.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 400, 0, 0"
    >Table Name:</TextBlock>
38.                 <TextBox x:Name="table_name" PlaceholderText="Enter table na
    me you want to delete from" HorizontalAlignment="Center" VerticalAlignment="
    Top" Margin="1000, 400, 0, 0" Width="400" Height="40" FontSize="20"></TextBo
    x>
39.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="T
    op" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 500, 0, 0"
    >Filter Condition:</TextBlock>
40.                 <TextBox x:Name="attribute1" PlaceholderText="attribute" Fon
    tSize="20" Width="150" Height="40" HorizontalAlignment="Center" VerticalAlig
    nment="Top" Margin="750, 500, 0, 0"></TextBox>
41.                 <ComboBox x:Name="operator1" Header="Operator" Width="100" H
    orizontalAlignment="Center" VerticalAlignment="Top" Margin="1050, 480, 0, 0"
    >
42.                     <ComboBoxItem x:Name="equal1">=</ComboBoxItem>
43.                     <ComboBoxItem x:Name="notequal1"><></ComboBoxItem>
44.                     <ComboBoxItem x:Name="less1"><</ComboBoxItem>
45.                     <ComboBoxItem x:Name="large1">></ComboBoxItem>
```

```
46.                     <ComboBoxItem x:Name="lessequal1"><=</ComboBoxItem>
47.                     <ComboBoxItem x:Name="largeequal1">>=</ComboBoxItem>
48.                 </ComboBox>
49.                 <TextBox x:Name="value1" PlaceholderText="attribute" FontSize="20" Width="150" Height="40" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="1350, 500, 0, 0"></TextBox>
50.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="Top" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 600, 0, 0">Logic Condition:</TextBlock>
51.                 <RadioButton x:Name="and" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="850, 600, 0, 0">AND</RadioButton>
52.                 <RadioButton x:Name="or" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="1050, 600, 0, 0">OR</RadioButton>
53.                 <TextBlock HorizontalAlignment="Center" VerticalAlignment="Top" Width="200" FontSize="20" TextAlignment="Center" Margin="400, 700, 0, 0">Filter Condition:</TextBlock>
54.                 <TextBox x:Name="attribute2" PlaceholderText="attribute" FontSize="20" Width="150" Height="40" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="750, 700, 0, 0"></TextBox>
55.                 <ComboBox x:Name="operator2" Header="Operator" Width="100" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="1050,680,0,0">
56.                     <ComboBoxItem x:Name="equal2">=</ComboBoxItem>
57.                     <ComboBoxItem x:Name="notequal2"><></ComboBoxItem>
58.                     <ComboBoxItem x:Name="less2"><</ComboBoxItem>
59.                     <ComboBoxItem x:Name="large2">></ComboBoxItem>
60.                     <ComboBoxItem x:Name="lessequal2"><=</ComboBoxItem>
61.                     <ComboBoxItem x:Name="largeequal2">>=</ComboBoxItem>
62.                 </ComboBox>
63.                 <TextBox x:Name="value2" PlaceholderText="attribute" FontSize="20" Width="150" Height="40" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="1350, 700, 0, 0"></TextBox>
64.             </Grid>
65.             <NavigationView.PaneFooter>
66.                 <HyperlinkButton x:Name="MoreInfoBtn" Content="Email Me" Margin="12,0" NavigateUri="mailto:ZJU.SLM@gmail.com"/>
67.             </NavigationView.PaneFooter>
68.         </NavigationView>
69.     </Grid>
70. </Page>
```