

Institut Supérieur  
d'Informatique et des  
Techniques de Communication  
Hammam Sousse



## Compilation

### TP 3 : Compilateur SQL

Réalisé par :  
Elagas Amel

Encadré par :  
M. Hachem Moez

**2DNI-G1**

Année universitaire : 2020 – 2021

## **Sommaire :**

### **Objectif :**

#### **I. Étude théorique**

1. Compilateur
2. Analyse lexicale
3. Lex
4. Analyse syntaxique
5. Yacc

#### **II. Travail demandé**

##### **Partie 1 : Exemple traité dans la classe**

1. Fichier yacc
2. Fichier lex
3. Les requêtes
4. Résultat de l'exécution

##### **Partie 2 : Compilateur SQL**

1. Fichier yacc
2. Fichier lex
3. Les requêtes
4. Résultat de l'exécution

## **Conclusion**

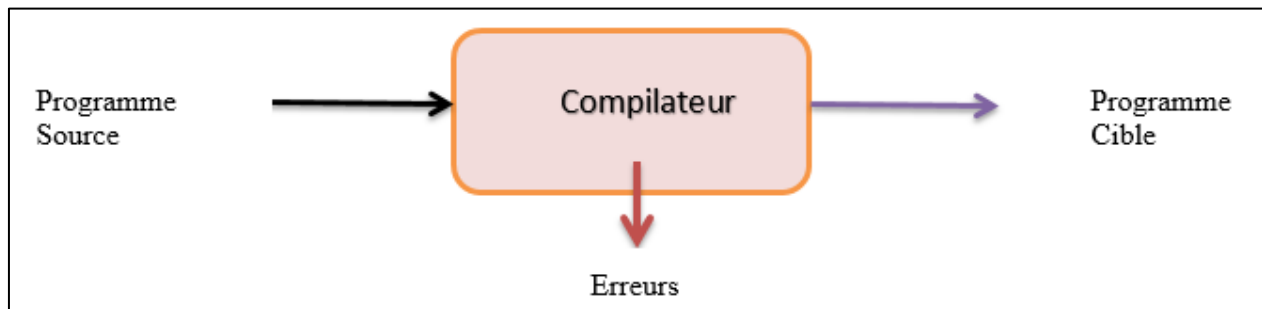
## Objectif :

L'objectif de ce TP est de créer un mini compilateur SQL, en utilisant l'analyse lexicale et l'analyse syntaxique.

### I. Etude Théorique

#### 1. Compilateur :

Un compilateur est un programme informatique qui transforme un code source écrit dans un langage (le langage source) en un autre langage (le langage cible) en indiquant les erreurs éventuelles que pourrait contenir le programme source.



#### 2. Analyse lexicale :

L'analyseur lexical constitue la première phase d'un compilateur. Sa tâche principale est de lire les caractères d'entrée et de produire comme résultat une suite d'unités lexicales que l'analyseur syntaxique va utiliser. Il lit les caractères d'entrée jusqu'à ce qu'il puisse identifier la prochaine unité lexicale. La spécification des unités lexicales est effectuée à l'aide d'expressions régulières.

#### 3. Lex :

Lex est un générateur d'analyseur lexical : il prend en entrée une description lexicale (càd, un ensemble de couples expression-régulière/action) et produit une fonction d'analyse lexicale. Cette fonction est écrite en C et s'appelle yylex(). Elle peut ensuite être intégrée à n'importe quel programme.

#### 4. Analyse syntaxique :

L'analyse syntaxique a pour but :

- De vérifier que le texte d'entrée est conforme à la grammaire.
- D'indiquer les erreurs de syntaxe et éventuellement de poursuivre l'analyse après une erreur (reprise sur erreur).
- De construire une représentation intermédiaire pour les autres modules du compilateur.

#### 5. Yacc :

Yacc (**Y**et **A**nother **C**ompiler **C**ompiler) est un programme destiné à compiler une grammaire et à produire le texte source d'un analyseur syntaxique du langage engendré par cette grammaire. Il est aussi possible, en plus de la vérification de la syntaxe de la grammaire, de lui faire effectuer des actions sémantiques.

