Universidad San Carlos de Guatemala
Centro Universitario de Occidente
División de Ciencias de la Ingeniería
Ingeniería en Ciencia y Sistemas
Laboratorio de Lenguajes Formales y de Programación
Proyecto 1

Objetivo

General

- Desarrollar un programa que lea una consulta SQL, que genera tokens y luego verifique que la sintaxis sea válida.

Específicos

- Implementar un Analizador léxico usando la herramienta JFlex.
- Crear una gramática independiente del contexto, que reconozca la estructura sintáctica de SQL.
- Comprender la relación entre el analizador léxico y sintáctico.

Descripción

La empresa LogiCode ha decidido asignarle un nuevo proyecto, dado su desempeño con los analizadores léxicos. El sistema consiste en la creación de un analizador léxico usando la herramient JFlex, el lenguaje que debe de reconocer es el de SQL, pero para esta ocasión también se le solicita que implemente una analizador sintáctico para que que reconozca la estructura del lenguaje y una vez que la entrada es analizada, debe extraer la información analizada y generar diagramas de las tablas.

Analizador Léxico

El lenguaje aceptara estructuras DDL y DML, además deberá contar con un editor de código en el cual los token deben de pintarse según sea el color asignado al tipo de token . A Continuación se presentan los token que debe de reconocer.

Lenguaje SQL

Nombre	Token	Color	Observación
CREATE	• CREATE • DATABASE • TABLE	Naranja	

	• KEY • NULL • PRIMARY • UNIQUE • FOREIGN • REFERENCES • ALTER • ADD • COLUMN • TYPE • DROP • CONSTRAINT • IF • EXIST • CASCADE • **ON** • DELETE • SET • UPDATE • INSERT • INTO • VALUES • SELECT • FROM • WHERE • AS • GROUP • ORDER • BY • ASC • DESC • LIMIT • JOIN		
TIPO DE DATO	• INTEGER, BIGINT • VARCHAR • DECIMAL • DATE • TEXT • BOOLEAN • SERIAL	MORADO	
Entero	Ejemplo: 100, 50, 2, 3, 25	Azul	
Decimal	Ejemplo: 65.50, 52.02, 6000.558	Azul	

Fecha	Ejemplo: ‘2024-10-15’ ‘1999-02-05’ ‘2555-12-22’	Amarillo	La fecha solo puede aceptar el formato: 'YYYY-MM-DD', y debe esta dentro de comillas simples
Cadena	Ejemplo: ‘abcd’, ‘saludo 321’ ‘camino’’	Verde	Una cadena debe estar dentro de comillas simples y puede aceptar cualquier carácter.
identificador	Ejemplos: empleados, departamento_publico categoria_por_rol	Fucsia	Se usa la nomenclatura de **snake_case**, es decir que solo se aceptan letras minúsculas , guión bajo y números enteros.
Booleano	TRUE, FALSE	Azul	
Funciones de Agregación	• SUM • AVG • COUNT • MAX • MIN	Azul	
Signos	• (•) • , • ; • . • =	Negro	
Aritméticos	+, -, *, /	Negro	
Relacionales	<, >, <=, >=	Negro	
Lógicos	• AND • OR • NOT	Naranja	
Comentario de línea	- - Este es un comentario - - Una Línea	Gris	Los comentarios empiezan por dos guiones medio, ambos separador por un espacio, luego puede venir cualquier carácter. Y son de una línea los comentarios.

Analizador Sintactico

Ya que un analizador sintáctico analiza la estructura, a continuación se presentan las estructuras del lenguaje SQL.

Para los DDL

Creación de Base de Datos

Estructura

```
CREATE DATABASE  <identificador> ;
```

Ejemplo

```
CREATE DATABASE  store;
CREATE DATABASE  shop_and_store;
```

Creación de Tablas

La Estructura_de_llaves es opcional puede o no venir, y puede tener multiples Estructura_de_declaracion separadas por coma.

Estructura

```
CREATE TABLE <identificador> (
[Estructura_de_declaracion] ,
[Estructura_de_llaves]
);
```

Estructura de declaración

Pueden tener múltiples estructuras de declaración separadas por comas.

