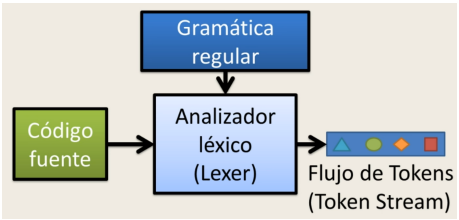
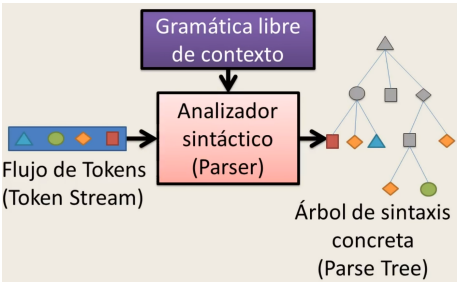
Talf laboratorio 1-b

Integrantes: Norton Irarrázabal
Profesor: Eric Castillo
Fecha: 29 abril 2019

0.0.1. Introducción

- Se generará una gramática utilizando ANTLR la gramática desarrollada se encuentra en el contexto del lenguaje de consulta SQL. Principalmente resuelve tres tipos de sentencias SELECT, UPDATE, DELETE.
- Para lograrlo se implementaron clausulas como WHERE ORDER BY y palabras claves FROM, ALL, DISTINCT, ASC, DESC.
- El objetivo del laboratorio es realizar solo el analizador Lexer que genera un token stream. Sin embargo, para poder realizar lo antes mencionado es necesario codificar un parser.
- Se ponen en práctica los contenidos vistos en clases de teorÃa respecto a las gramÃ¡ticas regulares, que se componen por terminales y no terminales.
- Se utilizaron 21 terminales y 12 no terminales.
- Las terminales las identificamos como letras minúsculas y las no terminales como mayÃºsculas.
- Se hizo uso de * para indicar si una terminal o no terminal se repite 0 o n veces.
- Se uso una gramática de contexto libre en que cada regla de producción es de la forma V ->w.
    - Donde V es un símbolo no terminal y w es una cadena de terminales y/o no terminales.





0.0.2. Plan general

Descripción del problema

- Generar una gramática utilizando ANTLR.

Ámbito

- Asignatura de Talf especÃficamente laboratorio tarea 1-B.

Alcance

- La gramática generada abordara 3 sentencias SQL.

Restricciones

- Lenguaje de programación java.
- Software de escritorio.
- Monousuario.
- No necesita conexión a internet.
- Utilizara sistema de control de versiones GitHub.

Meta

- Generar una gramática.

Objetivos

- Instalar, configurar y usar ANTLR.
- Generar una gramática.
- Corroborar que la gramática generada funciona.
- Generar documento similar a manual de usuario.

0.0.3. Requerimientos

Requerimientos funcionales

- La gramática debe tener un nombre asignado.
- Tendrá gramática libre de contexto.
- Tendrá gramática regular.
- Debe funcionar para la gramática SQL específicamente SELECT, UPDATE, DELETE.
- La gramática debe funcionar correctamente.
- La gramática será probada con 9 consultas.

```
1) SELECT * FROM Coches ORDER BY marca, modelo ASC;
2) SELECT ALL matricula, marca, modelo, color, numero_kilometros, num_plaza
   FROM Coches
   ORDER BY marca, modelo;
5) SELECT ALL matricula, marca, modelo, color, numero_kilometros, num_plaza
   FROM Coches
   WHERE matricula = 'MF234ZD' OR matricula = 'FK938ZL' ;
6) SELECT ALL matricula, marca, modelo, color, numero_kilometros, num_plazas
   FROM coches
   WHERE NOT matricula = 'MF-234-ZD';
7) SELECT DISTINCT marca, modelo FROM coches;
8) DELETE FROM tabla WHERE columna1 = 'valor1';
9) UPDATE My_table SET field1 = 'a' WHERE field2 = 'N';
```

- Se debe visualizar vista de la gramática.

Requerimientos no funcionales

- Sera monousuario.
- Debe ser capaz de ejecutarse en no más de 4 seg.
- Debe proporcionar mensajes de error en caso de que ocurran.
- El software debe contar con manual de usuario.
- No se tomará en cuenta la seguridad.
- Se utilizará nomenclatura snake_case.
- No debe ser complejo de usar.
- Utilizará lenguaje de programación java.
- Hará uso de ANTLR.
- Sera de escritorio.

