Chequeo de Consistencia de Procedimientos Almacenados y Triggers

Joaquín Gardiola, Fabio Zorzan, Mariana Frutos, Marcela Daniele, Ariel Arsaute
Departamento de Computación
Facultad de Ciencias Exactas, Físico-Químicas y Naturales - Universidad Nacional de Río Cuarto
joa.gardiola@gmail.com,{fzorzan, mfrutos, marcela, aarsaute }@dc.exa.unrc.edu.ar

CONTEXTO

La línea de investigación presentada en este trabajo se desarrolla en el marco del proyecto "Ingeniería de Software: Evaluación de la calidad de sistemas de software y la mejora continua de los procesos de desarrollo", presentado en el Programa de Convocatoria PPI 2019 de la Secretaría de Ciencia y Técnica de la Universidad Nacional de Río Cuarto.

RESUMEN

Esta línea de investigación pretende contribuir con un aporte al desarrollo de aplicaciones que utilizan triggers y procedimientos almacenados en Bases de datos Relacionales, en particular PostgreSQL y MySQL, proponiendo el desarrollo de una herramienta que permita realizar chequeos semánticos adicionales a las instrucciones SQL incluidas en los códigos implementados en triggers y procedimientos Almacenados, permitiendo de esta manera proveer una solución que facilite detectar de manera temprana, problemas semánticos en las instrucciones SQL incluidas en estos mecanismos de programación de bases de datos relacionales.

Palabras Clave: Triggers, Procedimientos almacenados, chequeo semántico, SQL, MySQL, PostgreSQL

1. INTRODUCCIÓN

En la actualidad las bases de datos relacionales son herramientas ampliamente utilizadas en el desarrollo de sistemas de información[1]. Las bases de datos relacionales proveen mecanismos para programar dentro de la base de datos, estos mecanismos se denominan procedimientos almacenados y triggers[2], estas características se pueden encontrar en los motores de bases de datos MySQL[3] y PostgreSQL[4].

Cuando una base de datos PostgreSQL y MySQL cuenta con triggers y procedimientos almacenados, cuando se crean, se chequean sintáctica y semánticamente, pero hay ciertos chequeos semánticos que no se realizan, por ejemplo, se puede escribir como parte del código de un trigger una consulta SQL sobre tablas que no existen.

El crecimiento sostenido de la industria del software y su constante expansión a nuevos ámbitos, genera un estado de alerta y constante revisión de las metodologías de desarrollo de software utilizadas, como así también de los métodos de trabajo, la adopción de nuevas herramientas para automatizar y mejorar las actividades de gestión e ingeniería, como la planificación, el diseño y la implementación, involucradas en el proceso de producción de software. A pesar de las investigaciones y estudios realizados, existen aún numerosos casos en los que no se ha podido evitar que los errores de software lleguen a etapas de producción. El proyecto marco propone mejorar la calidad de los sistemas de software que se desarrollan, continuando con la investigación y el estudio de técnicas y procesos de desarrollo de software con diferentes enfoques.

El objetivo principal de esta línea es el desarrollo de una herramienta que pretende mejorar la calidad del software, en particular

software que utiliza una base de datos relacional, detectando en una etapa temprana del desarrollo, antes de llegar al entorno de producción, posibles inconsistencias del código SQL incorporado en procedimientos almacenados y triggers.

2. LINEAS DE INVESTIGACIÓN Y DESARROLLO

El trabajo está planificado para desarrollarse en tres etapas que se detallan a continuación:

2.1. PRIMERA ETAPA

En una primera etapa (ya concluida), se realizó un estudio del estado del arte y se analizaron las herramientas existentes que abordan, en parte, el problema planteado. Luego de hacer una revisión en portales científicos, distintos buscadores, documentos de investigación, foros, etc., se pudo observar que hasta la actualidad las formas de solucionar el problema de chequeo de consistencia en procedimientos almacenados y triggers en MySQL y PostgreSQL son algo rústicas, y no atacan al problema de una manera específica, como sería la creación de una herramienta de software que permita realizar chequeos semánticos de consistencia en el código de procedimientos almacenados y triggers en estas base de datos.

