Programación de Base de Datos

Documentación de práctica 6

Trabajo realizado por:

Carlos Guillén Moreno

Para el profesor: Andrés Caro Lindo (correo: andresc@unex.es)

Contenido

INTRODUCCIÓN.

Para esta práctica, se usan las herramientas instaladas en la sesión anterior. Se utilizará el gestor de base de datos (Oracle 11g Express Edition) y el entorno (SQL Developer).

EJERCICIOS

APARTADO A

Sobre cualquiera de los ejemplos anteriores, se propone añadir, al menos, **tres nuevos procedimientos** en los que se realicen consultas espaciales que no sean del estilo de las propuestas en el presente documento. En el presente documento se ha hecho uso exclusivamente de tres funciones (SDO_GEOM.SDO_DISTANCE, SDO_GEOM.SDO_AREA y SDO_GEOM.RELATE). Se propone que se usen nuevas funciones en los procedimientos espaciales desarrollados (**que no sean SDO_DISTANCE ni SDO_AREA**), consultado la url que se proponía con anterioridad:

SDO_GEOM Package (Geometry)
https://docs.oracle.com/cd/B28359_01/appdev.111/b28400/sdo_objgeom.htm#SPATL120

Para la realización de esta práctica, he decidido utilizar el scrip **Espacial_2** de sql facilitado por el profesor. En él he insertado los tres métodos que me piden acorde a las exigencias del enunciado. Estos serían los métodos:

Procedimiento perimetroEmbalse

Este método muestra el perímetro de todos los embalses de Cáceres. Recordar que se ha creado una geometría principal llamada Extremadura y sobre ella se insertan los elementos de tipo localidad, embalses, autovías y zonas turísticas. Este sería el método:

```
  --Muestra el perimetro de todos los embalses
    PROCEDURE perimetroEmbalse IS
    geomEntrada SDO_GEOMETRY;
    perimetro NUMBER;
    nombreEmbalse Extremadura.Nombre%TYPE;
    tupla Extremadura%ROWTYPE;
    dim SDO_DIM_ARRAY;
    -- Cursor para recuperar los embalses
    CURSOR cursor_Embalses IS
      SELECT *
      FROM Extremadura
      WHERE Tipo = 'Embalse';

  BEGIN

    DBMS_OUTPUT.PUT_LINE('Procedimiento 1');
    OPEN cursor_Embalses;
    LOOP
        FETCH cursor_Embalses INTO tupla;
        EXIT WHEN cursor_Embalses%NOTFOUND;

            SELECT Geom INTO geomEntrada
            FROM Extremadura
            WHERE Nombre = tupla.Nombre;

            SELECT DIMINFO INTO dim
            FROM USER_SDO_GEOM_METADATA
            WHERE table_name='EXTREMADURA';

            nombreEmbalse:=tupla.Nombre;


            perimetro:=SDO_GEOM.SDO_LENGTH(geomEntrada,dim);
            DBMS_OUTPUT.PUT_LINE('El perimetro del embalse
'||nombreEmbalse||' es de '||perimetro);
    END LOOP;
    CLOSE cursor_Embalses;
    DBMS_OUTPUT.PUT_LINE('-----------------------------------------
----------------------------------------');
  END perimetroEmbalse;
```

Explicación de método: He decidido llamar al método perimetroEmbalse, y en este caso, no tiene parámetros de entrada ya que busca todos los embalses y no ninguno en específico. Lo primero que hago es crear un cursor para recuperar todas las tuplas de tipo “Embalse” que se corresponde con las geometrías de tipo embalse insertadas en la geometría principal de “Extremadura”. Dicho cursor lo nombro como “cursor_Embalses” y en él, selecciona todas las geometrías de la geometría Extremadura en las el atributo Tipo tome el valor de “Embalse”.

A continuación comenzamos mostrando un mensaje por pantalla usando la función `DBMS_OUTPUT.PUT_LINE` en el que se dice “Procedimiento 1”. Ahora almacenamos la dimensión de la geometría Extremadura sobre las que se encuentran todos los embalses en una variable local y auxiliar llamada “dim”. Esta variable la necesitaremos más adelante. Abrimos el cursor para trabajar con él e iniciamos el bucle. La condición del bucle dice que siga iterando hasta que dentro del cursor no haya tuplas sobre las que trabajar, eso se controla con esta parte del código →

```
EXIT WHEN cursor_Embalses%NOTFOUND;
```

Cada tupla del cursor la guardaremos sobre una variable local y auxiliar llamada **tupla** que es de tipo geometría.

