语法分析器报告

1.相关信息

a) **成员及分工**

代码开发：　石子跃3020244294
算法描述：　胡豆豆3020244319
　　刘玲君3020244151
流程图：　徐心蕙3020244310
输出说明：　徐心蕙
Ppt：　胡豆豆/刘玲君
讲解：　胡豆豆/刘玲君

b) **开发环境**

```
PS C:\Users\Ziyueshi\Desktop\大二下\编译原理\NFADFA> g++ -v
Using built-in specs.
COLLECT_GCC=/usr/bin/g++
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-pc-cygwin/11/lto-wrapper.exe
Target: x86_64-pc-cygwin
Configured with: /mnt/share/cygpkgs/gcc/gcc.x86_64/src/gcc-11.2.0/configure --srcdir=/mnt/share/cygpkgs/gcc/gcc.x86_64/src/gcc-11.2.0 --prefix=/usr --exec-prefix=/usr --localstatedir=/var --sysconfdir=/etc --docdir=/usr/share/doc/gcc --htmldir=/usr/share/doc/gcc/html -C --build=x86_64-pc-cygwin --host=x86_64-pc-cygwin --target=x86_64-pc-cygwin --without-libiconv-prefix --without-libintl-prefix --libexecdir=/usr/lib --with-gcc-major-version-only --enable-shared --enable-shared-libgcc --enable-static --enable-version-specific-runtime-libs --enable-bootstrap --enable-__cxa_atexit --with-dwarf2 --with-tune=generic --disable-bootstrap --enable-languages=c,c++,fortran,lto,objc,obj-c++,jit --enable-graphite --enable- threads=posix --enable-libatomic --enable-libgomp --enable-libquadmath -- enable-libquadmath-support --disable-libssp --enable-libada --disable-symvers --with-gnu-ld --with-gnu-as --with-cloog-include=/usr/include/cloog-isl --without-libiconv-prefix --without-libintl-prefix --with-system-zlib --enable-linker-build-id --with-default-libstdcxx-abi=gcc4-compatible --enable-libstdcxx-filesystem-ts
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 11.2.0 (GCC)
```

c) 文件说明

1. Syntax.h:定义了产生式、语法的结构，以及语法分析所用到的方法
2. Syntax_method.cpp:定义了生成及打印输出FIRST集、FOLLOW集、预测表的方法
3. Syntax_main.cpp:读取sql.txt和grammar.txt中的内容，并进行语法分析

2.构造FIRST集

a) 算法描述

```
void Syntax_Analysis::Init_FirstTable(Grammar grammar)
{
    if(X∈VT)
        FIRST(X)={X};
    初始化剩余的<文法符号,相应的First集>;
    构造FIRST集合的算法I:
        bool change = true;
        while (change)
        {
            change = false;
            找到每个产生式左侧对应的first集合;
            指向当前产生式右部符号集;
            bool flag = true;
            while (flag)
            {
                flag = false;
                找到产生式右侧符号的first集合;
                for (auto iter = tofind.begin(); iter != tofind.end(); iter++){
                    if(当前不在左侧的first集中){
                        tarfirst.insert(*iter);
                        change = true;
                    }
                    else if(有$){
                        去向后一个右部符号的first集;
                        flag = true;
                    }
                }
            }
            更新当前左部最近的first集;
        }
```

```
    }
    构造FIRST集合的算法Ⅱ：
        if (FirstTable.find(prdt->right) != FirstTable.end())
        {
            找到每个产生式右侧对应的first集合;
            指向当前产生式右部符号集;
            bool flag = true;
            while (flag)
            {
                flag = false;
                找到产生式右侧符号的first集合;
                    for (auto iter = tofind.begin(); iter != tofind.end(); iter++)
                    {
                        if (当前不在左侧的first集){
                            tarfirst.insert(*iter);
                        }
                        else if (有$){
                            去向后一个右部符号的first集;
                            flag = true;
                        }
                    }
            }
            更新当前右部最近的first集;
        }
```