## II. Travail demandé :

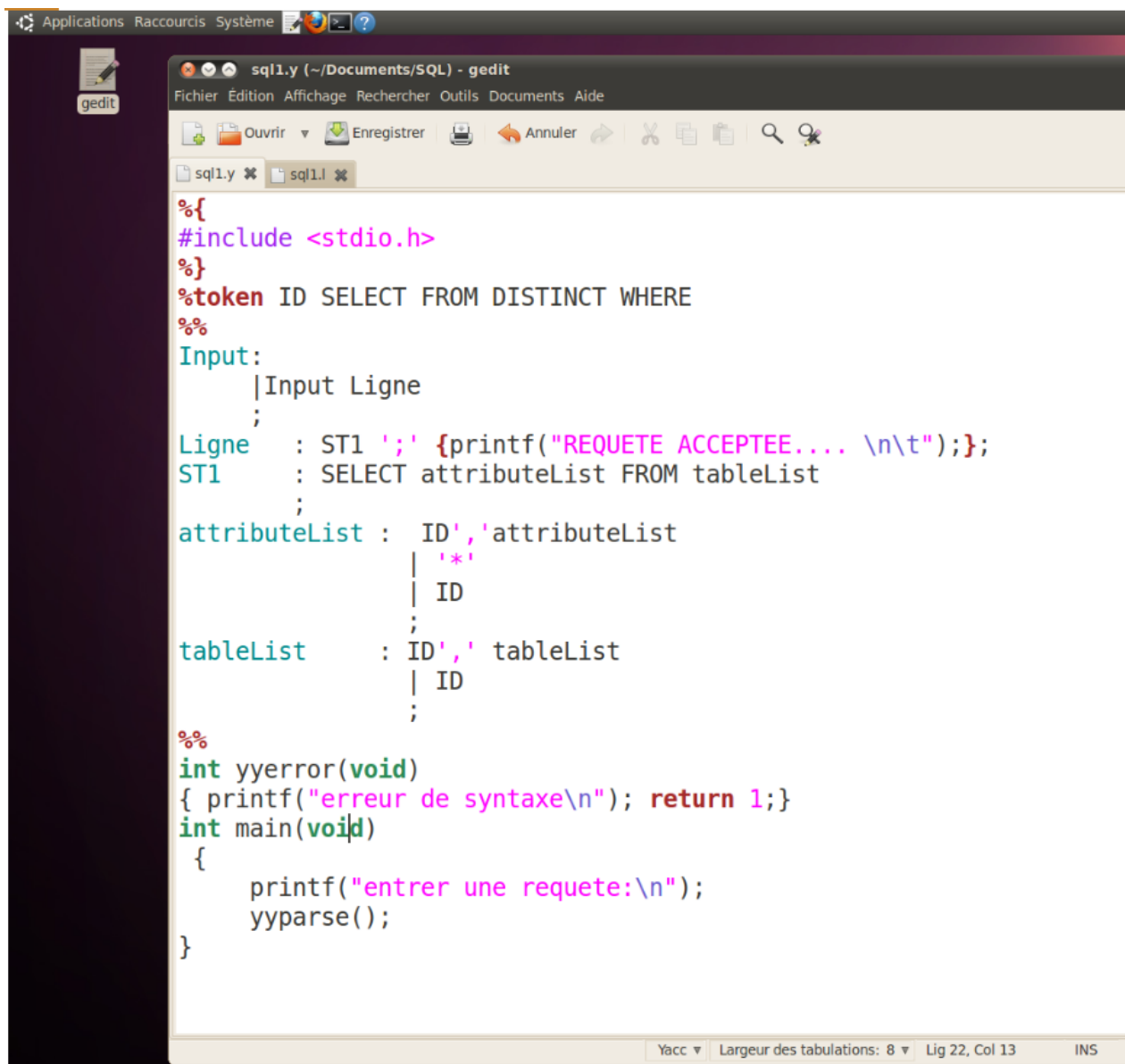
On va créer un compilateur de langage SQL. En effet, *SQL* pour « Structured Query Language » en français langage de requête structurée est un langage informatique normalisé servant à exploiter des bases de données relationnelles.

On a utilisé l'outil Lex pour l'analyse lexicale et l'outil Yacc pour l'analyse syntaxique.