A continuación almacenamos en la variable local “perímetro” el resultado de usar la función `SDO_GEOM.SDO_LENGTH` que usa de parámetros la geometría de la tupla (tupla.Geom) y la dimensión (dim) que habíamos almacenado previamente. Esta función calcula el perímetro de una geometría. Se muestra un mensaje por pantalla usando la función `DBMS_OUTPUT.PUT_LINE` en el que se muestra cada perímetro de cada embalse específicamente. Se cierra el bucle (`END LOOP`) y seguidamente el cursor (`CLOSE cursor_Embalses`). Se muestra por pantalla una serie de guiones para indicar que se acaba el método y separarlo del siguiente. Se finaliza el método con `END perimetroEmbalse;`

Esta sería la llamada usando este método:

```
perimetroEmbalse();
```

Y este el resultado:

```
Procedimiento 1
El perimetro del embalse   EAlcántara es de 5,23606797749979
El perimetro del embalse   EOrellana es de 6,47213595499958
El perimetro del embalse   ESerena es de 7,08276253029822
--------------------------------------------------------------------------
--------------
```

Procedimiento distanciaLocalidadEmbalse

Este método muestra la distancia entre una localidad y un embalse de Cáceres. Recordar que se ha creado una geometría principal llamada Extremadura y sobre ella se insertan los elementos de tipo localidad, embalses, autovías y zonas turísticas. Este sería el método:

```
  --Devuelve la distancia entre  una localidad y un embalse
  PROCEDURE distanciaLocalidadEmbalse (pLocalidad IN VARCHAR2,
pEmbalse IN VARCHAR2) IS
    geomEntrada1 SDO_GEOMETRY;
    geomEntrada2 SDO_GEOMETRY;
    puntosLocalidad SDO_GEOMETRY;
    puntosEmbalse SDO_GEOMETRY;
    dim1 SDO_DIM_ARRAY;
    distancia NUMBER;


    BEGIN
    DBMS_OUTPUT.PUT_LINE('Procedimiento 2');

    -- Obtener dim
    SELECT DIMINFO INTO dim1
    FROM USER_SDO_GEOM_METADATA
    WHERE table_name='EXTREMADURA'; --Ojo, EXTREMADURA en maysculas

    SELECT Geom INTO geomEntrada1
    FROM Extremadura
    WHERE Nombre = pLocalidad AND Tipo='Localidad';

    SELECT Geom INTO geomEntrada2
    FROM Extremadura
    WHERE Nombre = pEmbalse AND Tipo='Embalse';

    --Almaceno los puntos de la localidad
    puntosLocalidad:=SDO_GEOM.SDO_CENTROID(geomEntrada1, dim1);
    --Almaceno los puntos del embalse
    puntosEmbalse:=SDO_GEOM.SDO_CENTROID(geomEntrada2, dim1);

    distancia:=SDO_GEOM.SDO_DISTANCE(puntosLocalidad, dim1,
puntosEmbalse, dim1);

    DBMS_OUTPUT.PUT_LINE('La distancia entre la localidad
'||pLocalidad||' y el embalse '||pEmbalse||' es de '||distancia);

    DBMS_OUTPUT.PUT_LINE('-----------------------------------------
---------------------------------------');
   END distanciaLocalidadEmbalse;
```

Explicación de método: He decidido llamar al método distanciaLocalidadEmbalse, en este caso, tiene de parámetros de entrada el nombre de la localidad y el nombre del embalse de los cuales se quiere averiguar la distancia.

A continuación comenzamos mostrando un mensaje por pantalla usando la función DBMS_OUTPUT.PUT_LINE en el que se dice “Procedimiento 2”. Ahora almacenamos la dimensión de la geometría Extremadura sobre las que se encuentran todos los embalses en una variable local y auxiliar llamada “dim1”.

```
-- Obtener dim
SELECT DIMINFO INTO dim1
FROM USER_SDO_GEOM_METADATA
WHERE table_name='EXTREMADURA'; --Ojo, EXTREMADURA en maysculas
```

Ahora hay que almacenar la geometría de la localidad que entramos el nombre por parámetro (pLocalidad) en la variable auxiliar y local llamada geomEntrada1 de tipo SDO_GEOMETRY.

```
SELECT Geom INTO geomEntrada1
FROM Extremadura
WHERE Nombre = pLocalidad AND Tipo='Localidad';
```

También se almacena la geometría del embalse que entramos el nombre por parámetro (pEmbalse) en la variable auxiliar y local llamada geomEntrada2 de tipo SDO_GEOMETRY.

```
SELECT Geom INTO geomEntrada2
FROM Extremadura
WHERE Nombre = pEmbalse AND Tipo='Embalse';
```

A continuación, almacenamos en la variable local puntosLocalidad de tipo SDO_GEOMETRY el centroide de la localidad, para ello utilizamos la función SDO_GEOM.SDO_CENTROID en la que introducimos como parámetros la geometría de la localidad (geomEntrada1) y la dimensión que almacena a la localidad y que guardamos previamente (dim1).

```
puntosLocalidad:=SDO_GEOM.SDO_CENTROID(geomEntrada1, dim1);
```

