%{

#include <stdio.h>

#include <stdlib.h>

int yylex();

void yyerror(const char \*s);

struct Node

{

struct Node\* next;

struct Node \*cousin;

char s[150];

};

struct Node\* createNode(char\* s);

void printAST(struct Node\* root,int level);

%}

%start S

%token <node> ID NUM SELECT FROM WHERE LE GE EQ NE OR AND NOT SUM TABLE DROP CREATE UPDATE SET INSERT INTO VALUES OPERATOR '=' '>' '<' ')' '(' ',' '\*'

%type <node> S select ST3 ST4 ST5 ST6 ST7 op create insert update schema vallist columns tables attributes otheroptions COND E F options G

%union{

struct Node\* node;

}

%%

S : select';'

{

$$=createNode("S");

$$->next=$1;

printAST($$,0);

printf("---Success---\n");

exit(0);

};

| drop';'

{

$$=createNode("S");

$$->next=$1;

printAST($$,0);

printf("---Success---\n");

exit(0);

};

| create';'

{

$$=createNode("S");

$$->next=$1;

printAST($$,0);

printf("---Success---\n");

;exit(0);

};

| insert';'

{

$$=createNode("S");

$$->next=$1;

printAST($$,0);

printf("---Success---\n");

exit(0);

};

| update ';'

{

$$=createNode("S");

$$->next=$1;

printAST($$,0);

printf("---Success---\n");

exit(0);

};

select : SELECT options attributes FROM tables ST2 {

$$=createNode("select");

$1=createNode("SELECT");

$4=createNode("FROM");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

$3->cousin=$4;

$4->cousin=$5;

$5->cousin=$6;

}

;

options : otheroptions '(' ID ')' {

$$=createNode("options");

$2=createNode("(");

$3=createNode("ID");

$4=createNode(")");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

$3->cousin=$4;

}

| {

$$=createNode("options");

}

;

ST2 : WHERE COND ST3 {

$$=createNode("ST2");$1=createNode("WHERE");$$->next=$1;$1->cousin=$2;

$2->cousin=$3;}

| WHERE ID op '('select')'{

$$=createNode("ST2");

$1=createNode("WHERE");

$2=createNode("ID");

$4=createNode("(");

$6=createNode(")");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

$3->cousin=$4;

$4->cousin=$5;

$5->cousin=$6;

}

| ST3

{

$$=createNode("ST2");$$->next=$1;

}

;

ST3 : LIMIT NUM {

$$=createNode("ST3");

$1=createNode("LIMIT");

$2=createNode("NUM");

$$->next=$1;

$1->cousin=$2;

}

| ST4 {

$$=createNode("ST3");

$$->next=$1;

}

;

op : EXISTS {

$$=createNode("op");

$1=createNode("EXISTS");

$$->next=$1;

}

| OPERATOR ALL {

$$=createNode("op");

$1=createNode("OPERATOR");

$2=createNode("ALL");

$$->next=$1;

$1->cousin=$2;

}

| OPERATOR ANY {

$$=createNode("op");

$1=createNode("OPERATOR");

$2=createNode("ANY");

$$->next=$1;

$1->cousin=$2;

}

;

ST4 : GROUP attributes ST5 {

$$=createNode("ST4");

$1=createNode("GROUP");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

;

| ST5 {

$$=createNode("ST4");

$$->next=$1;

}

;

ST5 : HAVING COND ST6 {

$$=createNode("ST5");

$1=createNode("HAVING");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| ST6 {

$$=createNode("ST5");

$$->next=$1;

}

;

ST6 : ORDER ST7 {

$$=createNode("ST6");

$1=createNode("ORDER");

$$->next=$1;

$1->cousin=$2;

}

| {

$$=createNode("ST6---none");

}

;

ST7 : ID ST8',' ST7 {

$$=createNode("ST7");

$1=createNode("ID");

$3=createNode(",");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

$3->cousin=$4;

}

|ID ST8 {

$$=createNode("ST7");

$1=createNode("ID");

$$->next=$1;

$1->cousin=$2;

}

;

drop : DROP TABLE tables {

$$=createNode("drop");

$1=createNode("DROP");

$2=createNode("TABLE");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

;

create : CREATE TABLE ID'('columns')' {

$$=createNode("create");

$1=createNode("CREATE");

$2=createNode("TABLE");

$3=createNode("ID");

$4=createNode("(");

$6=createNode(")");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

$3->cousin=$4;

$4->cousin=$5;

$5->cousin=$6;