Estructura

```
<identificador> [Tipo_de_dato] PRIMARY KEY

<identificador> [Tipo_de_dato] NOT NULL

<identificador> [Tipo_de_dato] UNIQUE
```

```
<identificador> [Tipo_de_dato]
```

Tipo de Dato

Puede ser alguno de los siguientes

```
SERIAL | INTEGER | BIGINT | VARCHAR(<entero>) | DECIMAL(<entero>,
<entero>) | DATE | TEXT | BOOLEAN
```

Estructura De Llaves

Estructura

```
CONSTRAINT <identificador>
      FOREIGN KEY (<identificador>)
      REFERENCE <identificador>(<identificador>)
```

Ejemplo

```
CREATE TABLE empleados (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    puesto VARCHAR(50),
    salario DECIMAL(10, 2),
    fecha_contratacion DATE,
    departamento_id INTEGER,
    email VARCHAR(100) UNIQUE,
    CONSTRAINT fk_departamento
        FOREIGN KEY (departamento_id)
        REFERENCES departamentos(id)
);
```

Modificadores

Estructura

```
ALTER TABLE <identificador> ADD COLUMN <identificador> [Tipo_de_dato];

ALTER TABLE <identificador> ALTER COLUMN <identificador> TYPE
[Tipo_de_dato];

ALTER TABLE <identificador> DROP COLUMN <identificador>;

ALTER TABLE <identificador> ADD CONSTRAINT <identificador>
[Tipo_de_dato];

ALTER TABLE <identificador> ADD CONSTRAINT <identificador> UNIQUE
(<identificador>);

ALTER TABLE <identificador> ADD CONSTRAINT <identificador> FOREIGN KEY
(<identificador>) REFERENCES <identificador>(<identificador>);

DROP TABLE IF EXISTS <identificador> CASCADE;
```

Ejemplo

```
ALTER TABLE empleados ADD COLUMN fecha_nacimiento DATE;

ALTER TABLE empleados ALTER COLUMN salario TYPE NUMERIC(12, 2);

ALTER TABLE empleados DROP COLUMN puesto;

ALTER TABLE empleados ADD CONSTRAINT uc_email UNIQUE (email);

DROP TABLE IF EXISTS empleados CASCADE;

ALTER TABLE empleados
ADD CONSTRAINT fk_departamento
FOREIGN KEY (departamento_id)
REFERENCES departamentos(id)
ON DELETE SET NULL
ON UPDATE CASCADE;
```

Para los DML

Inserción

Estructura

```
INSERT INTO <identificador> (<identificador>,...) VALUES ([DATO],...);

INSERT INTO <identificador> (<identificador>,...) VALUES ([DATO],...),
...([DATO],...);
```

Dato
Para el datos puede ser alguno de los datos primitivos o alguna relación de expresión aritmética, racional o lógico, y puede aceptar la agrupación de instrucciones entre paréntesis.

```
<entero> | <decimal> | <fecha> | <cadena> | TRUE | FALSE

[DATO] + [DATO] * [DATO] / [DATO]

[DATO] < [DATO] OR ( [DATO] + [DATO] ) < [DATO]
```

Ejemplo

```
INSERT INTO empleados (nombre, puesto, salario, fecha_contratacion,
departamento_id, email)
VALUES ('Juan Pérez', 'Ingeniero', 3500.50, '2024-01-15' < '2024-01-11',
2, 'juan.perez@empresa.com');

INSERT INTO empleados (nombre, puesto, salario, fecha_contratacion,
departamento_id, email)
VALUES
('Ana Gómez', 'Gerente', 6000.00 + 1000, '2023-11-20', 1,
'ana.gomez@empresa.com'),
('Carlos Rodríguez', 'Analista', 2800.75 < 8000, '2023-12-01', 3,
'carlos.rodriguez@empresa.com');

INSERT INTO empleados (nombre, email)
VALUES ('Luis Martinez', 'luis.martinez@empresa.com');
```

Lectura

Para la selección de columnas, se acepta el valor *, <identificador>, [FUNCION_AGREGACION], también se admiten llamadas entre identificador <identificador>.<identificador>, y el uso de AS para nombrar alguna columna o valor.

[SENTENCIAS] es opcional

Estructura

```
SELECT * FROM <identificador> [SENTENCIAS];

SELECT [SELECCION_DE_COLUMNA] FROM <identificador> [SENTENCIAS];
```

Selección de Columna

```
<identificador> | [FUNCION_AGREGACION] | <identificador>.<identificador>
```

Función Agregación

```
SUM(<identificador>)
AVG(<identificador>)
COUNT(<identificador>)
MIN(<identificador>)
MAX(<identificador>)
```

Sentencia
Cada una de las siguientes sentencias son opcionales pueden o no venir.

```
[JOIN] [WHERE] [GROUP] [ORDER] [LIMIT]

JOIN <identificador> <identificador> ON <identificador>.<identificador>
= <identificador>.<identificador>

JOIN <identificador> <identificador> ON <identificador> =
<identificador>.<identificador>

WHERE <identificador> = [DATO] | <identificador> |
<identificador>.<identificador>
WHERE [DATO] [OP_RACIONAL] AND <identificador> =
<identificador>.<identificador>

GROUP BY [<identificador> | <identificador>.<identificador>]
```

```
ORDER BY [<identificador> | <identificador>.<identificador>] [DESC |
ASC]

LIMIT <entero>
```

