Universidad **T**ecnológica **N**acional
Facultad **R**egional **R**esistencia
Departamento de **I**ngeniería en **S**istemas de **I**nformación

Sintaxis y Semántica del Lenguaje

Trabajo Práctico Integrador: Diseño e implementación de Lexer y Parser

Ciclo lectivo: 2018

Equipo docente:

Director de Cátedra

Nombre y Apellido: Ing. Gabriela P. TOMASELLI

Categoría docente: Profesora Asociada

e-mail: gabriela.tomaselli@gmail.com

Docentes

Ing. Gabriela P. TOMASELLI	e-mail: gabriela.tomaselli@gmail.com
Ing. Rodrigo VIGIL	e-mail: rodrigovigil@gmail.com
Ing. Nicolas G. Tortosa	e-mail: nicotortosa@gmail.com
Ing. Juliana Torre	e-mail: julitorre025@gmail.com

Universidad **T**ecnológica **N**acional
Facultad **R**egional **R**esistencia
Departamento de **I**ngeniería en **S**istemas de **I**nformación

Práctico 1 (I)	*INTRODUCCION*

Se desea implementar un analizador léxico (lexer) y sintáctico (parser) de un pequeño subconjunto del lenguaje de consulta de Bases de Datos relacionales SQL. El mismo debe poder recibir una serie de definiciones de tablas y consultas y determinar si están correctamente escritas.

Para ello deberán construir los analizadores lexico y sintáctico. El analizador lexicográfico es un módulo que recibe una secuencia de caracteres que componen la consulta SQL y lo convierte lógicamente en una secuencia de tokens. El analizador sintáctico recibe la secuencia de tokens que le entrega el analizador lexicográfico y verifica que la secuencia pueda ser generada por la gramática del lenguaje.
Hay dos grupos de herramientas que se pueden usar para generar los analizadores:
1) Se utilizan expresiones regulares y autómatas finitos para el análisis lexicográfico y la técnica LALR para el análisis sintáctico. Ejemplos de esto son lex y yacc, que generan código C o C++, o JLex y CUP, que generan código Java. flex y bison son implementaciones libres y gratuitas de lex y yacc y se pueden conseguir en http://www.gnu.org/.
2) El otro utiliza la técnica LL(k) tanto para el análisis léxico como para el sintáctico, generando parsers descendentes recursivos. Ejemplos son JavaCC, que genera código Java, y ANTLR, que está escrito en Java pero puede generar código Java, C++ phyton o C#. ANTLR se puede conseguir en http://www.antlr.org/.
https://en.wikipedia.org/wiki/Comparison_of_parser_generators

Introduccion a SQL

Los sistemas de gestión de base de datos organizan y estructuran los datos de tal manera que puedan ser recuperados y manipulados por usuarios y programas de aplicación. Las estructuras de datos y las técnicas de acceso proporcionadas por un DBMS se denominan su modelo de datos. SQL es un lenguaje de base de datos para base de datos relacionales y utiliza el modelo de datos relacional.

Una base de datos relacional es una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores y en donde todas las operaciones de la base de datos actúan sobre estas tablas. Dicho de otra manera, una base de datos es una colección de tablas, que son análogas a los archivos. Cada tabla contiene filas y columnas, que son análogas a registros y campos.

SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos informática. El nombre SQL es una abreviatura de Structured Query Language (Lenguaje de Consultas Estructuradas). Específicamente SQL permite interactuar con un tipo específico de base de datos llamada base de datos relacional. El programa Informático que controla la base de datos es denomina Sistema de Gestión de Base de Datos (Database Management System) o DBMS.

Cuando se necesita recuperar datos de la base de datos, se utiliza el lenguaje SQL para efectuar la petición. El DBMS procesa la petición SQL, recupera los datos solicitados y los devuelve. Este proceso de solicitar datos de la base de datos y de recibir los resultados se denomina consulta (query), de aquí el nombre Structured Query Language.

A pesar de ello este nombre resulta inapropiado puesto que SQL es mucho más que una herramienta de consulta, aunque este fue su propósito original y una de sus funciones más importantes. SQL se utiliza para controlar todas las funciones que un DBMS proporciona a sus

usuarios tales como definición, recuperación, manipulación, compartición e integridad de datos así como control de acceso

Las últimas definiciones para SQL son los estándares publicados por ANSI e ISO, que incluyen ISO / IEC 9075-2: 2003, que define SQL.