0.0.4. ANTLR

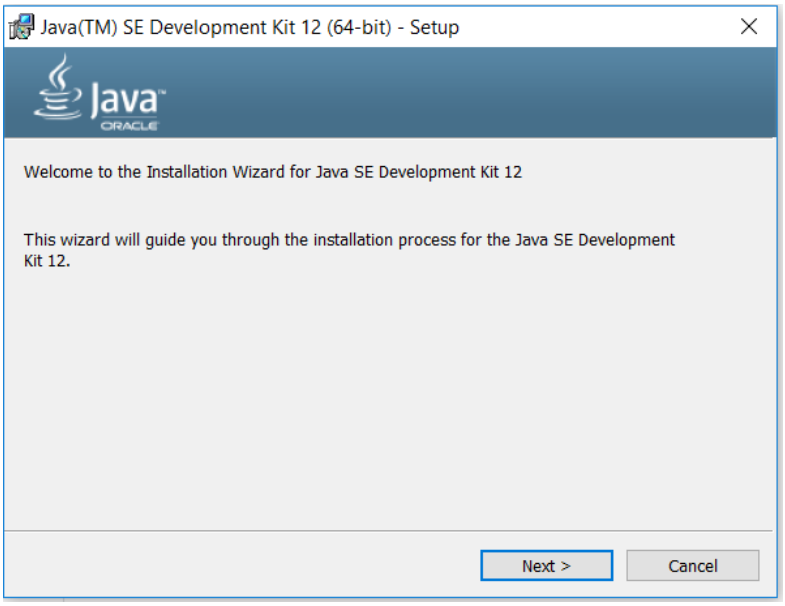
Herramienta que opera sobre lenguajes, proporcionando un marco para construir reconocedores (parsers), intérpretes, compiladores y traductores de lenguajes a partir de las descripciones gramaticales de los mismos.

Partiendo de la descripción formal de la gramática de un lenguaje, ANTLR genera un programa que determina si una sentencia o palabra pertenece a dicho lenguaje (reconocedor). Si a dicha gramática, se le añaden acciones escritas en un lenguaje de programación, el reconocedor se transforma en un traductor o intérprete.

Disponible para: Linux, Windows y Mac OS X.
Inicio: Año 1988.
Versión actual: 4.7.2.
Se hace uso de ANTLR en: Twitter, Pig Hive, Hadoop, Lex Machina, Oracle, NetBeans, Lenguaje HQL.

