Université Aboubekr BELKAID
كلية العلوم – تيجاني هدام
Faculté des Sciences
قسم الإعلام الآلي
Département d'informatique

Module : Compilation **Année universitaire 2021-2022**
Classe : L3- Informatique
Réalisé par : Mr MERZOUG Mohamed
Mr ETCHIALI Abdelhak

TP 5
ANALYSE SYNTAXIQUE

Etape1 : PREPARATION DE LA TABLE D'ANALYSE

1.1- Définition de la grammaire : elle est constituée de :

- **Les terminaux**

SELECT	FROM	WHERE	DISTINCT	COUNT	IDENT	NOMBRE	OP_ARTH	SEP_,	SEP_;	OP_REL	OP_LOG	PAR_(	PAR_)

- **Les non-terminaux**

S (axiome)
A
B
C
C1
D
E
E1
F
G
G1

Les règles de production

- S -> SELECT A FROM C D SEP_;
- A-> COUNT PAR_(B PAR_) | B
- B-> DISTINCT C | C
- C-> IDENT C1
- C1->SEP_ , B | EPSILONE
- D-> WHERE E | EPSILONE
- E-> F E1
- E1-> OP_LOG E | EPSILONE
- F-> G OP_REL G
- G-> PAR_(G PAR_) | IDENT G1 | NOMBRE G1
- G1-> OP_ARTH G | EPSILONE

1.2- Codage et sauvegarde des règles de production :

a- Affecter **des codes** pour chaque **terminal.**

Utiliser les codes affectés dans l'étape de l'Analyse Lexicale.

b- Affecter **des codes** pour chaque **non-terminal**.

```
#define VN_S    0
#define VN_A    1
#define VN_B    2
#define VN_C    3
#define VN_C1   4
#define VN_D    5
#define VN_E    6
#define VN_E1   7
#define VN_F    8
#define VN_G    9
#define VN_G1 10
```

c- Affecter **un code** pour chaque **règle de production**.

règle de production	Code
S -> SELECT A FROM C D SEP_;	0
A-> COUNT PAR_(B PAR_)	1
A-> B	2
B-> DISTINCT C	3
B-> C	4
C-> IDENT C1	5
C1-> SEP_ , B	6
C1-> EPSILONE	7
D-> WHERE E	8
D-> EPSILONE	9
E-> F E1	10
E1-> OP_LOG E	11

E1 -> EPSILONE	12
F-> G OP_REL G	13
G-> PAR_(G PAR_) G1	14
G-> IDENT G1	15
G-> NOMBRE G1	16
G1-> OP_ARTH G	17
G1-> EPSILONE	18

d- Sauvegarder toutes **les règles de production** dans une matrice sous la forme suivante :

0	268	1	269	3	5	264	-1
1	271	265	2	266	-1	-1	-1
2	2	-1	-1	-1	-1	-1	-1
3	272	3	-1	-1	-1	-1	-1
4	3	-1	-1	-1	-1	-1	-1
5	260	4	-1	-1	-1	-1	-1
6	263	2	-1	-1	-1	-1	-1
7	35	-1	-1	-1	-1	-1	-1
8	270	6	-1	-1	-1	-1	-1
9	35	-1	-1	-1	-1	-1	-1
10	8	7	-1	-1	-1	-1	-1
11	273	6	-1	-1	-1	-1	-1
12	35	-1	-1	-1	-1	-1	-1
13	9	267	9	-1	-1	-1	-1
14	265	9	266	10	-1	-1	-1
15	260	10	-1	-1	-1	-1	-1
16	261	10	-1	-1	-1	-1	-1
17	262	9	-1	-1	-1	-1	-1
18	35	-1	-1	-1	-1	-1	-1

Où :