Habría que hacer lo mismo para almacenar el centroide del embalse en la variable local puntosEmbalse.

```
puntosEmbalse:=SDO_GEOM.SDO_CENTROID(geomEntrada2, dim1);
```

Una vez obtenidos dichos puntos es posible conseguir la distancia entre una localidad y el centro de un embalse. Para ello usaremos la función SDO_GEOM.SDO_DISTANCE en la que introduciremos el centroide de la localidad (puntosLocalidad), la dimensión en la que se encuentra la localidad (dim1), el centroide del embalse (puntosEmbalse) y la dimensión en la que se encuentra el embalse.

```
distancia:=SDO_GEOM.SDO_DISTANCE(puntosLocalidad, dim1, puntosEmbalse,
dim1);
```

Se almacena en la variable local distancia de tipo NUMBER y seguidamente se muestra un mensaje en el que se nombra la localidad, el embalse y la distancia.

```
DBMS_OUTPUT.PUT_LINE('La distancia entre la localidad '||pLocalidad||'
y el embalse '||pEmbalse||' es de '||distancia);
```

Se muestra por pantalla una serie de guiones para indicar que se acaba el método y separarlo del siguiente. Se finaliza el método con `END distanciaLocalidadEmbalse;`

Esta sería la llamada usando este método:

```
distanciaLocalidadEmbalse ('Alcantara','EOrellana');
```

Y este el resultado:

```
Procedimiento 2
La distancia entre la localidad Alcántara y el embalse EOrellana es de
8,00347146902864
--------------------------------------------------------------------------
--------------
```

Procedimiento intZonaTurEmb

Este método me muestra si existe una intersección entre una zona turística y un embalse de Cáceres. Recordar que se ha creado una geometría principal llamada Extremadura y sobre ella se insertan los elementos de tipo localidad, embalses, autovías y zonas turísticas. Este sería el método:

```
  --Metodo que me devuelve la interseccion entre una zona turistica y
un embalse en caso de que la haya
  PROCEDURE intZonaTurEmb (pTurismo IN VARCHAR2, pEmbalse IN VARCHAR2)
IS
    geomEntrada1 SDO_GEOMETRY;
    geomEntrada2 SDO_GEOMETRY;
    geomSalida3 SDO_GEOMETRY;
    dim SDO_DIM_ARRAY;



  BEGIN

  DBMS_OUTPUT.PUT_LINE('Procedimiento 3');
  -- Recuperar la geometria del parametro de entrada
    SELECT Geom INTO geomEntrada1
    FROM Extremadura
    WHERE Nombre = pTurismo AND Tipo='Turismo';

    -- Recuperar la geometria del parametro de entrada
    SELECT Geom INTO geomEntrada2
    FROM Extremadura
    WHERE Nombre = pEmbalse AND Tipo='Embalse';

    SELECT DIMINFO INTO dim
    FROM USER_SDO_GEOM_METADATA
    WHERE table_name='EXTREMADURA';
```

```
    geomSalida3:= SDO_GEOM.SDO_INTERSECTION(geomEntrada1, dim,
geomEntrada2, dim);


    --si tiene intersección la muestro
    IF geomSalida3 IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Existe interseccion entre '||
pTurismo||' y '||pEmbalse||'.');
    --sino tiene intersección lo muestro por pantalla
    ELSE
        DBMS_OUTPUT.PUT_LINE('No existe intersección entre las dos
geometrías');
    END IF;


    DBMS_OUTPUT.PUT_LINE('----------------------------------------
---------------------------------------');


  END intZonaTurEmb;
```

Explicación de método: He decidido llamar al método intZonaTurEmb, en este caso, tiene de parámetros de entrada el nombre de la zona turística y el nombre del embalse de los cuales se quiere averiguar si hay o no intersección.

A continuación comenzamos mostrando un mensaje por pantalla usando la función `DBMS_OUTPUT.PUT_LINE` en el que se dice “Procedimiento 3”.