# Partie 1 : Exemple traité dans la classe :

## 1. Fichier yacc :

*Sql1.y*



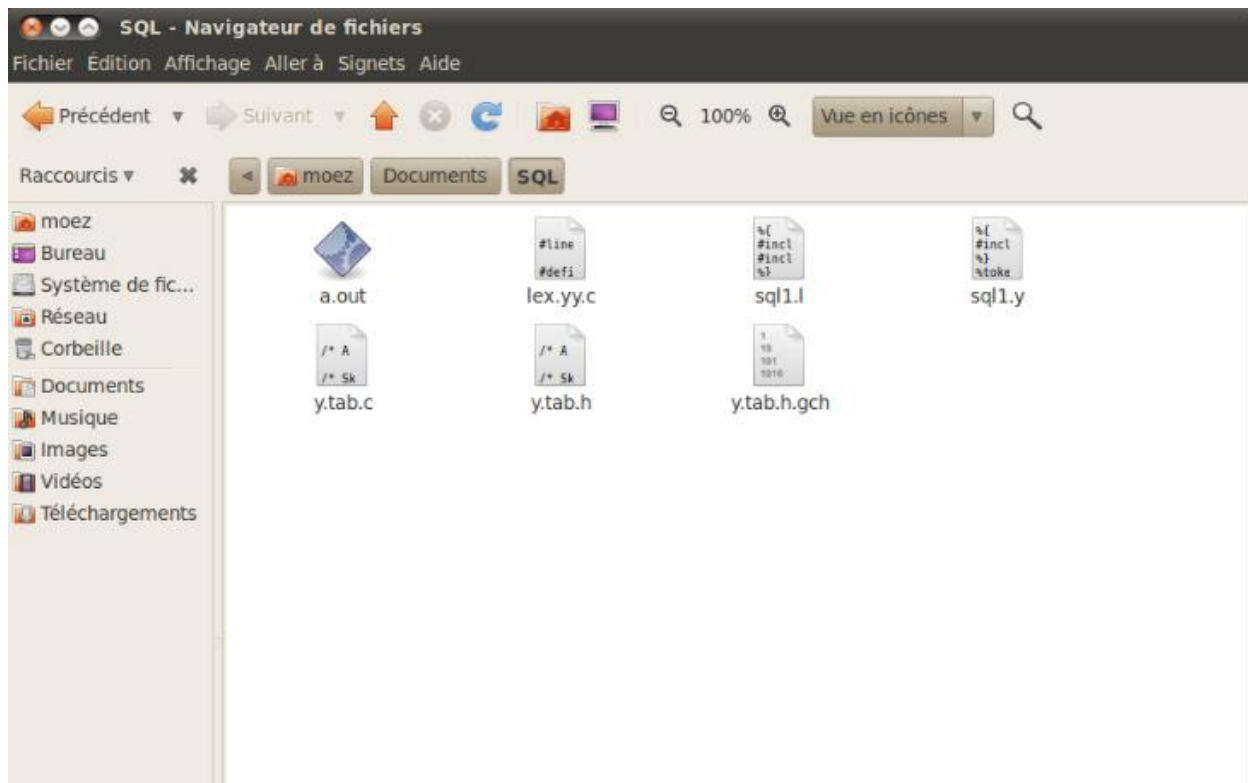
```
%{
#include <stdio.h>
%}
%token ID SELECT FROM DISTINCT WHERE
%%
Input:
    | Input Ligne
    ;
Ligne  : ST1 ';' {printf("REQUETE ACCEPTEE.... \n\t");};
ST1    : SELECT attributeList FROM tableList
        ;
attributeList : ID',' attributeList
                | '*'
                | ID
                ;
tableList    : ID',' tableList
                | ID
                ;
%%
int yyerror(void)
{ printf("erreur de syntaxe\n"); return 1;}
int main(void)
{
    printf("entrer une requete:\n");
    yyparse();
}
```

## 2. Fichier lex :

### Sql1.l

```
%{
#include <stdio.h>
#include "y.tab.h"
}%
alpha [A-Za-z]
digit [0-9]
%%
[ \t\n]
;
"select"
return SELECT;
"from"
return FROM;
"distinct"
return DISTINCT;
"where"
return WHERE;
{alpha}({alpha}|{digit})*
return ID;
.
return yytext[0];
%%
int yywrap(){
return 0;
}
```