Una de las formas que se encontró como más "habitual" para resolver dicho problema, pero poco práctico, fue seguir la siguiente secuencia:

1. Realizar una copia de la Base de Datos.
2. Crear un script (secuencia de comandos que se utiliza para automatizar tareas repetitivas) con todos los procedimientos almacenados y triggers.
3. Realizar un "drop" (eliminación) de todos los procedimientos almacenados y triggers.
4. Por último, correr el script para recrearlos.

Siguiendo el procedimiento anterior, los procedimientos y triggers que no se puedan recrear arrojaran un error. Este método, si bien en parte da respuesta al problema, no detecta todos los problemas que se pueden detectar con un análisis a fondo usando una herramienta a tal fin. Por ejemplo, hay triggers que no arrojan errores cuando se crean, pero si cuando se ejecutan. Muchos de estos errores se podrían detectar de manera estática. La idea del presente trabajo es no tener que llegar a detectar el problema en tiempo de ejecución, sino permitir su detección en una etapa más temprana del desarrollo con un chequeo semántico estático.

2.2. SEGUNDA ETAPA

En esta etapa se definieron que tipos de chequeos específicos de bases de datos se implementarán en una primera instancia del trabajo. Además se realizó un estudio y evaluación de diferentes herramientas de análisis léxico – sintáctico y semántico, y del lenguaje de programación para el desarrollo. De esta forma se evaluó cuáles son las que mejor se adaptaban a los requisitos de la aplicación a desarrollar.

El lenguaje seleccionado fue Java, que además de contar con herramientas de análisis léxico – sintáctico y semántico, como son las librerías JFlex [5] para análisis léxico y Java_Cup [6] y Java_Cup-11a [6] para análisis sintáctico, provee mecanismos de fácil conexión a bases de datos PostgreSQL y MySQL. Cabe aclarar que la primera versión de la herramienta está enfocada en el motor PostgreSQL.

2.3. TERCERA ETAPA

Durante esta etapa se comenzó con el proceso para la construcción de la herramienta, la cual dispone de una fase de análisis donde se interpretan las entradas y se generan estructuras intermedias.

Esta fase estará estructurada de la siguiente forma:

- Análisis Léxico
- Análisis Sintáctico
- Análisis Semántico

A continuación, se describen cada una de estas fases:

2.3.1. ANÁLISIS LÉXICO

El analizador léxico, esquematizado en la Fig. 1, cumple la función de scanner, lee los caracteres

uno a uno desde la entrada y va formando grupos de caracteres con alguna relación entre sí (tokens), que constituirán la entrada para la siguiente etapa (análisis sintáctico). 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.

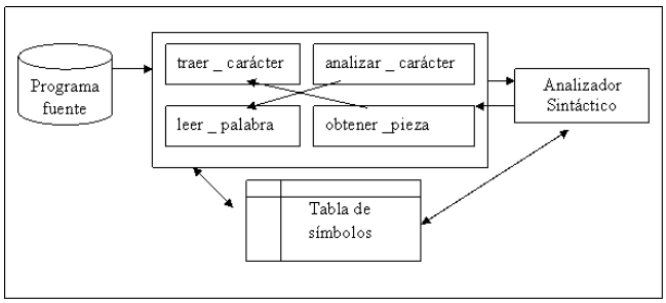

Fig.1: Esquema analizador Lexicográfico

Las funciones que realizará el analizador implementado en JFLEX son:

- Leer los caracteres de entrada del programa fuente (código fuente de Triggers y Procedimientos Almacenados).
- Generar la secuencia de componentes léxicos.
- Eliminar comentarios, espacios en blanco, y todo lo que esté excluido de la sintaxis del lenguaje.
- Reconocer identificadores, palabras reservadas del lenguaje, y tratarlos correctamente con respecto a la tabla de símbolos (solo en los casos que deben tratarse con dicha tabla).
- Avisar de errores léxicos. Por ejemplo, si @ no pertenece al lenguaje, avisar de un error.