Ejemplo

```
SELECT * FROM empleados;

SELECT nombre, puesto, salario
FROM empleados;

SELECT nombre, salario
FROM empleados
WHERE (salario > 3000) AND nombre='abc';

SELECT departamento_id, AVG(salario) AS salario_promedio
FROM empleados
GROUP BY departamento_id;

SELECT nombre, puesto, salario
FROM empleados
ORDER BY salario DESC;

SELECT nombre, salario
FROM empleados
WHERE salario > 3000
LIMIT 5;

SELECT e.nombre, e.puesto, d.nombre AS departamento
FROM empleados e
JOIN departamentos d ON e.departamento_id = d.id;
```

Actualización

[WHERE] estructura opcional.

Estructura

```
UPDATE <identificador> SET <identificador> = [DATO] [WHERE];

UPDATE <identificador> SET <identificador> = [DATO], <identificador> =
[DATO],...  [WHERE];
```

Ejemplo

```
UPDATE empleados
SET salario = 4000.00
WHERE id = 1;

UPDATE empleados
SET puesto = 'Supervisor', salario = salario * 1.10
WHERE departamento_id = 2;

UPDATE empleados
SET salario = salario + 500
WHERE salario < 3000 AND departamento_id = 1;
```

Eliminación

[WHERE] estructura opcional.

Estructura

```
DELETE FROM <identificador> [WHERE];
```

Ejemplo

```
DELETE FROM empleados
WHERE id = 3;

DELETE FROM empleados
WHERE salario < 2500;

DELETE FROM empleados;
```

Generacion de Graficos

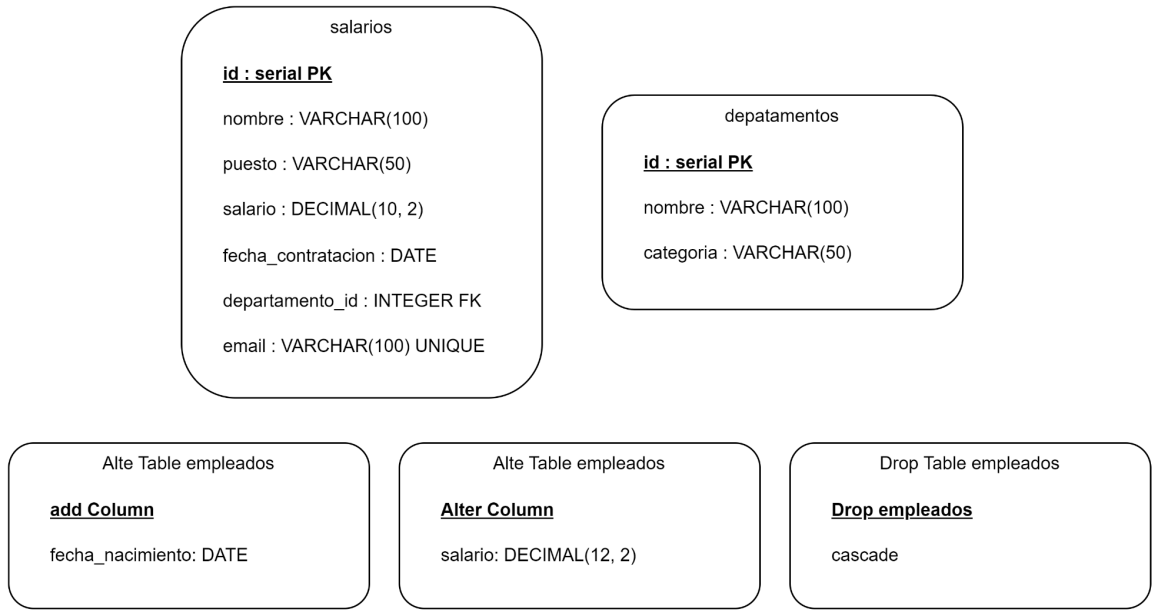
Para realizar los gráficos se recomienda usar graphviz.

Para los DDL

Para la creación de tablas, se debe hacer un diagrama de tablas, donde se detalle la configuración de las tablas encontradas en el código fuente, no es necesario hacer la relación que pueda existir entre tablas, cada tabla debe tener la información de sus columnas y llaves únicamente.

Para los modificadores solo debe realizar una diagrama sencillo que especifique el tipo de modificador y a que columnas y tabla está modificando.