3.构造FOLLOW集

a) 算法描述

```
void Syntax_Analysis::Init_FollowTable(Grammar grammar)
{
    1)对于文法的开始符号S，置#于FOLLOW(S)中;
    bool keepon = true;
    while (keepon)
    {
        keepon = false;
        for (auto prdct = grammar.productions.begin(); prdct != grammar.productions.end(); prdct++)
        {
            int length = 产生式右部的长度;
            去first中寻找对应的产生式右部是否存在$转换
            for (int i = 1; i < length; i++)
            {
                去first中寻找对应的产生式右部是否存在$转换;
            }
        }
```

```
    2)若A->αBβ是一个产生式，则把FIRST(β)\{ε}加至FOLLOW(B)中;
            for (int i = 0; i < length - 1; i++)
            {
                int j = i + 1;
                set<string> &B1 = FollowTable[prdct->rights[i]];
                int late = B1.size();
                do
                {
                    set<string> &Bnextfirst = B后面的符号的first集合
                    auto iter = Bnextfirst.find("$");
                    B1.insert(Bnextfirst.begin(), iter);
                    B1.insert(++iter, Bnextfirst.end());
                    j++;
                } while (j < length && existepsilon[j - 1]);
                if (发生改变)
                {
                    keepon = true;
                    继续向后寻找;
                }
            }
    3）若A->αB是一个产生式，FOLLOW(A)加至FOLLOW(B)
            set<string> &A = FollowTable[prdct->left];
            set<string> &B2 = FollowTable[prdct->rights[length - 1]];
            int late2 = B2.size();
            if (发生改变)
                keepon = true;
                继续寻找;
            for (int i = 0; i < length - 1; i++)
            {
                β => ε (即ε∈FIRST(β));
            }
    }
}
```

4.构造预测表

a) 算法描述

```
void Syntax_Analysis::Init_PredictTable(Grammar grammar)
{
    for (auto prdt = grammar.productions.begin(); prdt != grammar.productions.end(); prdt++)
    {
        (1) 对文法G的每个产生式A->α,执行第(2)和(3)步;
        (2) 对每个终结符a∈FIRST(α), 把A->α加入M[A, a]中;
        (3) 若ε∈ FIRST(α), 则对任何b∈FOLLOW(A), 把A->ε加入M[A, b]中;
```

```
    }
}
```

