**说明文档V2.0**

**对于第一阶段的大部分结构和接口都在之前的说明文档中。**

**1、对之前接口的改动**

将Database.h中的std::map<std::**string**, **Table**> table\_list从private改为public。大部分操作均需要table\_list为公开成员。

将Database.h中的CreateTable 和 DropTable 函数和Databasemap.h中的CreateTable函数新增了即时存储所需的修改功能。

Databasemap类增加了获取当前数据库名称的getname()函数。

ShowColumns函数增加了对两个新类型**DATE**和**TIME**的支持。

**Row**的成员函数into中新增对**DATE**和**TIME**两种类型是否有效的判断，若无效分别插入0DATE和0TIME。

**2、对之前函数的增补**

**Command.cpp:**

将进入原select函数的条件加了一堆特判（基本没有除select的关键词时才进入原select函数）。这也证实许多组批判我们第一阶段不封装select的想法是完全错误的。

Create函数增加了两个新类型**DATE**和**TIME**的支持。

Update函数中新增了即时存储所需的修改功能。

Insert函数中新增了对**DATE**和**TIME**类型处理的部分（Time根据原来的接口，需要额外stringstream输入一次）。

Insert函数中新增了即时存储所需的修改功能。

Delete函数中新增了即时存储所需的修改功能。

OutputData函数增加了两个新类型**DATE**和**TIME**的支持。

Select函数新增了对于输出到文件的支持以及对**LIKE**的特判。

**3、新增函数和文件**

**Row.cpp:**

增添获取表头类型的函数getAttrType，为了增加对两个新类型**DATE**和**TIME**的支持。

新增JudgeValidDate和JudgeValidTime函数，用于判断数据是不是有效的**TIME**和**DATE**类型。

**Clause\_deal.cpp:**

由于原处理模块有许多弊端，在处理新的复杂语句时我们不得不采用新的处理模块，于是增添了Clause\_deal.cpp。

Clause\_deal函数：将输入语句用strtok函数分割成数个单元并存入**string**数组，并单独分隔出**WHERE**语句，且根据关键词执行不同的具体函数。

对于以下情况进行特判：

* 是否需要导出到文件
* select + count + groupby
* select + count + groupby + orderby
* select + count + orderby
* select + 无其他关键字且有有机数字函数（只要有其他关键字，因为进入我们编写的新接口，均支持。此举是为了避开且不改动原Select接口）
* select + union
* select + join

是否有直接输出的mathfunction

OrderByCompare函数：判断两行的某行数据的大小关系

OrderByCompare2函数：判断数据的大小关系

Find函数：查询某个**string**是否在已分割好的**string**数组中

Input函数：从文件输入表信息

Output函数：将表信息输出至文件

Count函数：处理比较简单的**COUNT**情况

Group\_by函数：处理**GROUP BY**语句（比较复杂的**COUNT**情况）（order\_by\_attr不为空时代表有**ORDER BY**，排序也需要讨论是对列还是对**COUNT**）

NewSelect函数：处理新的无法在旧**SELECT**函数中处理的语句，比如**LIKE**、有机数字函数、**ORDER BY**等等。

Pickitem和multipickitem函数：处理在**UNION**和**UNION ALL**时需要加入哪些行到集合中，分别采用set和multiset。

UnionCompare函数：比较两个已选数据集Vector的大小，用于处理**UNION** + **ORDER BY**的情况。

Union函数：处理**UNION**和**UNION ALL**，支持**ORDER BY**、**LIKE**和有机函数

Join函数：处理**INNER JOIN，LEFT JOIN**和**RIGHT JOIN**，支持**ORDERBY、LIKE**和有机函数

**Wherenode.cpp**

拓展功能，处理**WHERE**语句的抽象语法树，支持括号

一个**Wherenode**类，其中包含：

Char type; 代表节点的运算类型。四种逻辑运算类型’a’->AND, ‘o’->OR, ‘n’->NOT, ‘x’->XOR，结果为子树经过相应运算得到的结果；三种比较运算类型’>’, ‘<’, ‘=’，直接比较fieldname下值与要被比较值的关系并返回布尔值。

**string** fieldname; 要比较的左值的表头名

**string** data；要被比较的右值

**string** datatype；要被比较的右值的类型

**Wherenode** \*next1; 子树1指针

**Wherenode** \*next2; 子树2指针

**string** func; 如果有有机函数，记录有机函数的类型是什么

StringSplit函数，用于重新分割字符串以便之后构建处理WHERE语句的抽象语法树。

Setfield函数，用于确定被比较的左值的表头名（支持有机数字函数，存入节点等待进一步处理）

Getdata函数，用于确定要被比较的右值（支持有机数字函数，原位处理）

BuildWhereTree函数，用于构建抽象语法树。

JudgeWhere函数，用于遍历抽象语法树。

三个比较函数comparemore, compareless, compareequal, 用于遍历抽象语法树时比较具体数据的大小。

Func函数，用于确定左值后对其使用有机数字函数。

Like函数，用于处理**LIKE**语句。

getWhereKeys函数，用于获取含有所有符合**WHERE**条件的行的集合。

getWhereLikeKeys函数，用于获取含有所有符合**WHERE** **LIKE**条件的行的集合。

**mf.cpp**

用于处理各种数字函数、日期函数和算术运算符。

数字函数包括Avg, Max, Min, Locate, Lcase, Ucase, Space, Left, Right, Strcmp，其中Lcase, Ucase, Left可以有机使用在各个语句中。

日期函数包括AddDate和AddTime，保证时间加减的准确性。

算术运算符包括+, -, \*, /, DIV, MOD。

**远程通信部分**

见实验报告。