}

;

insert : INSERT INTO ID schema VALUES '('vallist')' {

$$=createNode("insert");

$1=createNode("INSERT");

$2=createNode("INTO");

$3=createNode("ID");

$5=createNode("VALUES");

$6=createNode("(");

$8=createNode(")");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

$3->cousin=$4;

$4->cousin=$5;

$5->cousin=$6;

$6->cousin=$7;

$7->cousin=$8;

}

;

schema: '('vallist')' {

$$=createNode("schema");

$1=createNode("(");

$3=createNode(")");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| { $$=createNode("schema");}

;

vallist : F','vallist {

$$=createNode("vallist");

$2=createNode(",");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| F {$$=createNode("vallist");$$->next=$1;}

;

columns : ID ID','columns {

$$=createNode("columns");

$1=createNode("ID");

$2=createNode("ID");

$3=createNode(",");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

$3->cousin=$4;

}

|ID ID {

$$=createNode("columns");

$1=createNode("ID");

$2=createNode("ID");

$$->next=$1;

$1->cousin=$2;

}

;

attributes : ID','attributes {

$$=createNode("attributelist");

$1=createNode("ID");

$2=createNode(",");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| '\*' {$$=createNode("attributelist");$1=createNode("\*"); $$->next=$1;}

| ID {

$$=createNode("attributelist");

$1=createNode("ID");

$$->next=$1;

}

;

otheroptions : SUM {

$$=createNode("otheroptions");

$1=createNode("SUM");

$$->next=$1;

}

;

tables : ID',' tables {

$$=createNode("tablelist");

$1=createNode("ID");

$2=createNode(",");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| ID {

$$=createNode("tablelist");

$1=createNode("ID");

$$->next=$1;

}

;

COND : COND OR COND {

$$=createNode("COND");

$2=createNode("OR");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| COND AND COND {

$$=createNode("COND");

$2=createNode("AND");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| NOT COND {

$$=createNode("COND");

$1=createNode("NOT");

$$->next=$1;

$1->cousin=$2;

}

| E {$$=createNode("COND"); $$->next=$1;}

;

E : F '=' F {

$$=createNode("E");

$2=createNode("==");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| F '<' F {

$$=createNode("E");

$2=createNode("<");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| F '>' F {

$$=createNode("E");

$2=createNode(">");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| F LE F {

$$=createNode("E");

$2=createNode("LE");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| F GE F {

$$=createNode("E");

$2=createNode("GE");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| F EQ F {

$$=createNode("E");

$2=createNode("EQ");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| F NE F {

$$=createNode("E");

$2=createNode("NE");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| F OR F {

$$=createNode("E");

$2=createNode("OR");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

| F AND F {

$$=createNode("E");

$2=createNode("AND");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

}

;

F : ID {

$$=createNode("F");

$1=createNode("ID");

$$->next=$1;

}

| NUM {

$$=createNode("F");

$1=createNode("NUM");

$$->next=$1;

}

;

update : UPDATE ID SET ST9 ST2

{

$$=createNode("update");

$1=createNode("UPDATE");

$2=createNode("ID");

$3=createNode("SET");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

$3->cousin=$4;

$4->cousin=$5;

};

ST9 : ID'='G ',' ST9

{

$$=createNode("ST9");

$1=createNode("ID");

$2=createNode("=");

$4=createNode(",");

$$->next=$1;

$1->cousin=$2;

$2->cousin=$3;

$3->cousin=$4;

$4->cousin=$5;

}

|ID'='G

{

$$=createNode("ST9");

$1=createNode("ID");

$2=createNode("=");

$$->next = $1;

$1->cousin=$2;

$2->cousin = $3;

};

G : ID

{

$$=createNode("G");

$1=createNode("ID");

$$->next=$1;

}

| NUM

{

$$=createNode("G");

$1=createNode("NUM");

$$->next=$1;

}

;

%%

#include"lex.yy.c"

struct Node\* createNode(char\* s)

{

struct Node\* node=malloc(sizeof(struct Node));

node->next=NULL;

node->cousin=NULL;

strcpy(node->s,s);

return node;

}

void printAST(struct Node \*root,int level)

{

if(root==NULL)

return;

for(int i=0;i<level-1;i++)

printf(" ");

printf("|----%s\n",root->s);

if(root->next!=NULL)

{

root=root->next;

while(root!=NULL)

{

printAST(root,level+1);

root=root->cousin;

}

}

}

int main()

{

printf("Query:\n");

yyparse();

return 0;

}