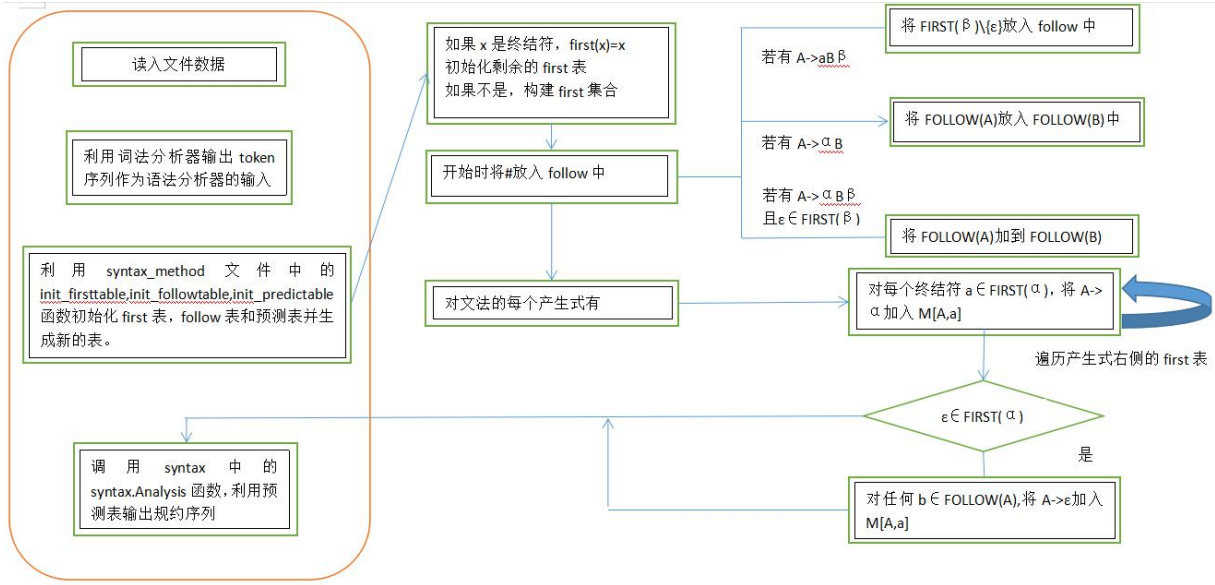
5.语法分析

a) 算法描述

```
int main(){
    利用 ifstream 函数读入 sql.txt 文件数据；
    利用 istreambuf_iterator 函数将数据写入字符串；
    利用 file.close 关闭文件；
    利用 SQL.ParseMain 进行解析，生成 Token 序列；

    利用 grammar 读入 grammar.txt 文件数据；

    构造 FIRST 表；
    打印 FIRST 表；
    构造 FOLLOW 表；
    打印 FOLLOW 表；
    构造预测表；
    打印预测表；

    Token ret;
    for(int i = 0; i < Token_List.size(); i++){
        ret = Token_List[i];
        cout<<ret.Tokentostring()<<"\n";
    }
    syntax.Analysis(grammar,Token_List);
    return 0;
}
```

```
bool Syntax_Analysis::Analysis(Grammar grammar, vector<Token> tokenlist)
{
    构造符号栈；
    栈底符号为#；
    Symbolzhan.push_back("root");
    定义串的第一个符号 a；
    构造指针 ip 指向 a；
    int step = 1;
    while (token 没有读取完，栈非空)
    {
        string X = Symbolzhan.back();
        if (ip 指向栈顶)
            a 即为#；
        else
        {
```

```
            if (token 的种别为关键字、运算符或界符")
            {
                a 即为词法单元;
            }
            else
                a 为 ip 所指 token 的种别;
        }
        if (!"#$%&')
        {
            if (X = a = #)
            {
                接受;
            }
            else if (X = a 且都不为#)
            {
                弹出 X,并使 ip 前进;
            }
        }
        else 即 X 为非终结符
        {
            if (PredictTable[X].count(a) != 0)
            {
                Production Y1Y2Y3 = PredictTable[X][a];
                Symbolzhan.pop_back();
                for (int i = Y1Y2Y3.rights.size() - 1; i >= 0; i--)
                {
                    if (Y1Y2Y3.rights[i] != "$")
                        Symbolzhan.push_back(Y1Y2Y3.rights[i]);
                }
                cout << step++ << "\t" << Y1Y2Y3.id << "\t" << X << "#" <<
a << "\t"
                     << "reduction" << endl;
            }
            else
            {
                出错;
            }
        }
    }
    return 0;
}
```



语法分析器流程图

b) 输出格式说明

语法分析器是以词法分析器的输出作为输入进行解析，此处采用LL（1）文法进行解析，需要输出FIRST集和FOLLOW集（按照如图1所示样例输出）

2、语法分析输出示例（以预测分析为例）

（1）【可选】输出 FIRST 集和 FOLLOW 集（此处仅展示一部分）：

```
FIRST:
   functionCall=[AVG, MAX, MIN, SUM],
   selectClause=[FROM, GROUP, HAVING, ORDER],
   expression=[NOT, !, IDN, -, STR, FLOAT, INT, TRUE, FALSE, AVG, MAX, MIN, SUM],
   comparisonOperator=[=, >, <, <=, >=, !=, <=>],
   selectElementHead=[*, IDN, AVG, MAX, MIN, SUM],
                                                    ......
FOLLOW:
   functionCall=[AS, IDN, ,, FROM, GROUP, HAVING, ORDER, =, >, <, <=, >=, !=, <=>, IS, AND, &&, XOR, OR,
   ||, UNION, JOIN, WHERE, LEFT, RIGHT, ]],
   selectElements=[FROM, GROUP, HAVING, ORDER],
   expression=[ORDER, UNION, ,, HAVING, JOIN, WHERE, GROUP, LEFT, RIGHT, ]],
   logicalOperator=[NOT, !, IDN, -, STR, FLOAT, INT, TRUE, FALSE, AVG, MAX, MIN, SUM],
                                                    ......
```