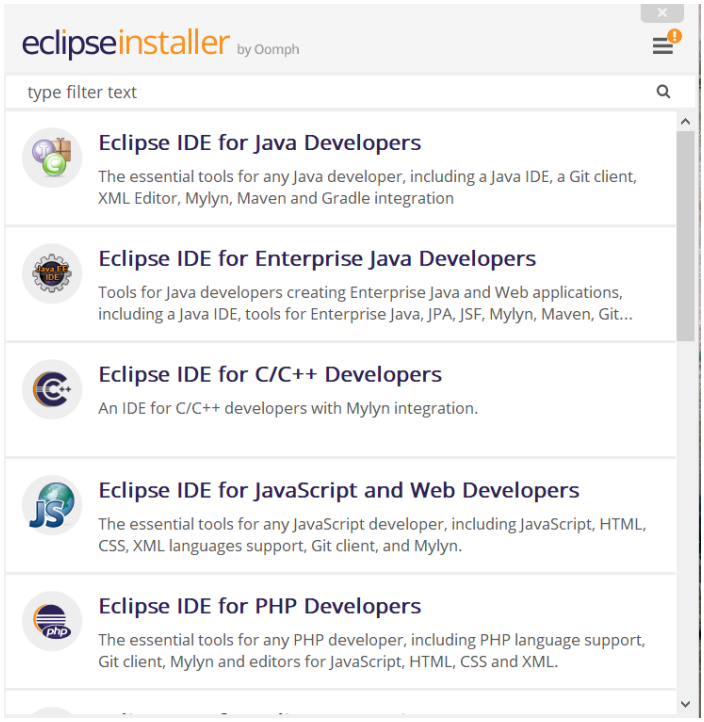
0.0.5. Manual de usuario

- **Instalaciones previas**
- JDK 12

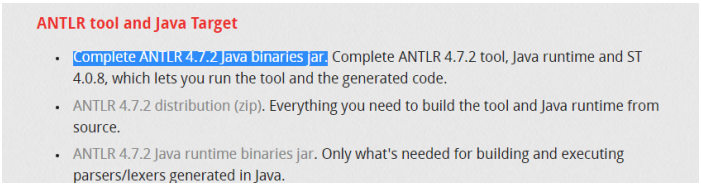


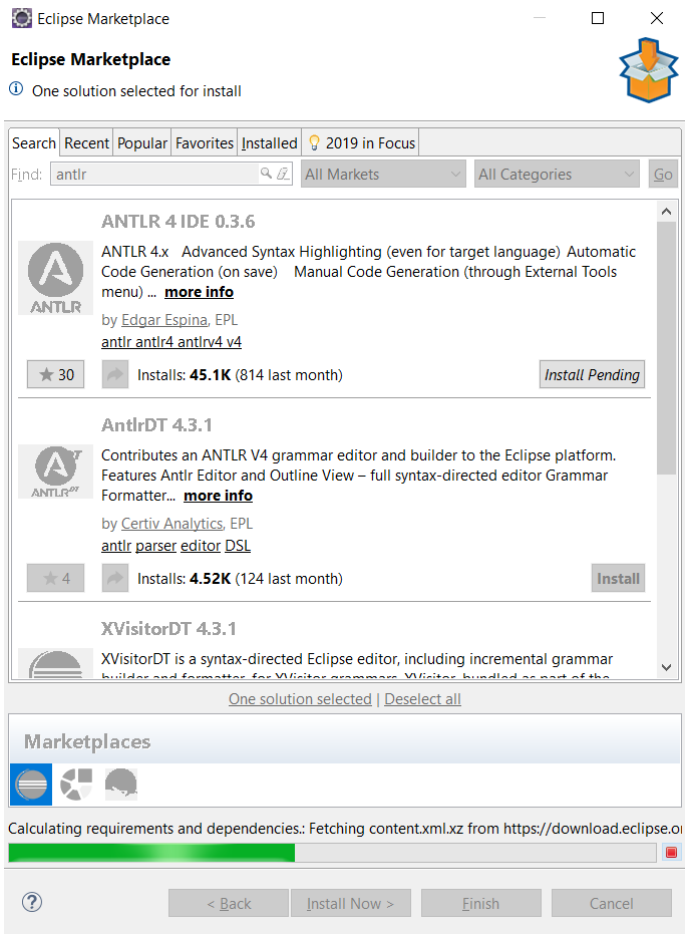
- JRE 1.8

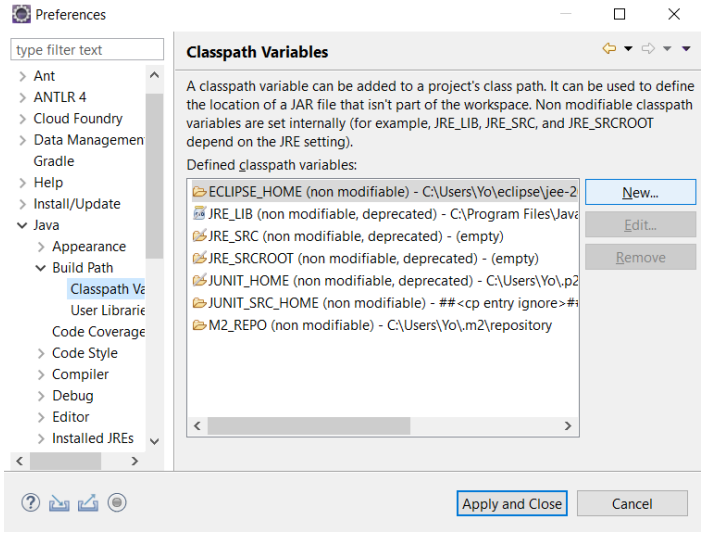
- Eclipse jee-2019-0

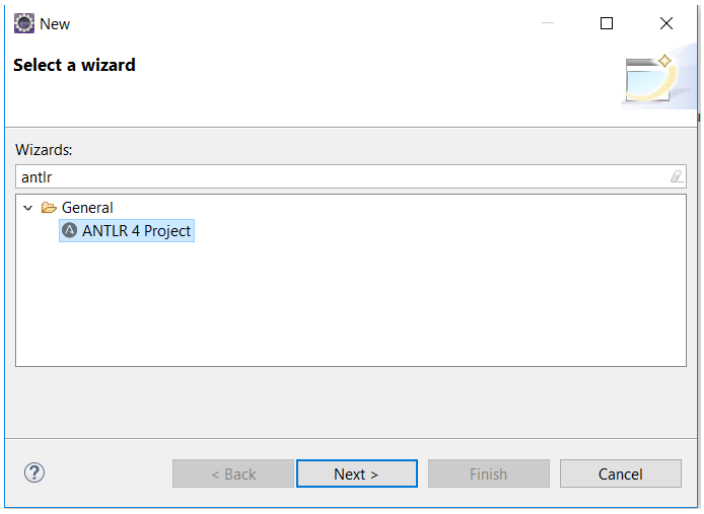


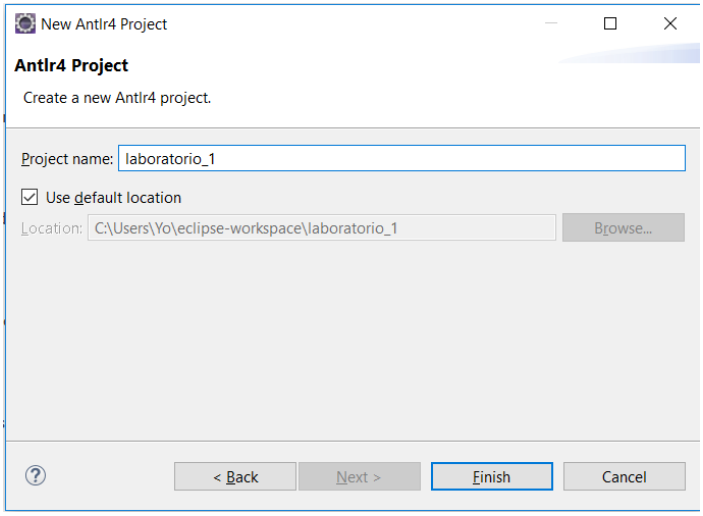
- Instalación de ANTLR






Preferences
type filter text
Ant
ANTLR 4
Cloud Foundry
Data Managemen
Gradle
Help
Install/Update
Java
Appearance
Build Path
Classpath Va
User Librarie
Code Coverage
Code Style
Compiler
Debug
Editor
Installed JREs
Classpath Variables
A classpath variable can be added to a project's class path. It can be used to define the location of a JAR file that isn't part of the workspace. Non modifiable classpath variables are set internally (for example, JRE_LIB, JRE_SRC, and JRE_SRCROOT depend on the JRE setting).
Defined classpath variables:
ECLIPSE_HOME (non modifiable) - C:\Users\Yo\eclipse\jee-2
JRE_LIB (non modifiable, deprecated) - C:\Program Files\Java
JRE_SRC (non modifiable, deprecated) - (empty)
JRE_SRCROOT (non modifiable, deprecated) - (empty)
JUNIT_HOME (non modifiable, deprecated) - C:\Users\Yo\.p2
JUNIT_SRC_HOME (non modifiable) - ##<cp entry ignore>#
M2_REPO (non modifiable) - C:\Users\Yo\.m2\repository
New...
Edit...
Remove
Apply and Close
Cancel


New
Select a wizard
Wizards:
antlr
General
ANTLR 4 Project
< Back
Next >
Finish
Cancel


New Antlr4 Project
Antlr4 Project
Create a new Antlr4 project.
Project name: laboratorio_1
Use default location
Location: C:\Users\Yo\eclipse-workspace\laboratorio_1
Browse...
< Back
Next >
Finish
Cancel