- Chaque **ligne de la matrice** représente **une règle de production**.
- Chaque **cellule de la matrice** contient **le code d'un terminal** ou **d'un non-terminal**.
- **Le code -1** indique la fin **d'une règle de production**.
- **Exemple de code source:**

```
int RPs[19][7]={
        {Code_MC_SELECT,VN_A,Code_MC_FROM,VN_C1,VN_D,Code_SEP_PTVIRG,-1}, //.....0
        {Code_MC_COUNT,Code_ACCOLADE_OVR,VN_B,Code_ACCOLADE_FER,-1,-1,-1},  //.....1
        {VN_B,-1,-1,-1,-1,-1,-1},   //.....2
        {Code_MC_DISTINCT,VN_C1,-1,-1,-1,-1,-1}, //.....3
        {VN_C1,-1,-1,-1,-1,-1,-1}, //.....4
        {Code_IDENT,VN_C2,-1,-1,-1,-1,-1}, //.....5
        {Code_SEP_VIRG,VN_B,-1,-1,-1,-1,-1}, //.....6
        {EPSILONE,-1,-1,-1,-1,-1,-1}, //.....7
        {Code_MC_WHERE,VN_E1,-1,-1,-1,-1,-1}, //.....8
        {EPSILONE,-1,-1,-1,-1,-1,-1}, //.....9
        {VN_F,VN_E2,-1,-1,-1,-1,-1}, //.....10
        {Code_OP_LOG,VN_E1,-1,-1,-1,-1,-1}, //.....11
        {EPSILONE,-1,-1,-1,-1,-1,-1}, //.....12
        {VN_G1,Code_OP_REL,VN_G1,-1,-1,-1,-1}, //.....13
        {Code_ACCOLADE_OVR,VN_G1,Code_ACCOLADE_FER,VN_G2,-1,-1,-1}, //.....14
        {Code_IDENT,VN_G2,-1,-1,-1,-1,-1}, //.....15
        {Code_NOMBRE,VN_G2,-1,-1,-1,-1,-1}, //.....16
        {Code_OP_ARTH,VN_G1,-1,-1,-1,-1,-1}, //.....17
        {EPSILONE,-1,-1,-1,-1,-1,-1}, //.....18
};
```

1.3- Calcul du First et Follow :

a- Pour chaque **non-terminal R**, calculer ***First(R)*** afin de remplir l'entré correspondante de la Table d'Analyse ***(TA).***

b- En cas où le **non-terminal** produit **Epsilone**, calculer ***Follow(A)*** est le rajouté au résultat précédent.

Calcul:

***TA (S)*= First(S) ={SELECT}**

***TA (A)*= First(A) ={COUNT, DISTINCT, IDENT}**

***TA (B)*= First(B) ={DISTINCT, IDENT}**

***TA (C)*= First(C) ={IDENT}**

***TA (C1)*= First(C1)U Follow(C1) ={SEP_, , FROM, WHERE, SEP_;, PAR_)}**

***TA (D)*= First(D)U Follow(D)= {WHERE, SEP_;}**

***TA (E)*= First(E)= {PAR_(, IDENT, NOMBRE}**

***TA (E1)*= First(E1)U Follow(E1)={OP_LOG , SEP_;}**

***TA (F)*= First(F)={ PAR_(, IDENT , NOMBRE}**

***TA (G)*= First(G)={ PAR_(, IDENT , NOMBRE}**

***TA (G1)*= First(G1)U Follow(G1)= {,OP_ARTH , OP_REL , SEP_; , PAR_), OP_LOG }**

1.4- Construction et remplissage de la Table d'Analyse (TA)

La Table d'Analyse (TA) est représentée sous forme d'une **matrice** où :

- **Chaque ligne** représente **un non-terminal**.
- **Chaque colonne** représente **un terminal**.

Si ***First*** ou ***Follow*** **d'un non-terminal** correspond à **un terminal** (ou plusieurs), **la cellule** correspondante dans **la Table d'Analyse (TA)** est remplie par le code de **la règle de production** qui a généré ce **terminal**.

Les cellules vides sont remplies par **un code d'erreur** : **#define ERREUR -1**

Table d'Analyse

	SELECT	FROM	WHERE	DISTINCT	COUNT	IDENT	NOMBRE	OP_ARTH	SEP_,	SEP_;	OP_REL	OP_LOG	PAR_(	PAR_)	#
S	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
A	-1	-1	-1	2	1	2	-1	-1	-1	-1	-1	-1	-1	-1	-1
B	-1	-1	-1	3	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1
C	-1	-1	-1	-1	-1	5	-1	-1	-1	-1	-1	-1	-1	-1	-1
C1	-1	7	7	-1	-1	-1	-1	-1	6	7	-1	-1	-1	7	-1
D	-1	-1	8	-1	-1	-1	-1	-1	-1	9	-1	-1	-1	-1	-1
E	-1	-1	-1	-1	-1	10	10	-1	-1	-1	-1	-1	10	-1	-1
E1	-1	-1	-1	-1	-1	-1	-1	-1	-1	12	-1	11	-1	-1	-1
F	-1	-1	-1	-1	-1	13	13	-1	-1	-1	-1	-1	13	-1	-1
G	-1	-1	-1	-1	-1	15	16	-1	-1	-1	-1	-1	14	-1	-1
G1	-1	-1	-1	-1	-1	-1	-1	17	-1	18	18	18	-1	18	-1

