Catalog manager 模块设计

计算机学院 软件工程 1201 尹依婷 3120102057

一、模块概述

Catalog Manager 负责管理数据库的所有模式信息，包括：

1．数据库中所有表的定义信息，包括表的名称、表中字段（列）数、主键、定义在该表上的索引。

2．表中每个字段的定义信息，包括字段类型、是否唯一等。

3．数据库中所有索引的定义，包括所属表、索引建立在那个字段上等。

Catalog Manager 还必需提供访问及操作上述信息的接口，供 Interpreter 和 API 模块使用。

二、主要功能

1．管理数据库中所有表的定义信息：用户进行 sql 操作的时候，通过 API 先调用 Catalog Manager 中的函数，如果数据库中有对应表格，则调用 buffermanager；如果没有，必须告诉用户失败原因。

2．管理表中每个字段的定义信息:用户创建、更新、删除表时，Catalog Manager 中对应的表的信息也改变，并把改变信息传递给 buffermanager。

3．管理数据库中所有索引的定义：当查询的属性有已建立的索引时，通过调用 indexmanager 查询。

三、对外提供的接口

与底层交互函数

1．与 buffer 交互

```
void setbufferManager(bufferManager* b)
```

2．管理数据库表中的信息

```
table_info transformArrayToTableInfo(char* b, int count);
```

与用户界面交互的函数

3．保存数据库表

```
void saveTable(table_info a);
```

4．读取数据库表

```
bool readTable(CString table_name, table_info& tableInfo);
```

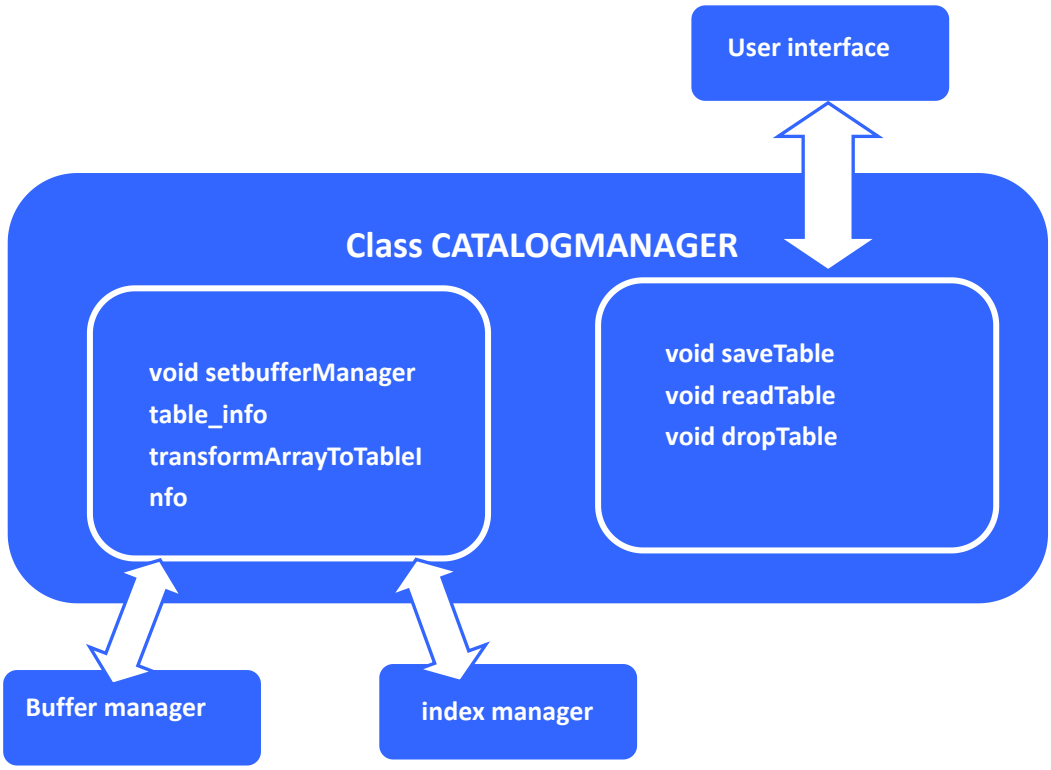
5．删除数据库表

```
void dropTable(CString table_name);
```

四、设计思路

CatalogManager 类为 API 调用 buffermanager 建立了桥梁，本模块中存储了数据库所

有表的定义信息、表中所有字段的定义信息和所有索引的信息。Class CatalogManager 主要有两类函数，一类是与底层交互的转化表格信息的函数，一类是与用户界面交互的管理表格信息的函数（如下图所示）。



五、 关键函数及代码

转化表信息

```
char* categoryManager::transformTableInfo(table_info a, int& count)
{
	for(int i = 0; i < a.attribute_count; i++)
		size[i] = a.info[i].attribute_name.GetLength() + a.info[i].type.GetLength() + sizeof(int)
+ sizeof(char)*2 + 2;   //规范表中每个字段的格式
	char* b = new char[table_size];
	char* p = b;
	_tcscpy(p, a.table_name); //存储表名、键名等信息
	p += a.table_name.GetLength();
	_tcscpy(p, a.primary_key);
	p += a.primary_key.GetLength();
	for(i=0; i < a.attribute_count; i++) //  按一定格式存储每个属性信息
	{
		p += 4;
			_tcscpy(p, a.info[i].attribute_name);
		p += a.info[i].attribute_name.GetLength();
		*p = '\0';
```

```
            p += 1;
                *p = (char)a.info[i].unique;
            p += 1;
            _tcscpy(p, a.info[i].type);
            p += a.info[i].type.GetLength();
            *p = '\0';
            p += 1;
            *((int*)p) = a.info[i].type_count;
            p += 4;
                *p = (char)a.info[i].has_index;
            p += 1;
        }
        count = table_size;
        return b;
}
```

将数组转化成表信息

```
table_info categoryManager::transformArrayToTableInfo(char* b, int count)
{
        for(i = 0; i < count && *(b + i) != '\0'; i++)
            a.table_name += *(b + i);
        for(i += 1; i < count && *(b + i) != '\0' ; i++)
                        a.primary_key += *(b + i);
        i += 1;
        for(k = 0; i < count; i++,k++)
        {
            size = *((int*)(b + i));
            i += 4;
            upper = i + size;
            for(; i < upper && *(b + i) != '\0'; i++)
                a.info[k].attribute_name += *(b + i);
            i += 1;
            a.info[k].unique = (*(b + i)) == 1 ? true : false;
            for(i += 1; i < upper && *(b + i) != '\0'; i++)
                a.info[k].type += *(b + i);
            i += 1;
            a.info[k].type_count = *((int*)(b + i));
            i += 4;
            a.info[k].has_index = (*(b + i)) == 1 ? true : false;
        }
        a.attribute_count = k - 1;
        return a;
}
```

保存表

```
void categoryManager::saveTable(table_info a)
{
	int arrysize;
	char* p = transformTableInfo(a, arrysize);
	bmp->addInfo(p, arrysize);
	delete [] p;
}
```

读取表

```
bool categoryManager::readTable(CString table_name, table_info& tableInfo)
{
	int arrysize;
	char* p;

	p = bmp->readInfo(table_name, arrysize);
	if(arrysize != -1)
	{
		tableInfo = transformArrayToTableInfo(p, arrysize);
		return true;
	}
	else
		return false;
}
```

删除表

```
void categoryManager::dropTable(CString table_name)
{
	bmp->rmInfo(table_name);
}
```