OPERACIÓN DE LAS SENTENCIAS DE SQL

A continuación se indican las principales operaciones que se implementan a través de sentencias SQL.

Para crear un conjunto de tablas se necesita un nombre y el tipo de cada columna:
La sentencia CREATE TABLE permite definir una nueva tabla y la prepara para aceptar datos. Su sintaxis es la siguiente:

CREATE TABLE *nombre de tabla* (definición de colurmas)

Las columnas de la tabla recién creada se definen en el cuerpo de la sentencia CREATE TABLE. Las definiciones de columnas aparecen en una lista separada por comas y encerrada entre paréntesis. El orden de las definiciones de las columnas determina el orden de izquierda a derecha de las columnas en la tabla. Cada definición especifica el nombre de la columna y el tipo de datos que la columna almacena. Por ejemplo:

```
CREATE TABLE Foods (
        name CHAR(8) NOT NULL,
        type CHAR(5),
        flavor    CHAR(6),
        PRIMARY KEY ( name )
)

CREATE TABLE Courses (
        course      CHAR(8) NOT NULL PRIMARY KEY,
        flavor      CHAR(6),
        sequence INTEGER
)
```

Consultas de datos

Las consultas de selección se utilizan para obtener información de las bases de datos, esta información es devuelta en forma de conjunto de registros.

La sintaxis básica de una consulta de selección es la siguiente:

SELECT Lista de campos FROM Nombre de tabla WHERE Condicion

En donde Lista de Campos es la relación de campos que se desean recuperar y Nombre de Tabla identifica a la tabla, origen de los mismos. El resultado de una consulta SQL es siempre una tabla de datos, semejante a las tablas almacenadas en la base de datos. Generalmente los resultados de la consulta generarán una tabla con varías columnas y varias filas. Las consultas más sencillas solicitan columnas de datos de una única tabla en la base de datos.

```
SELECT name, flavor
FROM  Foods
WHERE Foods.type = "fruit"
```

Entonces una sentencia de este lenguaje viene definida por tres clausulas o partes: la clausula SELECT, la clausula FROM y la clausula WHERE.

Universidad **T**ecnológica **N**acional
Facultad **R**egional **R**esistencia
Departamento de **I**ngeniería en **S**istemas de **I**nformación

Descripción del lenguaje de entrada
El siguiente ejemplo contiene todos los elementos sintácticos de SQL que utilizaremos para este trabajo práctico:

```
CREATE TABLE cliente
( codigo NUMBER, nombre STRING ) ;
CREATE TABLE factura
( codigo NUMBER, codcli NUMBER , importe NUMBER ) ;

SELECT  cliente.nombre, importe
FROM cliente, factura
WHERE  codcli = cliente.codigo
AND (importe > 100 OR nombre = 'Juan') ;

SELECT codigo
FROM cliente
WHERE nombre = 'Juan' ;
```

La entrada consiste en una secuencia de sentencias de creación de tablas, seguida por una secuencia de consultas. No se permiten renombramientos de tablas ni de columnas. El operador lógico AND tiene mayor precedencia que el operador lógico OR. Los únicos tipos de datos son el numérico (NUMBER) y cadena de caracteres (STRING). Los saltos de línea, tabulaciones y espacios no son significativos. Tampoco se distingue entre mayúsculas y minúsculas.

Práctico 1 (II)	*TRABAJO PRACTICO*

El trabajo práctico estará dividido en dos etapas:
• Análisis léxico
• Análisis sintáctico

La entrega debe incluir:
• Un programa que cumpla con lo solicitado. que reciba como entrada (standard o de un archivo) un conjunto de sentencias SQL genéricas e indique si es léxica y sintácticamente correcta o no, en lo posible indicando la ubicación del error en caso de existir.
• El código fuente del programa. Si se usaron herramientas generadoras de código, imprimir la fuente ingresada a la herramienta, no el código generado.
• Una lista de los tipos de token que componen el lenguaje, con la definición asociada a cada uno.
• Una gramática libre de contexto para el lenguaje que utilice como símbolos terminales los tokens definidos en el punto anterior.
• Consultas de ejemplo y resultados.
• Informe conteniendo:
    • Descripción de cómo se implementó la solución.
    • Información y requerimientos de software para ejecutar y recompilar el tp (versiones de compiladores, herramientas, plataforma, etc).
    • Casos de prueba con expresiones sintácticamente correctas e incorrectas, resultados obtenidos y conclusiones.