- **Exemple de code source:**

```
int Table_DAnalyse [11][15]={
{0,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR}, //S
{ERREUR,ERREUR,ERREUR,2,1,2,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR}, //A
{ERREUR,ERREUR,ERREUR,3,ERREUR,4,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR}, // B
{ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,5,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR}, //C
{ERREUR,7,7,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,6,7,ERREUR,ERREUR,ERREUR,7,ERREUR}, // C1
{ERREUR,ERREUR,8,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,9,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR}, // D
{ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,10,10,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,10,ERREUR,ERREUR}, // E
 {ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,12,ERREUR,11,ERREUR,ERREUR,ERREUR},// E1
{ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,13,13,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,13,ERREUR,ERREUR}, // F
 {ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,15,16,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,14,ERREUR,ERREUR}, // G
 {ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,ERREUR,17,ERREUR,18,18,18,ERREUR,18,ERREUR} // G1
};
```

Etape2 : PROGRAMMATION DE L'ANALYSEUR SYNTAXIQUE (LL1)

2.1- Structure de données:

On va utiliser deux structures de données :

- **Première structure** : c'est une **Liste Chainée** des **Unités Lexicales** générée par **l'Analyseur Lexical** qui se termine par « **#** ».
- **Deuxième structure** : c'est une **Pile** pour **l'Analyseur Syntaxique** initialisée à « **S #** ».
- **Remarque** : la structure des éléments de la **Pile** est identique à celle de **Liste Chainée** des **Unités Lexicales**.

2.2- L'Analyse Syntaxique :

Faire dérouler **l'Algorithme d'Analyse Syntaxique LL1** suivant :

PileAnalyseurSyntaxique=**S #** //Initialisation de la pile de l'analyseur syntaxique

CopieSuiteUL= une copie de la SuiteUL produit par l'analyseur lexicale

Répété tant que (Tete(**CopieSuiteUL**) ->Code !=# && Tete(**PileAnalyseurSyntaxique**) ->Code!=#) :{

Si Tete(**CopieSuiteUL**) ->Code== Tete(**PileAnalyseurSyntaxique**) ->Code { **//Action1**

Supprimer l'entête de **CopieSuiteUL**

Dépiler l'élément tète de **PileAnalyseurSyntaxique }**

Sinon

Si Tete(**PileAnalyseurSyntaxique**) est un non-Terminal {

Si TA[Tete(**PileAnalyseurSyntaxique**)->code][Tete(**CopieSuiteUL**) ->Code] !=**Erreur** { **//Action2**

Dépiler l'élément tète de **PileAnalyseurSyntaxique**

Empiler la Règle de production TA[Tete(**PileAnalyseurSyntaxique**)->code][Tete(**CopieSuiteUL**) ->Code] dans **PileAnalyseurSyntaxique** }

Sinon **Erreur Syntaxique** }

Sinon **Erreur Syntaxique**

Afficher (**PileAnalyseurSyntaxique et CopieSuiteUL)**

}

Si (Tete(**SuiteUL**) ->Code ==# && Tete(**PileAnalyseurSyntaxique**) ->Code==#) **Analyse Syntaxique réussite**

Sinon **Erreur Syntaxique**

Exemple de déroulement d'exécution

Code source à analyser :

```
select note
from tab1
where note > 10;
```

PileNVT: S Code_FIN_#

SuiteUL: Code_SELECT Code_IDENT Code_FROM Code_IDENT Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_SELECT Pile =S

Table_DAnalyse[0 , 0] = 0

Depiler S & Empiler RP=0 (Action2)

PileNVT: Code_SELECT A Code_FROM C D Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_SELECT Code_IDENT Code_FROM Code_IDENT Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_SELECT Pile =Code_SELECT

Depiler Code_SELECT (Action1)