0.0.6. Código

- **No terminales**

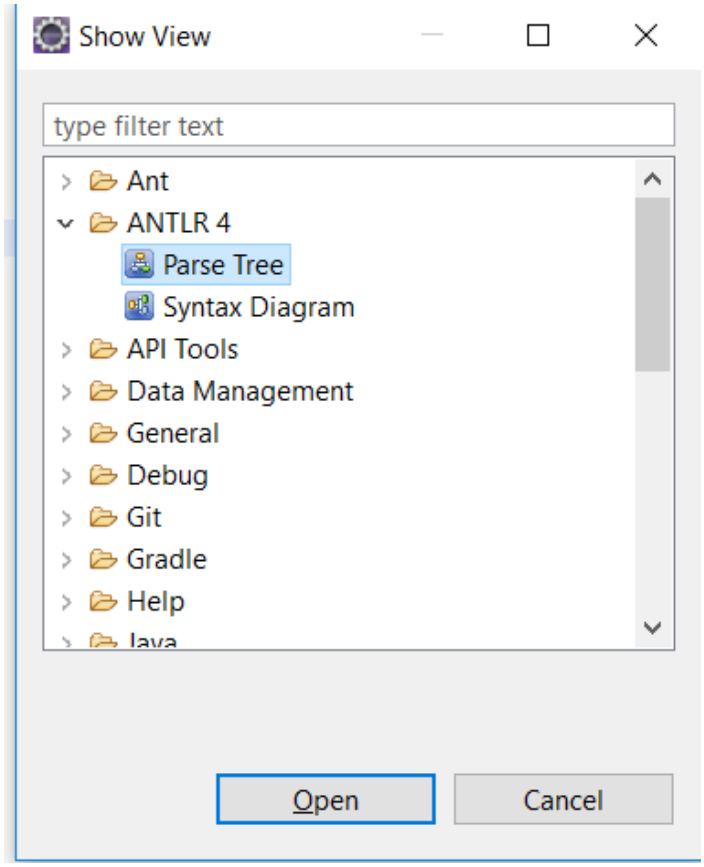
```
consulta: (select | update | delete);

select: SELECT (ASTERISCO| ( (ALL | DISTINCT) (nombre_de_campo_coma* ID))) from
update: UPDATE ID SET condicion_coma* condicion where;
delete: DELETE from;

from: FROM ID (PUNTO_COMA | order_by | where);

order_by: ORDER_BY nombre_de_campo_coma* condicion_asc_desc;
where: WHERE NOT* condicion_AND_OR* condicion PUNTO_COMA;

condicion_asc_desc: ID (ASC|DESC) PUNTO_COMA;
nombre_de_campo_coma: ID COMA;

condicion_AND_OR: condicion (AND|OR);
condicion_coma: condicion COMA;
condicion: ID IGUAL COMILLA ID COMILLA;
```

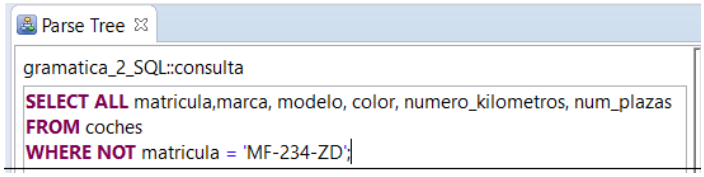
- **Terminales**

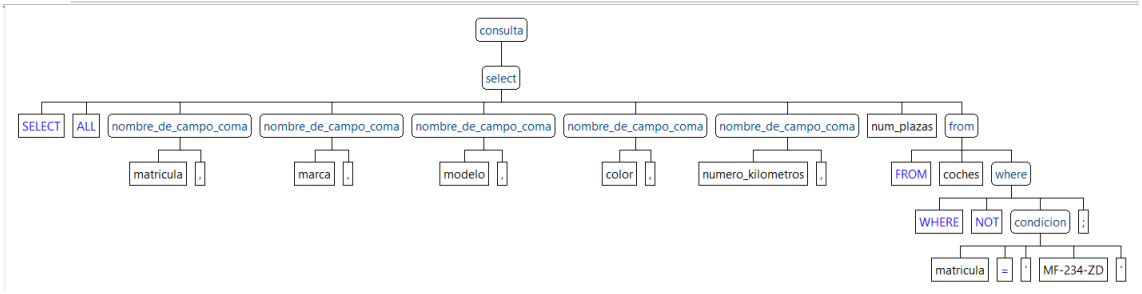
```
SELECT: 'SELECT';
DELETE: 'DELETE';
UPDATE: 'UPDATE';

ALL: 'ALL';
DISTINCT: 'DISTINCT';

FROM: 'FROM';
WHERE: 'WHERE';
NOT: 'NOT';

OR: 'OR';
AND: 'AND';

SET: 'SET';
ASC: 'ASC';
DESC: 'DESC';
ID: [a-zA-Z_][a-zA-Z0-9_\-]*;

ORDER_BY: 'ORDER BY';

PUNTO_COMA: ';';

ASTERISCO: '*';

COMA: ',';
```

```
IGUAL: '=';
COMILLA  : '\'';

WS:  [ \t\n\r]+ -> skip;
```

0.0.7. Siga los siguientes pasos para ver gramática






consulta
select
SELECT
ALL
nombre_de_campo_coma
matricula
,
nombre_de_campo_coma
marca
,
nombre_de_campo_coma
modelo
,
nombre_de_campo_coma
color
,
nombre_de_campo_coma
numero_kilometros
,
num_plazas
from
FROM
coches
where
WHERE
NOT
condicion
;
matricula
=
'
MF-234-ZD
'