Análisis Léxico (1ª entrega)
El analizador léxico, también conocido como scanner, lee los caracteres uno a uno desde la entrada y va formando grupos de caracteres con alguna relación entre si (tokens), que constituirán la entrada para la siguiente etapa del compilador, Cada token representa una

secuencia de caracteres que son tratados como una única entidad. Por ejemplo, en SQL un token es la palabra reservada SELECT,

Ya que es el que va leyendo los caracteres del programa, ignorará aquellos elementos innecesarios para la siguiente fase, como los tabuladores, comentarios, espacios en blanco, etc.

Se trata de implementar un analizador léxico, en el que exista una función que devuelva el siguiente token, es decir, que devuelva el tipo de token en forma de constante intera, y su lexema, en forma de cadena. En esta primera fase de la práctica, y con la finalidad de comprobar que funciona correctamente, esta función será llamada por una función main, que solicitará nuevos tokens hasta que se agote el texto del archivo que contiene las consultas en SQL.
El programa deberá imprimir una lista de tokens de la forma (tipo_de_token, lexema) como resultado, deteniéndose en el caso de un error léxico e indicando la línea y columna del texto fuente donde éste se produjo.

Algunos ejemplos de componentes léxicos a reconocer son los siguientes:

TKN_NUM = digito+ (. digito+)?
TKN_ID = letra (letra | digito)*
# siendo digito = 0 | 1 | ... | 9 letra = a | b | ... | z | A | B | ... | Z
#Símbolos especiales
TKN_APAR (
TKN_CPAR)
TKN_PTOCOMA ;
TKN_PTO .
#Palabras reservadas:
CREATE TKN_CREATE
TABLE TKN_TABLE
SELECT TKN_SELECT
FROM TKN_FROM
WHERE TKN_WHERE
GROUP TKN_GROUP
BY TKN_BY
#Funciones sobre campos agregados:
AVG TKN_AVG //calcula el promedio SUM TKN_SUM //calcula la suma MAX TKN_MAX

Operadores: <,>,>=,<=,= ,!=, and, or

Analisis SINTACTICO (2 entrega. Final)
El analizador sintáctico, también llamado parser, recibe como entrada los tokens que le pasa el Analizador Léxico (el analizador sintártico no maneja directamente caracteres) y comprueba si esos tokens van llegando en el orden correcto (orden permitido por el lenguaje). Así pues, sus funciones son:

- Aceptar lo que es válido sintácticamente y rechazar lo que no lo es.

- Hacer explícito el orden jerárquico que tienen los operadores en el lenguaje de que se trate.

- Guiar el proceso de traducción (traducción dirigida por la sintaxis).

Para esta etapa se solicita la construcción de una Gramática que genere el lenguaje a reconocer

Algunas restricciónes para facilitar el trabajo e implementación:

*Una sentencia SQL se compone de al menos de la clausula SELECT y FROM. La clausula WHERE, ORDER BY y GROUP BY son opcionales.
*Si aparecen todas, el orden es: SELECT, FROM, WHERE, ORDER BY y GROUP BY.
* Los nombres de palabras reservadas e identificadores no son case sensitive.
* No se pueden definir dos tablas con el mismo nombre y dentro de una tabla, no puede haber dos columnas con el mismo nombre.
* Las columnas referenciadas en las cláusulas SELECT y WHERE deben cumplir:
– Si se usó la forma tabla.columna, entonces tabla debe estar mencionada en la cláusula FROM y columna debe pertenecer a esa tabla.
– Si se usó la forma columna, debe haber exactamente una tabla mencionada en la cláusula FROM que tenga una columna con ese nombre.
* Las comparaciones sólo se pueden hacer entre campos o constantes de igual tipo.
* En la clausula SELECT pueden aparecer varios identificadores separados por comas, o funciones de agregados.
* Los identificadores pueden venir indicados mediante el atributo o bien mediante nombre_de_la_tabla.atributo.
* En la clausula FROM pueden aparecer varios nombres de tablas separados por comas.
* En la clausula WHERE pueden aparecer combinaciones de expresiones relacionales y operadores lógicos. Los operadores lógicos tienen mayor prioridad que los relacionales y que puede existir anidamiento en las expresiones.
* Supondremos que el programador puede escribir más de una sentencia SQL consecutiva. Una sentencia viene separada de otra por el carácter ;