PROYECTO BASES DE DATOS 2

Stiven Muñoz Murillo

Universidad de Caldas

25/02/2019

OBJETIVOS

- Implementar un paquete en PL/SQL que a través de backtracking (retroceso) y recursividad encuentre el camino adecuado para llegar a la salida de un laberinto representado por una estructura matricial.

- Comprender mejor el funcionamiento de Oracle como motor de base de datos y de PL/SQL como lenguaje de procedimientos y funciones.

- Aprender más acerca del manejo de excepciones en Oracle PL/SQL

- Aprender cómo manejar la información obtenida a través del motor de base de datos Oracle, e interpretar su respectivo retorno para la implementación desde una aplicación en el Frontend.

CÓDIGO PL/SQL

```
SET SERVEROUTPUT ON;



CREATE OR REPLACE PACKAGE PAQUETE_LABERINTO IS
  -- Declaraciones de tipos y registros públicos

  -- Declaraciones de variables y constantes publicas

  -- Declaraciones de procedimientos y funciones públicas
  FUNCTION LABERINTO(ANCHO NUMBER, ALTO NUMBER, START_X NUMBER, START_Y NUMBER,
END_X NUMBER, END_Y NUMBER) RETURN VARCHAR2;
END PAQUETE_LABERINTO;
/

CREATE OR REPLACE PACKAGE BODY PAQUETE_LABERINTO IS

-- INICIO DEL CUERPO DEL PAQUETE_LABERINTO
FUNCTION LABERINTO
(ANCHO NUMBER, ALTO NUMBER, START_X NUMBER, START_Y NUMBER, END_X NUMBER,
END_Y NUMBER)
RETURN VARCHAR2 IS

-- INICIO, DECLARACION DE VARIABLES
  -- RECORRIDO DEL LABERINTO
  RECORRIDO VARCHAR2(3000);

  -- VARIABLES PARA CALCULAR EL TIEMPO DE EJECUCION
  START_TIME NUMBER;
```

```
END_TIME NUMBER;

-- VECTOR DE NUMEROS
TYPE VECTOR IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;

-- MATRIZ DE VECTORES DE NUMEROS
TYPE MATRIX IS TABLE OF VECTOR INDEX BY BINARY_INTEGER;

-- INFORMACION DEL LABERINTO
MAZE MATRIX;

-- CONDICIONAL QUE INDICA SI SE ENCONTRO UN CAMINO
ENCONTRADO BOOLEAN := FALSE;

-- PAREDES Y CAMINOS DEL LABERINTO
PARED  NUMBER(1) := 0;
ESPACIO NUMBER(1) := 1;

-- PRIMER CAMINO ENCONTRADO
CAMINO NUMBER(1) := 2;

-- CONTORNO DEL LABERINTO
CONTORNO NUMBER(1) := 3;

-- DECLARO EXCEPCIONES
NO_CAMINO_EXCEPTION EXCEPTION;
NO_DIMENSIONES_EXCEPTION EXCEPTION;
FINAL_INCORRECTO_EXCEPTION EXCEPTION;
INICIO_INCORRECTO_EXCEPTION EXCEPTION;
```

```
-- FIN, DECLARACION DE VARIABLES



-- INICIO, DECLARACION DE PROCEDIMIENTOS
PROCEDURE ENCONTRAR_CAMINO_RECURSIVO(X NUMBER, Y NUMBER) IS
  -- VECTORES DE DIRECCIONES EN EL EJE X Y EN EL EJE Y
  DIR_X VECTOR;
  DIR_Y VECTOR;

  -- NUMERO ALEATORIO DEL 0 AL 3 QUE DETERMINA UNA DIRECCION ALEATORIA
  RAND_DIR  NUMBER(1) := DBMS_RANDOM.VALUE(0,3);

  -- CONTADOR QUE DETERMINA EL FIN DE LA EJECUCION DEL CICLO