代码 3 FIRST 集、FOLLOW 集部分输出示例

图 1FIRST FOLLOW 输出示例

产生的预测分析表输出至语法分析器文件夹下的exel表中（在下面给出部分截图）
最终产生的移进规约序列如图2所示，需要以

[序号] [TAB] [选用规则序号] [TAB] [栈顶符号]#[面临输入符号] [TAB] [执行动作]

的格式进行输出。

(2)【必须按规定格式输出】输出规约序列（此处仅展示一部分）：

输出格式：

[序号] [TAB] [选用规则序号] [TAB] [栈顶符号]#[面临输入符号] [TAB] [执行动作]

其中，选用规则序号见附件文法规则；执行动作为“reduction”（归约），“move”（LL 分析的跳过或 LR 分析的移进），“accept”（接受）或“error”（错误）。

```
1   1   root#SELECT   reduction
2   2   dmlStatement#SELECT    reduction
3   6   selectStatement#SELECT reduction
4   /   SELECT#SELECT move
5   14  unionType#IDN reduction
6   26  selectElements#IDN     reduction
7   28  selectElementHead#IDN  reduction
8   31  selectElement#IDN reduction
9   49  fullColumnName#IDN     reduction
10  48  uid#IDN  reduction
11  /   IDN#IDN  move
12  50  dottedId#.    reduction
13  /   .#.  move
14  48  uid#IDN  reduction
15  /   IDN#IDN  move
                .....
```

