### **explication details de code**

### **Fichier .l (Analyseur lexical)**

lex

%{

#include<stdio.h>

#include<math.h>

#include"mini.tab.h"

#include <stdarg.h>

#include <string.h>

int yyerror(char \*s);

int oldstate;

int numTOK=0; int numID=0; int numNUM=0; int numOP=0; int numCAR=0;

int errlx=0;

%}

%option yylineno

* %{ ... %} : Délimite la section de déclarations de code C. Ces déclarations sont copiées directement dans le fichier de sortie de lex.yy.c.
* #include : Inclut les en-têtes nécessaires.
* int yyerror(char \*s); : Prototype de fonction pour yyerror.
* Déclarations de variables globales pour compter les jetons, les identifiants, les nombres, les opérateurs et les caractères spéciaux, ainsi qu'une variable pour les erreurs lexicales.
* %option yylineno : Active le suivi du numéro de ligne pour les jetons.

lex

LETTRE [a-zA-Z]

QUIT [Qq][Uu][Ii][Tt]

* Définition de macros pour une lettre et pour le mot-clé "QUIT".

lex

%%

{QUIT} {printf("\nStatistiques : \nMots cle: %d\nIDs: %d\nNumeros: %d\nOperateurs: %d\nCaracteres speciaux(\*,;): %d\nCaracteres inconnus : %d\n", numTOK,numID,numNUM,numOP, numCAR,errlx); return 0;}

* Règle pour détecter le mot-clé "QUIT" et afficher les statistiques.

lex

(c|C)(r|R)(e|E)(a|A)(t|T)(e|E) {++numTOK; return CREATE;}

(t|T)(a|A)(b|B)(l|L)(e|E) {++numTOK; return TABLE;}

(i|I)(n|N)(s|S)(e|E)(r|R)(t|T) {++numTOK; return INSERT;}

(d|D)(r|R)(o|O)(p|P) {++numTOK; return DROP;}

* Définition des règles pour les mots-clés CREATE, TABLE, INSERT et DROP.

lex

(i|I)(n|N)(t|T)(o|O) {++numTOK; return INTO;}

* Définition de la règle pour le mot-clé INTO.

lex

(v|V)(a|A)(l|L)(u|U)(e|E)(s|S) {++numTOK; return VALUES;}

* Définition de la règle pour le mot-clé VALUES.

lex

(d|D)(e|E)(l|L)(e|E)(t|T)(e|E) {++numTOK; return DELETE;}

(u|U)(p|D)(a|A)(t|T)(e|E) {++numTOK; return UPDATE;}

(f|F)(r|R)(o|O)(m|M) {++numTOK; return FROM;}

(s|S)(e|E)(l|L)(e|E)(c|C)(t|T) {++numTOK; return SELECT;}

(d|D)(a|A)(t|T)(a|A)(b|B)(a|A)(s|S)(e|E) {++numTOK; return DATABASE; }

* Définition des règles pour les mots-clés DELETE, UPDATE, FROM, SELECT et DATABASE.

lex

(v|V)(a|A)(r|R)(c|C)(h|H)(a|A)(r|R) {yylval.subtok=10; return TP; }

(n|N)(u|U)(m|M)(b|B)(e|E)(r|R) {yylval.subtok=11; return TP; }

(r|R)(e|E)(a|A)(l|L) { yylval.subtok=12; return TP; }

(i|I)(n|N)(t|T) { yylval.subtok=13; return TP; }

(d|D)(a|A)(t|T)(e|E) { yylval.subtok=14; return TP;}

* Définition des règles pour les types de données TP.

lex

(p|P)(r|R)(i|I)(m|M)(a|A)(r|R)(y|Y)" "(k|K)(e|E)(y|Y) { yylval.subtok=21; return CONDITION;}

(u|U)(n|N)(i|I)(q|Q)(u|U)(e|E) { yylval.subtok=22; return CONDITION;}

(f|F)(o|O)(r|R)(i|I)(e|E)(g|G)(n|N)" "(k|K)(e|E)(y|Y) { yylval.subtok=23; return CONDITION;}

(a|A)(u|U)(t|T)(o|O)" "(i|I)(n|N)(c|C)(r|R)(e|E)(m|M)(e|E)(n|N)(t|T)(e|E)(d|D) { yylval.subtok=24; return CONDITION;}

* Définition des règles pour les types de conditions.

lex

(w|W)(h|H)(e|E)(r|R)(e|E) {++numTOK; return WHERE;}

(s|S)(e|E)(t|T) {++numTOK; return SET;}

(o|O)(r|R)(d|D)(e|E)(r|R) {++numTOK; return ORDER;}

(g|G)(r|R)(o|O)(u|U)(p|P) {++numTOK; return GROUP;}

(b|B)(y|Y) {++numTOK; return BY;}

* Définition des règles pour les mots-clés WHERE, SET, ORDER, GROUP et BY.