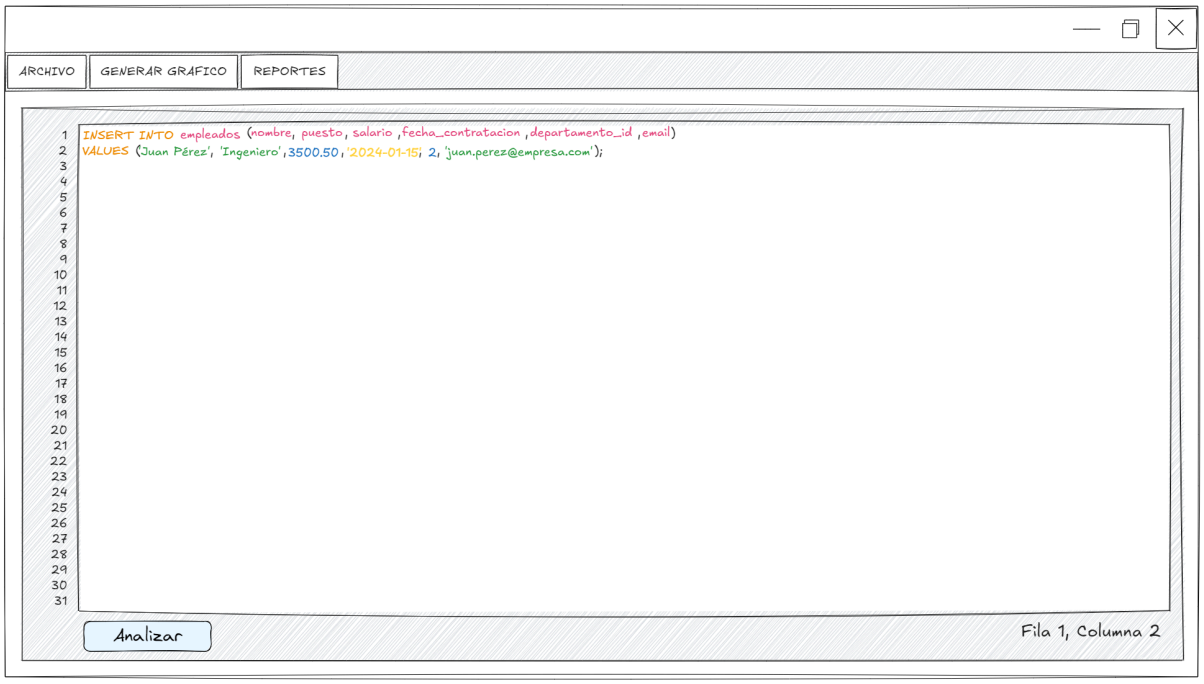
Ejemplo sugerido:



Editor

La interfaz debe incluir un editor de texto que muestre la línea y columna donde se encuentre el cursor, además de eso debe colorear las palabras según el color especificado para el grupo de tokens que se encuentran en la tabla del analizador léxico. También debe permitir cargar, crear, guardar, y guardar como, para los archivos, si inicialmente no se encuentra abierto ningún archivo debe permitir escribir en el editor de texto.

Los siguiente mockup presentan un ejemplo de cómo se puede realizar la interfaz, pero el diseño queda a criterio de cada uno.



Reporte de Errores

Léxico

Se deberá de listar los tokens identificados, el reporte debe incluir Token, línea, y columna, descripción del error, ejemplo:

Token	Línea	Columna	Descripción
<>	1	3	Token no reconocido
?	4	5	Carácter no reconocido
TBL	7	8	Token no reconocido

Sintáctico

Token	Tipo Token	Línea	Columna	Descripción
<	Racional	1	3	Se esperaba un token <entero>
nombre	Identificador	4	5	Secuencia de token invalida
TBL	Identificador	7	8	Secuencia de token invalida

Otros Reportes

- Tablas encontradas
- Tablas modificadas y el tipo de modificación.
- Número de operación por sección: create, delete, update, select, alter

Todos los reportes deben contener línea y columna donde empezó la instrucción encontrada, excepto el de número de operaciones.

Importante

- JFlex
- Usar lenguaje de programación JAVA
- Es válido utilizar algún IDE o cualquier editor de texto
- Las copias obtendrán nota de cero y se notificarán a coordinación
- **No es válido usar herramientas para el análisis sintáctico**
- No es válido copy/paste desde internet, si va a usar algo de internet debe entender el código y poder explicarlo.

Entrega

La fecha de entrega es el día 30 de octubre del 2024 antes de las 23:59 horas por medio de la plataforma Classroom. Los entregables son:

- Código Fuente
- Código compilado (Ejecutable), en la rama principal deberá tener una carpeta con el ejecutable
- **Enlace de repositorio y debe ser público, si no entrega el enlace de repositorio o es privado se tomará como no entregado.**
- Manual Técnico.
- Manual de Usuario.

Calificación

La calificación se realizará el día 31 de octubre del 2024. El estudiantes debe llevar impresa la hoja de calificación que se publicará en los siguientes días.