代码 4 归约序列部分输出示例

图 2 规约序列的输出格式

此处采用测试用例2得到的规约序列输出如下：

```
1       1       root#SELECT     reduction
2       2       dmlStatement#SELECT     reduction
3       6       selectStatement#SELECT  reduction
4       15      querySpecification#SELECT       reduction
5       /       SELECT#SELECT   move
6               unionType#IDN   reduction
7       26      selectElements#IDN      reduction
8       28      selectElementHead#IDN   reduction
9       31      selectElement#IDN       reduction
10      49      fullColumnName#IDN      reduction
11      48      uid#IDN reduction
12      /       IDN#IDN move
13      50      dottedId#.      reduction
14      /       .#.     move
15      52      dottedIdOrStar#IDN      reduction
16      48      uid#IDN reduction
17      /       IDN#IDN move
18              elementNameAlias#,      reduction
19      29      selectElementListRec#,  reduction
20      /       ,#,     move
21      32      selectElement#SUM       reduction
22      100     functionCall#SUM        reduction
23      101     aggregateWindowedFunction#SUM   reduction
24      105     function#SUM    reduction
25      /       SUM#SUM move
26      /       (#(     move
27              unionType#IDN   reduction
28      49      fullColumnName#IDN      reduction
29      48      uid#IDN reduction
30      /       IDN#IDN move
31      50      dottedId#.      reduction
32      /       .#.     move
33      52      dottedIdOrStar#IDN      reduction
34      48      uid#IDN reduction
35      /       IDN#IDN move
36      /       )#)     move
37              elementNameAlias#FROM   reduction
38              selectElementListRec#FROM       reduction
39      17      selectClause#FROM       reduction
40      18      fromClause#FROM reduction
41      /       FROM#FROM       move
42      36      tableSources#IDN        reduction
43      39      tableSource#IDN reduction
44      42      tableSourceItem#IDN     reduction
45      44      tableName#IDN   reduction
46      48      uid#IDN reduction
47      /       IDN#IDN move
48              elementNameAlias#JOIN   reduction
49      40      joinParts#JOIN  reduction
```

```
46      48      uid#IDN reduction
47      /       IDN#IDN move
48              elementNameAlias#JOIN   reduction
49      40      joinParts#JOIN  reduction
50      106     joinPart#JOIN   reduction
51      /       JOIN#JOIN       move
52      42      tableSourceItem#IDN     reduction
53      44      tableName#IDN   reduction
54      48      uid#IDN reduction
55      /       IDN#IDN move
56              elementNameAlias#ON     reduction
57      108     joinRightPart#ON        reduction
58      /       ON#ON   move
59      58      expression#IDN  reduction
60      72      predicate#IDN   reduction
61      76      expressionAtom#IDN      reduction
62      49      fullColumnName#IDN      reduction
63      48      uid#IDN reduction
64      /       IDN#IDN move
65      50      dottedId#.      reduction
66      /       .#.     move
67      52      dottedIdOrStar#IDN      reduction
```

```
67      52      dottedIdOrStar#IDN      reduction
68      48      uid#IDN reduction
69      /       IDN#IDN move
70      73      predicateRight#=        reduction
71      85      comparisonOperator#=    reduction
72      /       =#=     move
73      72      predicate#IDN   reduction
74      76      expressionAtom#IDN      reduction
75      49      fullColumnName#IDN      reduction
76      48      uid#IDN reduction
77      /       IDN#IDN move
78      50      dottedId#.      reduction
79      /       .#.     move
80      52      dottedIdOrStar#IDN      reduction
81      48      uid#IDN reduction
82      /       IDN#IDN move
83              predicateRight#WHERE    reduction
84              expressionRight#WHERE   reduction
85              joinParts#WHERE reduction
86              tableSourceListRec#WHERE        reduction
87      132     whereExpression#WHERE   reduction
88      /       WHERE#WHERE     move
89      58      expression#IDN  reduction
90      72      predicate#IDN   reduction
91      76      expressionAtom#IDN      reduction
92      49      fullColumnName#IDN      reduction
93      48      uid#IDN reduction
94      /       IDN#IDN move
95      50      dottedId#.      reduction
96      /       .#.     move
97      52      dottedIdOrStar#IDN      reduction
98      48      uid#IDN reduction
99      /       IDN#IDN move
100     73      predicateRight#>        reduction
101     86      comparisonOperator#>    reduction
102     /       >#>     move
103     72      predicate#INT   reduction
104     75      expressionAtom#INT      reduction
105     80      constant#INT    reduction
106     84      decimalLiteral#INT      reduction
107     /       INT#INT move
108             predicateRight#AND      reduction
109     59      expressionRight#AND     reduction
110     92      logicalOperator#AND     reduction
111     /       AND#AND move
112     58      expression#IDN  reduction
113     72      predicate#IDN   reduction
114     76      expressionAtom#IDN      reduction
115     49      fullColumnName#IDN      reduction
116     48      uid#IDN reduction
117     /       IDN#IDN move
118     50      dottedId#.      reduction
119     /       .#.     move
120     52      dottedIdOrStar#IDN      reduction
121     48      uid#IDN reduction
122     /       IDN#IDN move
```

```
120     52      dottedIdOrStar#IDN      reduction
121     48      uid#IDN reduction
122     /       IDN#IDN move
123     73      predicateRight#<        reduction
124     87      comparisonOperator#<    reduction
125     /       <#<     move
126     72      predicate#FLOAT reduction
127     75      expressionAtom#FLOAT    reduction
128     80      constant#FLOAT  reduction
129     83      decimalLiteral#FLOAT    reduction
130     /       FLOAT#FLOAT     move
131             predicateRight#OR       reduction
132     59      expressionRight#OR      reduction
133     95      logicalOperator#OR      reduction
```