Para llevar a cabo esta fase, se debe especificar las reglas léxicas. A continuación se muestra un fragmento de las reglas especificadas en el analizador lexicográfico implementado en JFLEX:

```
import static codigo.Tokens.*;
%%
%class Lexer
%type Tokens
%ignorecase

L=[a-zA-Z_]+
D=[0-9]+
espacio=[ ,\t,\r]+
%{
    public String lexeme;
%}
%%

( "," ) {lexeme = yytext(); return COMA;}

...

/* Palabra reservada Select*/
"select" {lexeme=yytext(); return SELECT;}
/* Palabra reservada Distinct */
( distinct ) {lexeme=yytext(); return
DISTINCT;}
/* Palabra reservada From */
( from ) {lexeme=yytext(); return FROM;}
/* Palabra reservada Where */
( where ) {lexeme=yytext(); return WHERE;}
...
```

2.3.2. ANÁLISIS SINTÁCTICO

El análisis sintáctico, esquematizado en la fig. 2, también llamado parser, se encargará de chequear el texto de entrada, suministrado por el analizador léxico, en base a una gramática dada (gramática del lenguaje SQL). En caso que el texto corresponda a una sentencia válida, suministrará el árbol sintáctico que la reconoce. Además de determinar si la gramática establecida por el lenguaje se respeta, el analizador sintáctico a implementar en Java_Cup tendrá las siguientes funciones:

•Acceder a la tabla de símbolos para armar el árbol sintáctico.

•Chequear los tipos.

•Generar código intermedio.

•Generar errores cuando se producen.

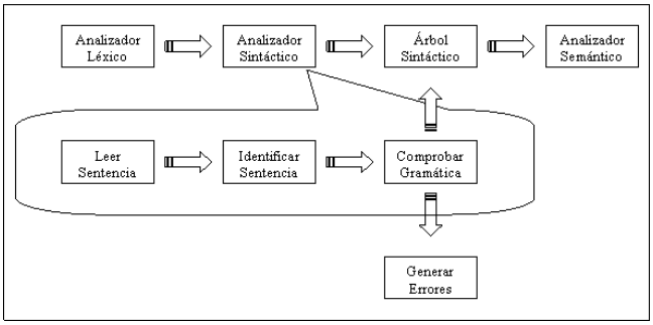

Fig.2: Esquema analizador Sintáctico

A continuación se muestra un fragmento de la especificación del analizador sintáctico implementado en JAVA_CUP:

```
import java_cup.runtime.Symbol;

parser code
{:
    private Symbol s;

    public void syntax_error(Symbol s){
        this.s = s;
    }

    public Symbol getS(){
        return this.s;
  }
:};
--
terminal SELECT, DISTINCT, FROM, WHERE,
MAYOR, DESC, ASC, GROUP, HAVING, ORDER,
     ID, NUM, P_COMA, COMA, OR, AND, EQ,
MENOR, LE, GE, NE, LIKE;
non terminal S, ST1, ST2, ST3, ST4, ST5,
ST6, attributeList, tableList, COND, E, F;

/* Precedencias */
precedence left ID, NUM;
precedence left OR, AND;
precedence left GROUP, HAVING, ORDER,
P_COMA, COMA;

start with S;
S::= ST1 P_COMA;
ST1::=SELECT attributeList FROM tableList ST2
    | SELECT DISTINCT attributeList FROM
      tableList ST2;
ST2 ::= WHERE COND ST3
    | ST3;
ST3 ::= GROUP attributeList ST4
    | ST4;
```

2.3.3. ANALIZADOR SEMÁNTICO

El analizador semántico revisa que las operaciones sean realizables al determinar que los tipos de las entidades sean correctos y que los operadores son capaces de manipularlos.