lex

"(" {numCAR++; return PARAO;}

")" { numCAR++; return PARAF;}

* Règles pour les parenthèses ouvrantes et fermantes.

lex

(a|A)(v|V)(g|G) {++numTOK;return AVG;}

(s|S)(u|U)(m|M) {++numTOK;return SUM;}

(c|C)(o|O)(u|U)(n|N)(t|T) {++numTOK;return COUNT;}

(l|L)(i|I)(m|M)(i|I)(t|T) {++numTOK;return LIMIT;}

* Définition des règles pour les fonctions AVG, SUM, COUNT et LIMIT.

lex

(a|A)(n|N)(d|D) {++numTOK;return ANDOP;}

(o|O)(r|R) {++numTOK;return OR;}

* Définition des règles pour les opérateurs logiques AND et OR.

lex

"\*" {++numCAR; return ETOILE;}

"'" {++numCAR; return POSTROFE;}

* Définition des règles pour l'astérisque (\*) et l'apostrophe (').

lex

"=" {numOP++;return EQUALS;}

* Règle pour l'opérateur d'égalité (=).

lex

"<=>" {yylval.subtok = 2; return COMPARISON;}

">=" {yylval.subtok = 3; return COMPARISON;}

">" {yylval.subtok = 4; return COMPARISON;}

"<=" {yylval.subtok = 5; return COMPARISON;}

"<" {yylval.subtok = 6; return COMPARISON;}

"!=" {yylval.subtok = 7; return COMPARISON;}

* Définition des règles pour les opérateurs de comparaison.

lex

"." {++numCAR;return POINT;}

* Règle pour le point (.) comme caractère spécial.

lex

-?[0-9]+ { ++numNUM; yylval.intval = atoi(yytext); return INTNUM; }

* Règle pour les nombres entiers.

lex

({LETTRE})({LETTRE}|\\_)\* {++numID;return ID;return numID;}

* Règle pour les identifiants.

lex

";" {numCAR++;return FIN;}

[ \t\n] {}

"," {numCAR++;return SEP;}

* Règles pour les caractères spéciaux point-virgule (;), espace, tabulation et virgule (,).

lex

. {printf("Erreur lexicale %s a la ligne %d\n", yytext, yylineno);++errlx;}

* Règle par défaut pour gérer les caractères non reconnus.

lex

%%

* Délimiteur de la section de règles.

lex

int yywrap()

{return 1;}

* Fonction yywrap nécessaire pour la fin du fichier.

### **Fichier .y (Analyseur syntaxique)**

yacc

%{

#include<stdio.h>

int yylex(void);

int yyerror(char \*s);

%}

%union {

int intval;

int subtok;

}

%token QUIT;

%token ID;

%token DELETE;

%token UPDATE;

%token LIMIT;

%token FROM;

%token ORDER;

%token BY;

%token USING;

%token WHERE;

%token FIN;

%token ANDOP;

%token OR;

%token SET;

%left <subtok> COMPARISON /\* != < > <= >= <=> \*/;

%token POINT;

%token INT;

%token SEP;

%token GROUP;

%token INTNUM;

%token EQUALS;

%token POSTROFE;

%token ETOILE;

%token SELECT;

%token AVG;

%token SUM;

%token COUNT;

%token INSERT;

%token INTO;

%token CREATE;

%token TABLE;

%token DATABASE;

%token PARAO;

%token PARAF;

%token VALUES;

%token DROP;

%left <subtok> TP ;

%left <subtok> CONDITION ;

%%

* Déclarations initiales du fichier .y comprenant les en-têtes nécessaires, la définition des tokens et des unions.

yacc

D : DELETE FROM S1 FIN { printf("Suppression de toutes les lignes de la table \n");};

| DELETE FROM S1 WHERE C {printf("Suppression d'une ligne de la table \n");};

| UPDATE S1 SET S WHERE C {printf("Mise a jour avec succes d'une ligne \n");};

| SELECT ETOILE FROM S1 FIN {printf("Affichage de toutes les lignes de la table \n");};

| SELECT COL FROM S1 FIN {printf("Affichage des lignes choisi de la table \n");};

| SELECT COL FROM S1 WHERE C {printf("Affichage des lignes choisi de la table avec Conditions\n");};

| SELECT AVG PARAO S1 PARAF FROM S1 WHERE C {printf("AVG\n");};

| SELECT SUM PARAO S1 PARAF FROM S1 WHERE C {printf("SUM\n");};

| SELECT COUNT PARAO S1 PARAF FROM S1 WHERE C {printf("COUNT\n");};