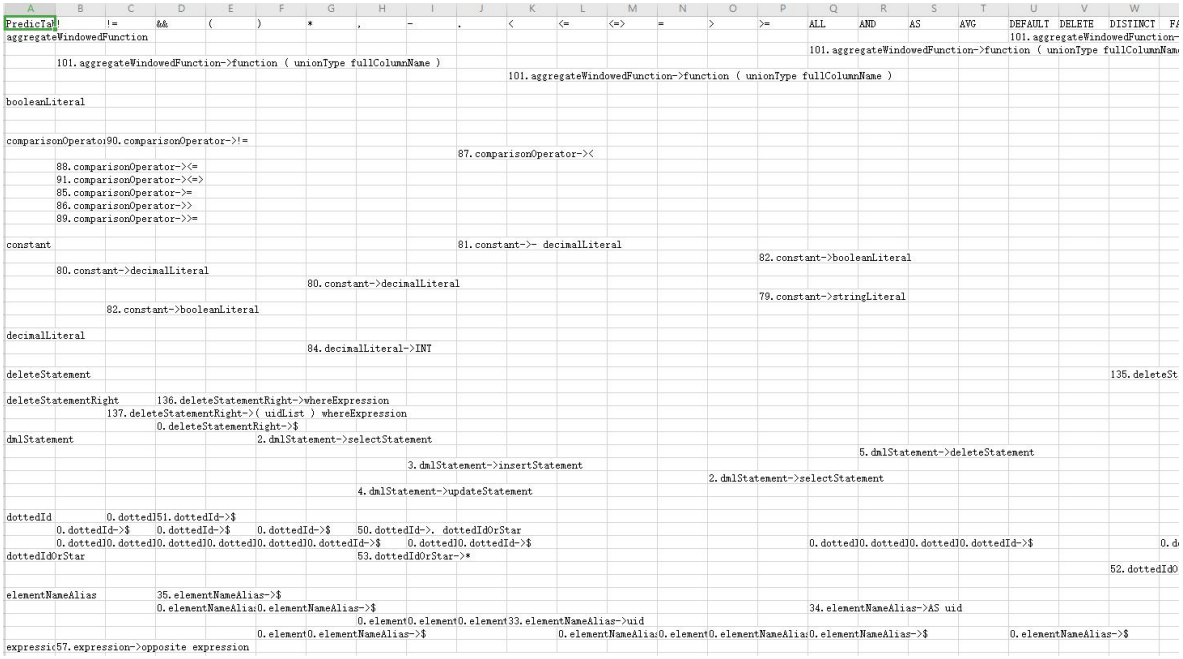
```
130     /       FLOAT#FLOAT     move
131             predicateRight#OR       reduction
132     59      expressionRight#OR      reduction
133     95      logicalOperator#OR      reduction
134     /       OR#OR   move
135     58      expression#FLOAT        reduction
136     72      predicate#FLOAT reduction
137     75      expressionAtom#FLOAT    reduction
138     80      constant#FLOAT  reduction
139     83      decimalLiteral#FLOAT    reduction
140     /       FLOAT#FLOAT     move
141             predicateRight#IS       reduction
142     60      expressionRight#IS      reduction
143     /       IS#IS   move
144     64      oppositeOrNot#NOT       reduction
145     /       NOT#NOT move
146     66      nullOrTrueValue#NULL    reduction
147     68      nullValue#NULL  reduction
148     /       NULL#NULL       move
149     20      groupByClause#GROUPBY   reduction
150     /       GROUPBY#GROUPBY move
151     54      expressions#IDN reduction
152     58      expression#IDN  reduction
153     72      predicate#IDN   reduction
154     76      expressionAtom#IDN      reduction
155     49      fullColumnName#IDN      reduction
156     48      uid#IDN reduction
157     /       IDN#IDN move
158     50      dottedId#.      reduction
159     /       .#.     move
160     52      dottedIdOrStar#IDN      reduction
161     48      uid#IDN reduction
162     /       IDN#IDN move
163             predicateRight#HAVING   reduction
164             expressionRight#HAVING  reduction
165             expressionRec#HAVING    reduction
166     22      havingClause#HAVING     reduction
167     /       HAVING#HAVING   move
168     58      expression#IDN  reduction
169     72      predicate#IDN   reduction
170     76      expressionAtom#IDN      reduction
```

```
170     76      expressionAtom#IDN      reduction
171     49      fullColumnName#IDN      reduction
172     48      uid#IDN reduction
173     /       IDN#IDN move
174     50      dottedId#.      reduction
175     /       .#.     move
176     52      dottedIdOrStar#IDN      reduction
177     48      uid#IDN reduction
178     /       IDN#IDN move
179     73      predicateRight#=        reduction
180     85      comparisonOperator#=    reduction
181     /       =#=     move
182     72      predicate#STRING        reduction
183     75      expressionAtom#STRING   reduction
184     79      constant#STRING reduction
185     97      stringLiteral#STRING    reduction
186     /       STRING#STRING   move
187             predicateRight##        reduction
188             expressionRight##       reduction
189             orderByClause## reduction
190             unionStatements##       reduction
191     #       accept
PS C:\Users\Ziyueshi\Desktop\大二下\编译原理\大作业\Syntax_Analysis>
```