## Les fichiers générés



### 3. Les requêtes

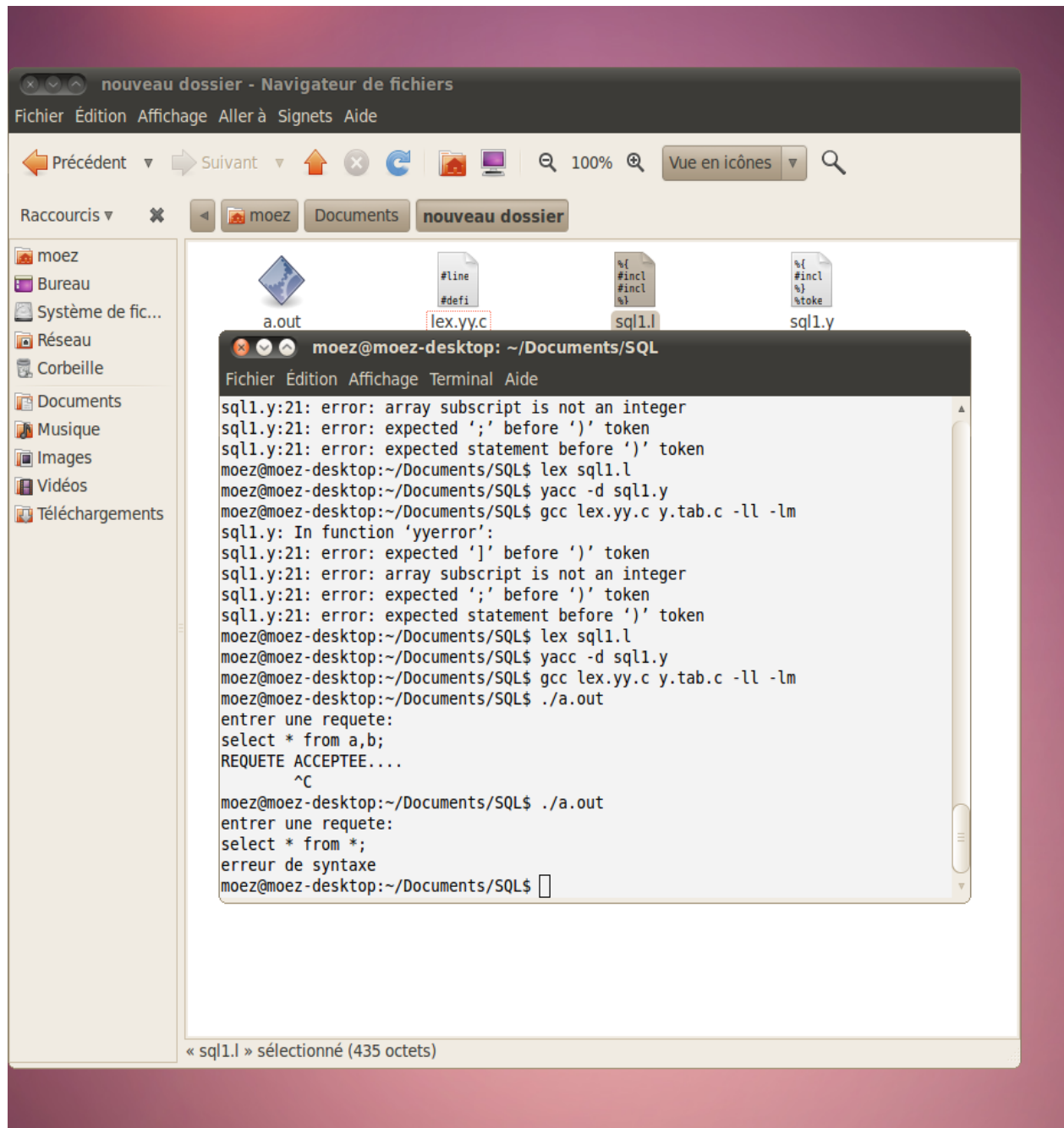
Les requêtes à exécuter :

requête - Bloc-notes

Fichier Édition Format Affichage Aide

```
select * from a,b ;  
select * from * ;
```

## 4. Résultat de l'exécution





## Partie 2 : Compilateur SQL

Dans les codes ci-dessus nous avons défini les règles nécessaires pour analyser et valider une expression qui contient une requête SQL mathématique. Les règles sont présentées dans les deux fichiers LEX et YAAC (**req.l** et **req.y**) qui permet de valider un ensemble des requêtes donner dans un fichier (requête).