  CONT NUMBER(1) := 0;

  -- POSICION A LA QUE AVANZO
  X1 NUMBER(9);
  Y1 NUMBER(9);

  -- POSICION A LA QUE PRETENDO AVANZAR
  X2 NUMBER(9);
  Y2 NUMBER(9);

  -- CONDICIONALES PARA LA VERIFICACION DEL CAMINO
  IZQUIERDA BOOLEAN;
  DERECHA BOOLEAN;
  ARRIBA BOOLEAN;
  ABAJO BOOLEAN;
  ENCONTRADO_X1 BOOLEAN := FALSE;
```

```
 ENCONTRADO_X2 BOOLEAN := FALSE;

BEGIN
 -- INICIALIZO EL VECTOR DE DIRECCIONES EN EL EJE X
 DIR_X(0) := 1;
 DIR_X(1) := -1;
 DIR_X(2) := 0;
 DIR_X(3) := 0;

 -- INICIALIZO EL VECTOR DE DIRECCIONES EN EL EJE Y
 DIR_Y(0) := 0;
 DIR_Y(1) := 0;
 DIR_Y(2) := 1;
 DIR_Y(3) := -1;

 WHILE CONT < 4 LOOP
  X1 := X + DIR_X(RAND_DIR);
  Y1 := Y + DIR_Y(RAND_DIR);
  X2 := X1 + DIR_X(RAND_DIR);
  Y2 := Y1 + DIR_Y(RAND_DIR);

  -- SI LA POSICIÓN A LA QUE AVANZO Y LA QUE PRETENDO AVANZAR SON PAREDES
  IF MAZE(X1)(Y1) = PARED AND MAZE(X2)(Y2) = PARED THEN
   -- SI YA ENCONTRE UN CAMINO SIGO CONSTRUYENDO EL LABERINTO SIN MARCAR EL CAMINO
   IF ENCONTRADO THEN
    MAZE(X1)(Y1) := ESPACIO;
    MAZE(X2)(Y2) := ESPACIO;
   ELSE
    MAZE(X1)(Y1) := CAMINO;
```

```
  MAZE(X2)(Y2) := CAMINO;
END IF;

-- AGREGO LAS COORDENADAS DE LAS PAREDES DESTRUIDAS AL RECORRIDO
RECORRIDO := RECORRIDO || ' (' || X1 || ',' || Y1 || ')';
RECORRIDO := RECORRIDO || ' (' || X2 || ',' || Y2 || ')';

-- VERIFICO SI YA ENCONTRE UN CAMINO A LA POSICION FINAL
IZQUIERDA := X1 + DIR_X(0) = END_X AND Y1 + DIR_Y(0) = END_Y;
DERECHA   := X1 + DIR_X(1) = END_X AND Y1 + DIR_Y(1) = END_Y;
ARRIBA    := X1 + DIR_X(2) = END_X AND Y1 + DIR_Y(2) = END_Y;
ABAJO     := X1 + DIR_X(3) = END_X AND Y1 + DIR_Y(3) = END_Y;
ENCONTRADO_X1 := IZQUIERDA OR DERECHA OR ARRIBA OR ABAJO;

-- VERIFICO TAMBIEN EN TODAS LA DIRECCIONES DE X2
IZQUIERDA := X2 + DIR_X(0) = END_X AND Y2 + DIR_Y(0) = END_Y;
DERECHA   := X2 + DIR_X(1) = END_X AND Y2 + DIR_Y(1) = END_Y;
ARRIBA    := X2 + DIR_X(2) = END_X AND Y2 + DIR_Y(2) = END_Y;
ABAJO     := X2 + DIR_X(3) = END_X AND Y2 + DIR_Y(3) = END_Y;
ENCONTRADO_X2 := IZQUIERDA OR DERECHA OR ARRIBA OR ABAJO;

-- SI ENCUENTRO UN CAMINO AVANZO HASTA EL FINAL
-- Y ME REGRESO A SEGUIR ARMANDO EL LABERINTO
IF ENCONTRADO_X1 THEN
  ENCONTRADO := TRUE;

  RECORRIDO := RECORRIDO || ' (' || X1 || ',' || Y1 || ')';
  RECORRIDO :=  RECORRIDO || ' (' || END_X || ',' || END_Y || ')';
  RECORRIDO := RECORRIDO || ' (' || X1 || ',' || Y1 || ')';
```

```
    ELSIF ENCONTRADO_X2 THEN
      ENCONTRADO := TRUE;

      RECORRIDO := RECORRIDO || ' (' || X2 || ',' || Y2 || ')';
      RECORRIDO :=  RECORRIDO || ' (' || END_X || ',' || END_Y || ')';
      RECORRIDO := RECORRIDO || ' (' || X2 || ',' || Y2 || ')';