Algunas de las funciones más importantes, relacionadas específicamente al chequeo del código SQL incorporado en triggers y procedimientos almacenados, que se deberán realizar son:

- Comprobación de tipos en expresiones de filtros: La aplicación de los operadores y operandos deben ser compatibles (puede ser constantes, funciones, columnas de tablas, etc.).
- Comprobación de existencia de tablas y columnas en la base de datos.

Para implementar estos chequeos semánticos la herramienta deberá acceder al catálogo de la base de datos donde están definidos los procedimientos almacenados y triggers.

3. RESULTADOS OBTENIDOS/ESPERADOS

Los resultados de la investigación aportarán a la mejora del desarrollo de aplicaciones que utilizan Bases de datos Relacionales, y en particular, sus mecanismos de programación como lo son procedimientos almacenados y triggers, esto se logrará con el desarrollo de una herramienta de chequeos semánticos, adicionales a los realizados por los motores de bases de datos, mediante la utilización de herramientas para el desarrollo de parsers de lenguajes de programación.

El beneficio de esta herramienta se verá fundamentalmente en el desarrollo de aplicaciones que utilizan procedimientos almacenados y triggers en bases de datos relacionales, en particular PostgreSQL, detectando errores semánticos en una etapa temprana y antes que se produzcan en el entorno de producción.

En síntesis, los principales objetivos planteados en esta investigación son:

Específicos:

- ✓ Implementar un chequeador semántico de código SQL incorporado en procedimientos almacenados y triggers.
- ✓ Integrar la solución a desarrollar a herramientas oficiales de administración de bases de datos.

De Formación de los estudiante

- ✓ Potenciar el desarrollo de metodologías de trabajo científicas.
- ✓ Profundizar los conocimientos de ambientes de desarrollo y testing de aplicaciones.
- ✓ Desarrollar capacidades para la redacción de artículos científicos y documentación técnica.
- ✓ Profundizar la experiencia en el uso de bases de datos relacionales, en particular PostgreSQL.
- ✓ Adquirir experiencia en el uso de chequeadores sintácticos y semánticos de lenguajes de programación.

4. FORMACION DE RECURSOS HUMANOS

Los estudios realizados en esta línea de investigación sirven como marco para la elaboración de trabajos finales de grado. En este punto, dos estudiantes de la carrera Licenciatura en Ciencias de la Computación han iniciado su trabajo final que consiste en el desarrollo de la herramienta de cheuque semántico presentada en este trabajo.

Los temas abordados en esta línea de investigación brindan un fuerte aporte al proceso de perfeccionamiento continuo de los autores de este trabajo, que se desempeñan como docentes de las carreras de computación que se dictan en la Universidad Nacional de Río Cuarto y participan en asignaturas relacionadas a dichos temas.

5. BIBLIOGRAFIA

[1] Database System Concepts. 7th Edition, Edition Silberschatz, Korth, Sudarshan. McGraw Hill Company, 2019.

[2] Fundamentals of Database Systems. Elmasri, Navathe. 5th Edition Addison Wesley, 2006.

[3] Manual Oficial del motor de Bases de Datos MySQL, http://downloads.mysql.com/docs/refman-8.0-en.a4.pdf. Ultimo acceso 04/03/2022.

[4] Manual Oficial del motor de Bases de Datos PostgresSQL, https://www.postgresql.org/files/documentation/pdf/13/postgresql-13-A4.pdf. Ultimo acceso 01/03/2022.

[5] Manual oficial de JFlex https://www.jflex.de/. Ultimo acceso 23/02/2022.

[6] LALR Generator for JAVA http://www2.cs.tum.edu/projects/cup/. Ultimo acceso 24/02/2022.