### 1. Fichier LEX

*req.l*

```
req.l  req.y
/*Déclarations et définitions pour le programme C */
%{
//declarations des bibliotheques
#include <stdio.h>
#include <string.h>
#include "y.tab.h"
%}
alpha [a-zA-Z]
Digit [0-9]+
%%
[ \t] {}
\n return FIN;
distinct {return DISTINCT;}
add {return ADD;}
alter {return ALTER;}
references {return REFERENCES;}
foreign {return FOREIGN;}
constraint {return CONSTRAINT;}
primary {return PRIMARY;}
key {return KEY;}
select {return SELECT;}
create {return CREATE;}
drop {return DROP;}
table {return TABLE;}
from {return FROM;}
where {return WHERE;}
like {return LIKE;}
desc {return DESC;}
asc {return ASC;}
grant {return GRANT;}
group {return GROUP;}
order {return ORDER;}
by {return BY;}
or {return OR;}
```

```

or {return OR;}
and {return AND;}
insert {return INSERT;}
into {return INTO;}
values {return VALUES;}
update {return UPDATE;}
delete {return DELETE;}
set {return SET;}
int {return DATATYPE;}
date {return DATATYPE;}
time {return DATATYPE;}
year {return DATATYPE;}
smallint {return DATATYPE;}
number {return DATATYPE;}
float {return DATATYPE;}
varchar {return DATATYPE;}
avg {return AVG;}
count {return COUNT;}
first {return FIRST;}
last {return LAST;}
{Digit} {return NUM;}
{alpha}({alpha}|{Digit})* {return ID;}
"<=" {return LE;}
">=" {return GE;}
"==" {return EQ;}
"!=" {return NE;}
";" {return COL;}
"$" {return DOLLAR ;}
. {return yytext[0];}
%%
//Fonctions et programme principal
int yywrap()
{
    return 1;
}

```

## 2. Fichier yacc :

req.y



```

/***** Section de définition *****/
%{
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
extern int yylineno;
extern char* yytext;
extern FILE *yyin;
%}

//Abréviations d'expressions régulières
%token SELECT CREATE DROP TABLE FROM DOLLAR ASC DESC DISTINCT
%token ALTER ADD GROUP BY AVG COUNT FIRST LAST MAX MIN SUM NUME IDENT
%token WHERE LIKE INSERT INTO VALUES GRANT ON TO PRIMARY KEY CONSTRAINT
%token UPDATE DELETE DATATYPE NUM ID SET ORDER COL FOREIGN REFERENCES
%token FIN

%left '>' '<' LE GE EQ NE
%right '='
%left AND OR
%start program
%%

//Expressions régulières et actions associées
program:
| program Ligne
;
Ligne:
FIN
| S FIN      { printf("Requete correcte \n"); }
| error FIN  { yyerrok; }
;

S :ST1
  |ST3
  |ST4

```

```

Ouvrir Enregistrer Annuler
req.l req.y

|ST5
|ST7
|ST8
|ST9
|
;

/*****Select*****/
ST1 : SELECT attributeList FROM tablename ST2 ST6 { printf("Requete correcte \n"); }
| SELECT function('attributeList') FROM tablename { printf ("Requete correcte: \n \t fonction effectuée
avec succès!\n");}
| SELECT DISTINCT attributeList FROM tablename ST2 ST6 { printf("Requete correcte \n"); }
| SELECT attributeList FROM tablename ST2 GROUP BY attributeList { printf("Requete correcte \n"); }
| SELECT attributeList error { printf("syntax error : \n \t Mot clé FROM manquant !!!\n");}
| SELECT error { printf ("syntax error : \n \t Requete incomplete !!!\n");}
| SELECT attributeList FROM error {printf("syntax error : \n\t Nom de la table manquant apres from ou nom
de table invalid !!!\n");}

;

/*****FONCTIONS*****/
function: AVG
|COUNT
|FIRST
|LAST
|
;

/*****CREATE*****/
ST3:CREATE TABLE ID('listesdeschamps') {printf("Requete correcte: \n\t table crée avec succès\n");}
|CREATE TABLE error {printf("syntax error : \n\t nom du table manquant !!!\n");}
;

/*****DROP*****/
ST4:DROP TABLE ID {printf("Requete correcte:\n\t table supprimée \n");}
|DROP TABLE error {printf("syntax error : \n \t nom du table manquant !!!\n");}
;

/*****ALTER TABLE*****/
ST5:ALTER TABLE ID ADD PRIMARY KEY '('ID')' {printf("Requete correcte:\n\t clé primaire ajoutée \n");}
|ALTER TABLE error {printf("syntax error : \n\t nom du table manquant !!!\n");}