    END IF;

    -- LLAMADO RECURSIVO DE LA FUNCION
    ENCONTRAR_CAMINO_RECURSIVO(X2, Y2);
   ELSE
    -- MIRAR HACIA OTRA DIRECCION
    RAND_DIR := (RAND_DIR + 1) MOD 4;

    -- CANTIDAD DE DIRECCIONES VISTAS AUMENTA EN UNO
    CONT := CONT + 1;
   END IF;
  END LOOP;
END ENCONTRAR_CAMINO_RECURSIVO;


PROCEDURE CREAR_LABERINTO IS
  -- CONDICIONALES PARA LA VERIFICACION DE BORDES
  BORDE_IZQ  BOOLEAN;
  BORDE_DER  BOOLEAN;
  BORDE_UP   BOOLEAN;
  BORDE_DOWN BOOLEAN;

  -- CONDICIONALES PARA LA VERIFICACION DE ESQUINAS
```

```
 ESQUINA_SUP_IZQ BOOLEAN;
 ESQUINA_SUP_DER BOOLEAN;
 ESQUINA_INF_IZQ BOOLEAN;
 ESQUINA_INF_DER BOOLEAN;

BEGIN
 -- SI LAS DIMENSIONES SON PARES LANZO UNA EXCEPCION
 IF (ANCHO MOD 2 = 0) OR (ANCHO MOD 2 = 0) THEN
   RAISE NO_DIMENSIONES_EXCEPTION;
 ELSIF (ALTO MOD 2 = 0) OR (ALTO MOD 2 = 0) THEN
   RAISE NO_DIMENSIONES_EXCEPTION;
 END IF;

 -- LLENO EL LABERINTO CON PAREDES
 FOR X IN 0..(ANCHO-1) LOOP
   FOR Y IN 0..(ALTO-1) LOOP
    MAZE(X)(Y) := PARED;
   END LOOP;
 END LOOP;

 -- AGREGO LAS LINEAS HORIZONTALES DEL CONTORNO DEL LABERINTO
 FOR X IN 0..(ANCHO-1) LOOP
   MAZE(X)(0) := CONTORNO;
   MAZE(X)(ALTO-1) := CONTORNO;
 END LOOP;

 -- AGREGO LAS LINEAS VERTICALES DEL CONTORNO DEL LABERINTO
 FOR Y IN 0..(ALTO-1) LOOP
   MAZE(0)(Y) := CONTORNO;
```

```
  MAZE(ANCHO-1)(Y) := CONTORNO;
END LOOP;

-- SI LA SALIDA NO ESTA EN EL BORDE DEL LABERINTO LANZO UNA EXCEPCION
BORDE_IZQ := END_X = 1 AND (END_Y>=1 AND END_Y<=ALTO-2);
BORDE_DER := END_X = ANCHO-2 AND (END_Y>=1 AND END_Y<=ALTO-2);
BORDE_UP := END_Y = 1 AND (END_X>=1 AND END_X<=ANCHO-2);
BORDE_DOWN := END_Y = ALTO-2 AND (END_X>=1 AND END_X<=ANCHO-2);

IF NOT(BORDE_IZQ OR BORDE_DER OR BORDE_UP OR BORDE_DOWN) THEN
  RAISE FINAL_INCORRECTO_EXCEPTION;
END IF;

-- SI LA SALIDA ESTA EN LAS ESQUINAS LANZO UNA EXCEPCION
ESQUINA_SUP_IZQ := END_X = 1 AND END_Y = 1;
ESQUINA_SUP_DER := END_X = ANCHO-2 AND END_Y = 1;
ESQUINA_INF_IZQ := END_X = 1 AND END_Y = ALTO-2;
ESQUINA_INF_DER := END_X = ANCHO-2 AND END_Y = ALTO-2;

IF ESQUINA_SUP_IZQ OR ESQUINA_SUP_DER OR ESQUINA_INF_IZQ OR ESQUINA_INF_DER THEN
  RAISE FINAL_INCORRECTO_EXCEPTION;
END IF;

-- INSTANCIO LA COORDENADA DE SALIDA
MAZE(END_X)(END_Y) := 5;

-- SI EL INICIO ES IMPAR LANZO UNA EXCEPCION
IF (START_X MOD 2 != 0) OR (START_Y MOD 2 != 0) THEN
  RAISE INICIO_INCORRECTO_EXCEPTION;
```

```
  END IF;

  -- Y SI EL INICIO NO ESTA DENTRO DEL BORDE DEL LABERINTO
  IF NOT(START_X > 1 AND START_X < ANCHO-2 AND START_Y > 1 AND START_Y < ALTO-2) THEN
    RAISE INICIO_INCORRECTO_EXCEPTION;
  END IF;