FOLLOW集的部分输出：

```
FOLLOW:
        !       =       [ ! - AVG FALSE FLOAT IDN INT MAX MIN NOT STRING SUM TRUE ]
        !=      =       [ - AVG FALSE FLOAT IDN INT MAX MIN STRING SUM TRUE ]
        $       =       [ != # && ( ) * , . < <= <=> = > >= AND AS AVG FALSE FROM GROUPBY HAVING IDN IS JOIN LEFT MAX MIN NULL ON OR ORDERBY RIGHT SELECT SET SUM TRUE UNION UNKNOWN WHERE XOR
|| ]
        &&      =       [ ! - AVG FALSE FLOAT IDN INT MAX MIN NOT STRING SUM TRUE ]
        (       =       [ ! != # && ( ) , - . < <= <=> = > >= ALL AND AS AVG DEFAULT DISTINCT FALSE FLOAT FROM GROUPBY HAVING IDN INT IS JOIN LEFT MAX MIN NOT OR ORDERBY RIGHT SELECT STRING S
UM TRUE UNION WHERE XOR || ]
        )       =       [ != # && ) , < <= <=> = > >= AND AS FROM GROUPBY HAVING IDN IS JOIN LEFT ON OR ORDERBY RIGHT UNION VALUE VALUES WHERE XOR || ]
        *       =       [ != # && ) , < <= <=> = > >= AND AS FROM GROUPBY HAVING IDN IS JOIN LEFT OR ORDERBY RIGHT UNION WHERE XOR || ]
        ,       =       [ ! ( - AVG DEFAULT FALSE FLOAT IDN INT MAX MIN NOT STRING SUM TRUE ]
        -       =       [ FLOAT INT ]
        .       =       [ * IDN ]
        <       =       [ - AVG FALSE FLOAT IDN INT MAX MIN STRING SUM TRUE ]
        <=      =       [ - AVG FALSE FLOAT IDN INT MAX MIN STRING SUM TRUE ]
        <=>     =       [ - AVG FALSE FLOAT IDN INT MAX MIN STRING SUM TRUE ]
        =       =       [ ! - AVG DEFAULT FALSE FLOAT IDN INT MAX MIN NOT STRING SUM TRUE ]
        >       =       [ - AVG FALSE FLOAT IDN INT MAX MIN STRING SUM TRUE ]
        >=      =       [ - AVG FALSE FLOAT IDN INT MAX MIN STRING SUM TRUE ]
        ALL     =       [ ( * AVG IDN MAX MIN SELECT SUM ]
        AND     =       [ ! - AVG FALSE FLOAT IDN INT MAX MIN NOT STRING SUM TRUE ]
        AS      =       [ IDN ]
        AVG     =       [ ( ]
        DEFAULT =       [ # ) , WHERE ]
        DELETE  =       [ FROM ]
        DISTINCT        =       [ ( * AVG IDN MAX MIN SELECT SUM ]
        FALSE   =       [ != # && ) , < <= <=> = > >= AND GROUPBY HAVING IS JOIN LEFT OR ORDERBY RIGHT UNION WHERE XOR || ]
        FLOAT   =       [ != # && ) , < <= <=> = > >= AND GROUPBY HAVING IS JOIN LEFT OR ORDERBY RIGHT UNION WHERE XOR || ]
        FROM    =       [ ( IDN ]
        GROUPBY =       [ ! - AVG FALSE FLOAT IDN INT MAX MIN NOT STRING SUM TRUE ]
        HAVING  =       [ ! - AVG FALSE FLOAT IDN INT MAX MIN NOT STRING SUM TRUE ]
        IDN     =       [ != # && ( ) , . < <= <=> = > >= AND AS FROM GROUPBY HAVING IDN IS JOIN LEFT ON OR ORDERBY RIGHT SET UNION VALUE VALUES WHERE XOR || ]
        INSERT  =       [ IDN INTO ]
        INT     =       [ != # && ) , < <= <=> = > >= AND GROUPBY HAVING IS JOIN LEFT OR ORDERBY RIGHT UNION WHERE XOR || ]
        INTO    =       [ IDN ]
        IS      =       [ # ) , FALSE GROUPBY HAVING JOIN LEFT NOT NULL ORDERBY RIGHT TRUE UNION UNKNOWN WHERE ]
        JOIN    =       [ ( IDN ]
        LEFT    =       [ JOIN ]
        MAX     =       [ ( ]
        MIN     =       [ ( ]
        NOT     =       [ ! - AVG FALSE FLOAT IDN INT MAX MIN NOT NULL STRING SUM TRUE UNKNOWN ]
        NULL    =       [ # ) , GROUPBY HAVING JOIN LEFT ORDERBY RIGHT UNION WHERE ]
        ON      =       [ ! - AVG FALSE FLOAT IDN INT MAX MIN NOT STRING SUM TRUE ]
        OR      =       [ ! - AVG FALSE FLOAT IDN INT MAX MIN NOT STRING SUM TRUE ]
        ORDERBY =       [ ! - AVG FALSE FLOAT IDN INT MAX MIN NOT STRING SUM TRUE ]
        RIGHT   =       [ JOIN ]
        SELECT  =       [ # ) * ALL AVG DISTINCT IDN MAX MIN SUM UNION ]
        SET     =       [ IDN ]
        STRING  =       [ != # && ) , < <= <=> = > >= AND GROUPBY HAVING IS JOIN LEFT OR ORDERBY RIGHT UNION WHERE XOR || ]
```