```



```

DROP TABLE error {printf("syntax error : \n \t nom du table manquant !!!\n");}

/*****ALTER TABLE*****/
ST5: ALTER TABLE ID ADD PRIMARY KEY ('ID') {printf("Requete correcte:\n\t clé primaire ajoutée \n");}
ALTER TABLE error {printf("syntax error : \n\t nom du table manquant !!!\n");}
DROP TABLE ID error {printf("syntax error : \n\t requette manquante !!!\n");}
ALTER TABLE ID ADD CONSTRAINT ID PRIMARY KEY('attributeList') {printf("Requete correcte:\t liste des
clés primaires modifiée \n");}
ALTER TABLE ID ADD FOREIGN KEY ('ID') REFERENCES tablename('ID') {printf("Requete correcte:\t clé
étrangère ajoutée \n");}
ALTER TABLE ID DROP PRIMARY KEY {printf("Requete correcte:\n\t clé primaire supprimée \n");}
ALTER TABLE ID DROP CONSTRAINT ID {printf("Requete correcte:\n\t clé supprimée \n");}

F : ID
  | NUM
  ;
VALEUR:F', 'VALEUR
  | F
  ;

/*****INSERT*****/
ST7: INSERT INTO tablename VALUES ('VALEUR') {printf("Requete correcte: \n\t valeur(s) insérée(s) \n");}
INSERT INTO tablename ('attributeList') VALUES ('VALEUR') {printf("Requete correcte: \n\t valeur(s)
insérée(s) dans les colonnes \n");}

/*****Update*****/
ST8: UPDATE tablename SET attributeList=' VALEUR ST2 {printf("Requete correcte: \n\t valeur(s) mise(s) à jour
\n");}

/*****DELETE*****/
ST9: DELETE FROM tablename ST2 { printf ("Requete correcte:\n\t colonne supprimée \n");}

listesdeschamps:ID DATATYPE', 'listesdeschamps
  | ID DATATYPE { printf ("Requete correcte\n");}
  | ID {printf("syntax error \n \t type inconnu !!!\n");}
  | ID DATATYPE', ' error {printf("syntax error \n\t Champs manquant !!!\n");}
  | ID error {printf("syntax error \n\t type manquant !!!\n");}
  | PRIMARY KEY('ID') {printf("Requete correcte:\n\t clé primaire attribuée\n");}
  | CONSTRAINT ID PRIMARY KEY('attributeList') {printf("Requete correcte:\n\t clé primaire attribuée\n");}
  | FOREIGN KEY('ID') REFERENCES tablename('ID') {printf("Requete correcte:\n\t clé étrangère attribuée
\n");}

attributeList : ID', 'attributeList
  | ID
  | ID', ' error {printf("syntax error: \n\t Champs manquant !!!\n");}
  | '*'
  ;

tablename : ID', 'tablename
  | ID
  | ID', ' error {printf("syntax error : \n\t nom de la table manquant !!!\n");}

/*****where*****/
ST2 : WHERE COND { printf("Requete correcte \n"); }
  | WHERE error { printf("syntax error \n\t erreur dans la condition\n");}
  | { printf("Requete correcte \n"); }

/*****order by ascendant et descendant*****/
ST6 : ORDER BY attributeList DESC {printf("\t trie décroissante\n");}
  | ORDER BY attributeList ASC {printf("\t trie croissante\n");}
  | ORDER BY attributeList ASC ', 'attributeList DESC
  | ORDER BY attributeList DESC ', 'attributeList ASC
  | { printf("Requete correcte \n"); }