  -- INSTANCIO LA COORDENADA DE INICIO
  MAZE(START_X)(START_Y) := 8;

  -- COMIENZO A ENCONTRAR EL CAMINO DE FORMA RECURSIVA
  ENCONTRAR_CAMINO_RECURSIVO(START_X, START_Y);

  -- SI NO SE ENCONTRO CAMINO LANZO UNA EXCEPCION
  IF NOT ENCONTRADO THEN
    RAISE NO_CAMINO_EXCEPTION;
  END IF;
END CREAR_LABERINTO;



PROCEDURE MOSTRAR_LABERINTO IS
BEGIN
  -- RECORRO EL LABERINTO Y LO IMPRIMO
  FOR Y IN 0..(ALTO-1) LOOP
    FOR X IN 0..(ANCHO-1) LOOP
      DBMS_OUTPUT.PUT(MAZE(X)(Y));
      /*
      IF MAZE(X)(Y) = PARED THEN
        DBMS_OUTPUT.PUT('[]');
```

```
    ELSE
      DBMS_OUTPUT.PUT(' ');
    END IF;
    */
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('');
  END LOOP;
END MOSTRAR_LABERINTO;
-- FIN, DECLARACION DE PROCEDIMIENTOS

BEGIN
  -- INICIO, EJECUCION DE LA FUNCION LABERINTO
  START_TIME := DBMS_UTILITY.GET_TIME();

  CREAR_LABERINTO;
  MOSTRAR_LABERINTO;

  -- TERMINO, LA EJECUCION DE LA FUNCION LABERINTO
  END_TIME := DBMS_UTILITY.GET_TIME() - START_TIME;

-- RETORNO LA INFORMACION OBTENIDA EN UN JSON
RETURN '{"ERRORES":"NO", "RECORRIDO":"'||RECORRIDO||' ",
"TIEMPO_ms":"'||END_TIME||'","MENSAJE":"Todo Correcto."}';

-- CONTROLO LAS EXCEPCIONES DE LA FUNCION LABERINTO
EXCEPTION
  -- EXCEPCIONES CREADAS
  WHEN NO_CAMINO_EXCEPTION THEN
    RETURN '{"ERRORES":"SI", "RECORRIDO":"  ", "TIEMPO_ms":"0",
```

```
   "MENSAJE":"No se encontro un camino, por favor vuelve a intentarlo."}';
  WHEN NO_DIMENSIONES_EXCEPTION THEN
   RETURN '{"ERRORES":"SI", "RECORRIDO":"  ", "TIEMPO_ms":"0",
   "MENSAJE":"Upps, el ancho y el alto del laberinto deben ser impares."}';
  WHEN FINAL_INCORRECTO_EXCEPTION THEN
   RETURN '{"ERRORES":"SI", "RECORRIDO":"  ", "TIEMPO_ms":"0",
   "MENSAJE":"La coordenada final debe estar en el borde del laberinto, sin incluir las esquinas."}';
  WHEN INICIO_INCORRECTO_EXCEPTION THEN
   RETURN '{"ERRORES":"SI", "RECORRIDO":"  ", "TIEMPO_ms":"0",
   "MENSAJE":"La coordenada inicial debe estar dentro del borde del laberinto y debe ser par."}';

  -- EXCEPCIONES DE ORACLE
  WHEN VALUE_ERROR THEN
   RETURN '{"ERRORES":"SI", "RECORRIDO":"  ", "TIEMPO_ms":"0",
   "MENSAJE":"VALUE_ERROR -> Quizá el camino encontrado es muy grande, trata con un
laberinto más pequeño."}';
  WHEN OTHERS THEN
   RETURN '{"ERRORES":"SI", "RECORRIDO":"  ", "TIEMPO_ms":"0",
   "MENSAJE":"Upps, ocurrio algo inesperado. '||SQLERRM||'"}';

END LABERINTO;
-- FIN DEL CUERPO DEL PAQUETE_LABERINTO

END PAQUETE_LABERINTO;
/


-- LLAMADO DE LA FUNCION LABERINTO DEL PAQUETE LABERINTO
SELECT PAQUETE_LABERINTO.LABERINTO(11,9,4,4,9,5) FROM DUAL;
```