| CREATE TABLE S1 PARAO T {printf("TABLE EST CREER \n");};

| CREATE DATABASE S1 FIN{printf("BASE DE DONNES EST CREER \n");};

| INSERT INTO S1 VALUES PARAO K {printf("Insertion d'une ligne dans la table \n");};

| DROP DATABASE S1 FIN {printf("Suppression de la Base de donnee \n");};

| DROP TABLE S1 FIN {printf("Suppression de la Table \n");};

| DELETE FROM S1 FIN D {printf("Suppression de toutes les lignes de la table \n");};

| DELETE FROM S1 WHERE C D {printf("Suppression d'une ligne de la table \n");};

| UPDATE S1 SET S WHERE C D {printf("Mise a jour avec succes d'une ligne \n");};

| SELECT ETOILE FROM S1 FIN D {printf("Affichage de toutes les lignes de la table \n");};

| SELECT COL FROM S1 FIN D {printf("Affichage des lignes choisi de la table \n");};

| SELECT COL FROM S1 WHERE C D{printf("Affichage des lignes choisi de la table avec Conditions\n");};

| SELECT AVG PARAO S1 PARAF FROM S1 WHERE C D{printf("AVG\n");};

| SELECT SUM PARAO S1 PARAF FROM S1 WHERE C D {printf("SUM\n");};

| SELECT COUNT PARAO S1 PARAF FROM S1 WHERE C D {printf("COUNT\n");};

| CREATE TABLE S1 PARAO T D {printf("TABLE EST CREER \n");};

| CREATE DATABASE S1 D {printf("BASE DE DONNES EST CREER \n");};

| INSERT INTO S1 VALUES PARAO K D {printf("Insertion d'une ligne dans la table \n");};

| DROP DATABASE S1 FIN D {printf("Suppression de la Base de donnee \n");};

| DROP TABLE S1 FIN D {printf("Suppression de la Table \n");};

* Définition des règles de grammaire pour les différentes commandes SQL.

yacc

K : S1 PARAF FIN

|S1 SEP K

|SEP S1 SEP K

|INTNUM PARAF FIN

|SEP INTNUM SEP K

|INTNUM SEP K

;

* Définition de la règle pour la séquence de valeurs dans une insertion.

yacc

T : S1 TP PARAF FIN

| S1 TP SEP T

| SEP S1 TP COND SEP T

| S1 TP COND SEP T

;

* Définition de la règle pour les types de données dans la création de table.

yacc

COND : CONDITION | CONDITION COND;

* Définition de la règle pour les conditions.

yacc

S1 : ID | ID S1 ;

* Définition de la règle pour les séquences de noms d'objets.

yacc

COL : S1 SEP COL

|S1

;

* Définition de la règle pour les colonnes dans les instructions SELECT.

yacc

S : S1 EQUALS S1

| S1 EQUALS INTNUM

| S1 EQUALS INTNUM SEP S

| S1 EQUALS S1 SEP S

| SEP S1 EQUALS INTNUM

| SEP S1 EQUALS S1

;

* Définition de la règle pour les attributions dans les instructions UPDATE.

yacc

C : S1 COMPARISON INTNUM FIN

| S1 COMPARISON S1 FIN

| S1 COMPARISON S1 ANDOP C

| S1 COMPARISON S1 OR C

| S1 COMPARISON INTNUM ANDOP C

| S1 COMPARISON INTNUM OR C

| S1 EQUALS INTNUM FIN

| S1 EQUALS S1 FIN

| S1 EQUALS S1 ANDOP C

| S1 EQUALS S1 OR C

| S1 EQUALS INTNUM ANDOP C

| S1 EQUALS INTNUM OR C

;

* Définition de la règle pour les conditions dans les instructions WHERE.

yacc

%%

* Délimiteur de la section des règles.

yacc

#include "lex.yy.c"

* Inclusion du fichier généré par flex.

yacc

int yyerror(char \*s){printf (" ERREUR syntaxique a la ligne %d \n ", yylineno );return (0);}

* Définition de la fonction yyerror.

yacc

void main(){

printf(" debut de l'analyse lexicale \n ");

if(!yyparse()) // lance l'analyse syntaxique

printf("FIN de L'analyse synaxique");getchar();

}

* Fonction principale qui démarre l'analyse lexicale et syntaxique.

yacc

%{

#include<stdio.h>

#include<math.h>

#include"mini.tab.h"

#include <stdarg.h>

#include <string.h>

int yyerror(char \*s);

int oldstate;

int numTOK=0; int numID=0; int numNUM=0; int numOP=0; int numCAR=0;

int errlx=0;

%}

* Déclarations de code C incluses dans le fichier généré.