PileNVT: A Code_FROM C D Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_IDENT Code_FROM Code_IDENT Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_IDENT Pile =A

Table_DAnalyse[1 , 5] = 2

Depiler A & Empiler RP=2 (Action2)

PileNVT: B Code_FROM C D Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_IDENT Code_FROM Code_IDENT Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_IDENT Pile =B

Table_DAnalyse[2 , 5] = 4

Depiler B & Empiler RP=4 (Action2)

PileNVT: C Code_FROM C D Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_IDENT Code_FROM Code_IDENT Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_IDENT Pile =C

Table_DAnalyse[3 , 5] = 5

Depiler C & Empiler RP=5 (Action2)

PileNVT: Code_IDENT C1 Code_FROM C D Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_IDENT Code_FROM Code_IDENT Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_IDENT Pile =Code_IDENT

Depiler Code_IDENT (Action1)

PileNVT: C1 Code_FROM C D Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_FROM Code_IDENT Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_FROM Pile =C1

Table_DAnalyse[4 , 1] = 7

Depiler C1 & Empiler RP=7 (Action2)

PileNVT: Code_FROM C D Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_FROM Code_IDENT Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_FROM Pile =Code_FROM

Depiler Code_FROM (Action1)

PileNVT: C D Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_IDENT Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_IDENT Pile =C

Table_DAnalyse[3 , 5] = 5

Depiler C & Empiler RP=5 (Action2)

PileNVT: Code_IDENT C1 D Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_IDENT Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE

Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_IDENT Pile =Code_IDENT

Depiler Code_IDENT (Action1)

PileNVT: C1 D Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG

Code_FIN_#

SuiteUL Code_WHERE Pile =C1

Table_DAnalyse[4 , 2] = 7

Depiler C1 & Empiler RP=7 (Action2)

PileNVT: D Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG

Code_FIN_#

SuiteUL Code_WHERE Pile =D

Table_DAnalyse[5 , 2] = 8

Depiler D & Empiler RP=8 (Action2)

PileNVT: Code_WHERE E Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_WHERE Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG

Code_FIN_#

SuiteUL Code_WHERE Pile =Code_WHERE

Depiler Code_WHERE (Action1)

PileNVT: E Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_IDENT Pile =E

Table_DAnalyse[6 , 5] = 10

Depiler E & Empiler RP=10 (Action2)

PileNVT: F E1 Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_IDENT Pile =F

Table_DAnalyse[8 , 5] = 13

Depiler F & Empiler RP=13 (Action2)

PileNVT: G Code_OP_REL G E1 Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_IDENT Pile =G

Table_DAnalyse[9 , 5] = 15

Depiler G & Empiler RP=15 (Action2)

PileNVT: Code_IDENT G1 Code_OP_REL G E1 Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_IDENT Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_IDENT Pile =Code_IDENT

Depiler Code_IDENT (Action1)

PileNVT: G1 Code_OP_REL G E1 Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_OP_REL Pile =G1

Table_DAnalyse[10 , 10] = 18

Depiler G1 & Empiler RP=18 (Action2)

PileNVT: Code_OP_REL G E1 Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_OP_REL Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_OP_REL Pile =Code_OP_REL

Depiler Code_OP_REL (Action1)

PileNVT: G E1 Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_NOMBRE Pile =G

Table_DAnalyse[9 , 6] = 16

Depiler G & Empiler RP=16 (Action2)

PileNVT: Code_NOMBRE G1 E1 Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_NOMBRE Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_NOMBRE Pile =Code_NOMBRE

Depiler Code_NOMBRE (Action1)

PileNVT: G1 E1 Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_SEP_PTVIRG Pile =G1

Table_DAnalyse[10 , 9] = 18

Depiler G1 & Empiler RP=18 (Action2)

PileNVT: E1 Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_SEP_PTVIRG Pile =E1

Table_DAnalyse[7 , 9] = 12

Depiler E1 & Empiler RP=12 (Action2)

PileNVT: Code_SEP_PTVIRG Code_FIN_#

SuiteUL: Code_SEP_PTVIRG Code_FIN_#

SuiteUL Code_SEP_PTVIRG Pile =Code_SEP_PTVIRG

Depiler Code_SEP_PTVIRG (Action1)

PileNVT: Code_FIN_#

SuiteUL: Code_FIN_#

SuiteUL Code_FIN_# Pile =Code_FIN_#

FIN D'ANALYSE SYNTAXIQUE AVEC SUCCES