```

```

listesdeschamps:ID DATATYPE', 'listesdeschamps
  | ID DATATYPE { printf ("Requete correcte\n");}
  | ID {printf("syntax error \n \t type inconnu !!!\n");}
  | ID DATATYPE', ' error {printf("syntax error \n\t Champs manquant !!!\n");}
  | ID error {printf("syntax error \n\t type manquant !!!\n");}
  | PRIMARY KEY('ID') {printf("Requete correcte:\n\t clé primaire attribuée\n");}
  | CONSTRAINT ID PRIMARY KEY('attributeList') {printf("Requete correcte:\n\t clé primaire attribuée\n");}
  | FOREIGN KEY('ID') REFERENCES tablename('ID') {printf("Requete correcte:\n\t clé étrangère attribuée
\n");}

attributeList : ID', 'attributeList
  | ID
  | ID', ' error {printf("syntax error: \n\t Champs manquant !!!\n");}
  | '*'
  ;

tablename : ID', 'tablename
  | ID
  | ID', ' error {printf("syntax error : \n\t nom de la table manquant !!!\n");}

/*****where*****/
ST2 : WHERE COND { printf("Requete correcte \n"); }
  | WHERE error { printf("syntax error \n\t erreur dans la condition\n");}
  | { printf("Requete correcte \n"); }

/*****order by ascendant et descendant*****/
ST6 : ORDER BY attributeList DESC {printf("\t trie décroissante\n");}
  | ORDER BY attributeList ASC {printf("\t trie croissante\n");}
  | ORDER BY attributeList ASC ', 'attributeList DESC
  | ORDER BY attributeList DESC ', 'attributeList ASC
  | { printf("Requete correcte \n"); }

```

```

;
COND: COND OR COND
      | COND AND COND
      | E { printf("Requete correcte \n"); }
;
E : F '=' F
  | F '=' ' ' F
  | F '<' F
  | F '>' F
  | F LE F
  | F GE F
  | F EQ F
  | F NE F
  | F OR F
  | F AND F
  | F LIKE F
;

%%
/***** Section du code C *****/
//Fonctions et programme principal
FILE *yyin;

int yyerror(char *s) {
    // printf("%s\n",s);
}

int main(void) {
    yyin = fopen("requete", "r");
    yyparse();
    return 0;
}

```

### 3. Requêtes

requete - Bloc-notes

Fichier Edition Format Affichage Aide

```

create table admin (userid , username );
DELETE FROM user where userid =3;
INSERT INTO user (username) VALUES (amel);
update user set username=amel woo where id=1 ;
SELECT * FROM user;
create table films (idFilm number, titre varchar,realisateur var
select ;

```

## 5. Résultat de l'exécution :

```
moez@moez-desktop:~/Documents/SQL2$ lex req.l
moez@moez-desktop:~/Documents/SQL2$ yacc -d req.y
moez@moez-desktop:~/Documents/SQL2$ gcc lex.yy.c y.tab.c -ll -lm
moez@moez-desktop:~/Documents/SQL2$ ls
a.out lex.yy.c nouveau fichier req.l req.y y.tab.c y.tab.h
moez@moez-desktop:~/Documents/SQL2$ ./a.out
create table admin (userid , username );
DELETE FROM user where userid =3;
INSERT INTO user (username) VALUES (amel);
update user set username=amel woo where id=1 ;
SELECT * FROM user;
create table films (idFilm number, titre varchar,realisateur varchar);
select ;

syntax error
      type manquant !!!
syntax error :
      nom du table manquant !!!
Requete correcte
Requete correcte:
      valeur(s) mise(s) à jour
Requete correcte
Requete correcte:
      table crée avec succès
syntax error :
      Requete incomplete !!!
```

SQL2 - Navigateur de ... req.l (~/Documents/S... SQL2 - Navigateur de ... moez@moez-desktop



### III. Conclusion :

Nous avons réalisé un mini compilateur SQL avec les outils Lex et Yacc. Nous avons donc créé pour cela des fichiers **.l** et **.y**. Ensuite, nous avons les compilés avec **Lex** et **Yacc**. Ces derniers nous permettent de générer des fichiers **.c** exécutés par la suite avec **gcc** et finalement, on a testé notre mini compilateur.