CONCLUSIONES

- Se aprendieron y repasaron algunos conceptos importantes al realizar este proyecto.

- Al indagar más acerca del lenguaje PL/SQL se encontró que tiene una sintaxis basada en el lenguaje de programación ADA con Pascal como su antepasado común, ***https://en.wikipedia.org/wiki/PL/SQL#Similar_languages***. Esto hizo más sencillo el entendimiento del lenguaje para su posterior implementación.

- Al investigar los diferentes algoritmos para la exploración de caminos en laberintos decidí utilizar una modificación del algoritmo de retroceso recursivo, ***https://en.wikipedia.org/wiki/Maze_generation_algorithm#Recursive_backtracker***. La razón por la que decidí utilizarlo es la relación entre su baja dificultad de implementación contra su avanzada complejidad a la hora de generar laberintos.

- Para manejar las excepciones de la función contenida en el paquete de Oracle se decidió lanzar excepciones propias y comunicar sus respectivos mensajes a través de un retorno en formato JSON; dicho retorno contiene los errores, el recorrido de Laberinto y el tiempo de ejecución de la operación en el motor de bases de datos.

- Leyendo toda la información obtenida como resultado de la consulta al motor de base de datos Oracle en formato JSON procedí a realizar la aplicación en el Frontend, para ello recopilé dicha información utilizando el lenguaje de programación PHP y la envié. Estando en el Frontend analizo dicha información para saber si ocurrió algún error o se pudo hallar un camino, y muestro dicha información al usuario.

- Para mostrar el camino obtenido hago uso de expresiones regulares para obtener las coordenadas, luego muestro una animación al usuario recorriendo dichos puntos obtenidos en un intervalo de tiempo.

- En general, se lograron los objetivos siendo la realización de este taller una experiencia de aprendizaje enriquecedora.