部分预测表的截图：



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	PredicTab	!	!=	&&	(	)	*	,	-	.	<	<=	<=>	=	>	>=	ALL	AND	AS	AVG	DEFAULT	DELETE	DISTINCT
	aggregateWindowedFunction																				101.aggregateWindowedFunction-		
		101.aggregateWindowedFunction->function (unionType fullColumnName)															101.aggregateWindowedFunction->function (unionType fullColumnNam							
											101.aggregateWindowedFunction->function (unionType fullColumnName)													
	booleanLiteral																						
	comparisonOperator	90.comparisonOperator->!=																					
										87.comparisonOperator-><													
		88.comparisonOperator-><=																					
		91.comparisonOperator->(<=>)																					
		85.comparisonOperator->>=																					
		86.comparisonOperator->>																					
		89.comparisonOperator->>=																					
	constant									81.constant->- decimalLiteral													
																82.constant->booleanLiteral							
		80.constant->decimalLiteral																					
							80.constant->decimalLiteral																
																79.constant->stringLiteral							
		82.constant->booleanLiteral																					
	decimalLiteral																						
							84.decimalLiteral->INT																
	deleteStatement																						135.deleteSt
	deleteStatementRight			136.deleteStatementRight->whereExpression																			
		137.deleteStatementRight->(uidList) whereExpression																					
				0.deleteStatementRight->$																			
	dmlStatement					2.dmlStatement->selectStatement																	
																		5.dmlStatement->deleteStatement					
									3.dmlStatement->insertStatement														
															2.dmlStatement->selectStatement								
								4.dmlStatement->updateStatement															
	dottedId	0.dottedId	51.dottedId->$																				
		0.dottedId->$	0.dottedId->$	0.dottedId->$	0.dottedId->$		50.dottedId->. dottedIdOrStar																
		0.dottedId	0.dottedId	0.dottedId	0.dottedId	0.dottedId	0.dottedId->$	0.dottedId	0.dottedId->$								0.dottedId	0.dottedId	0.dottedId	0.dottedId->$			0.d
	dottedIdOrStar							53.dottedIdOrStar->*															
																							52.dottedId
	elementNameAlias			35.elementNameAlias->$																			
				0.elementNameAlias	0.elementNameAlias->$												34.elementNameAlias->AS uid							
								0.element	0.element	0.element	33.elementNameAlias->uid												
						0.element	0.elementNameAlias->$					0.elementNameAlia	0.element	0.elementNameAlia	0.elementNameAlias->$						0.elementNameAlias->$		
	expressic	57.expression->opposite expression																					