Ahora hay que almacenar la geometría de la zona turística que entramos el nombre por parámetro (pTurismo) en la variable auxiliar y local llamada geomEntrada1 de tipo `SDO_GEOMETRY`.

```
    SELECT Geom INTO geomEntrada1
    FROM Extremadura
    WHERE Nombre = pTurismo AND Tipo='Turismo';
```

También se almacena la geometría del embalse que entramos el nombre por parámetro (pEmbalse) en la variable auxiliar y local llamada geomEntrada2 de tipo `SDO_GEOMETRY`.

```
    SELECT Geom INTO geomEntrada2
    FROM Extremadura
    WHERE Nombre = pEmbalse AND Tipo='Embalse';
```

Ahora almacenamos la dimensión de la geometría Extremadura sobre las que se encuentran todos los embalses en una variable local y auxiliar llamada “dim”.

```
    SELECT DIMINFO INTO dim
    FROM USER_SDO_GEOM_METADATA
    WHERE table_name='EXTREMADURA';
```

Con esta información ya podríamos utilizar la nueva función `SDO_GEOM.SDO_INTERSECTION` con la que podríamos averiguar la intersección entre dos geometrías. Los parámetros que utiliza la función son la geometría de la zona turística (geomEntrada1), la dimensión donde se encuentra la geometría (dim), la geometría del embalse (geomEntrada2) y la dimensión donde

se encuentra (dim). Ese resultado lo almaceno en la variable local geomSalida3 que es de tipo SDO_GEOMETRY.

```
geomSalida3:= SDO_GEOM.SDO_INTERSECTION(geomEntrada1, dim,
geomEntrada2, dim);
```

Para comprobar si existe intersección entre ambas geometrías, compruebo que la geometría resultante de ambas no sea nula. Si no es nula significa que existe intersección y muestro un mensaje por pantalla confirmándolo. `DBMS_OUTPUT.PUT_LINE('Existe interseccion entre '|| pTurismo||' y '||pEmbalse||'.');`

En cambio, si la geometría resultante de la intersección es nula significa que no hay intersección y por lo que habrá que mostrar un mensaje diciendo que no existe intersección.

```
DBMS_OUTPUT.PUT_LINE('No existe intersección entre las dos
geometrías');
```

Código de la comprobación:

```
    --si tiene intersección la muestro
    IF geomSalida3 IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Existe intersección entre '||
pTurismo||' y '||pEmbalse||'.');
    --sino tiene intersección lo muestro por pantalla
    ELSE
        DBMS_OUTPUT.PUT_LINE('No existe intersección entre las dos
geometría');
    END IF;
```

Se muestra por pantalla una serie de guiones para indicar que se acaba el método y separarlo del siguiente. Se finaliza el método con `END intZonaTurEmb;`

Esta sería la llamada usando este método:

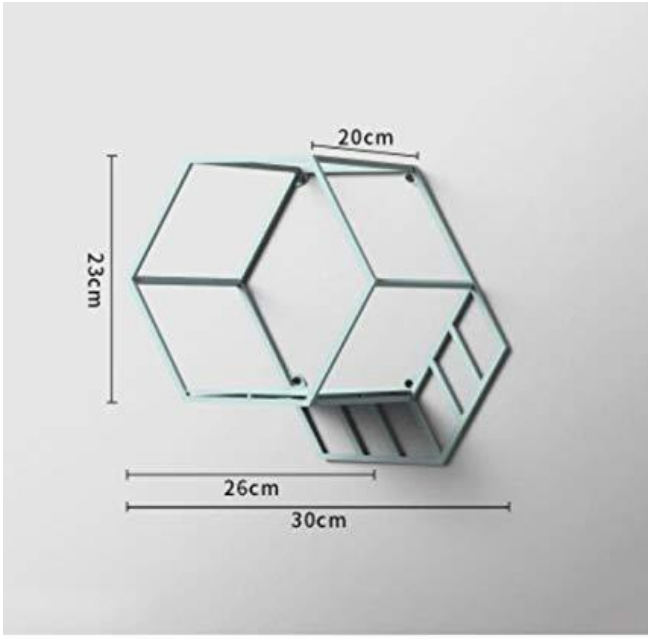
```
intZonaTurEmb ('Zona1', 'EAlcantara');
```

Y este el resultado:

```
Procedimiento 3
Existe interseccion entre Zona1 y EAlcántara.
--------------------------------------------------------------------------
--------------
```

APARTADO B

Sobre cualquiera de los dos ejemplos anteriores, añadir alguna geometría nueva y realizar alguna operación espacial que la tenga en cuenta.



Geometría insertada

Esta vez he decidido realizar la otra parte de la entrega sobre el archivo “Espacial_1.sql”. He decidido añadir una nueva Geometría de tipo **Luz** para hacer referencia a farolas que insertaré por la geometría de Cáceres. He pensado en insertar farolas debido a que al haber personas y barrios insertados en la geometría Cáceres, sería un problema que al caer la noche las personas no pudiesen dar un paseo por falta de luz por sus barrios…

Este sería el código correspondiente a la inserción de las farolas en la geometría de Cáceres:

```
--Insertar farolas
INSERT INTO Caceres VALUES(
numGeom.nextval,
'Farola1',
'Luz',
SDO_GEOMETRY(
    2001,
    NULL,
    SDO_POINT_TYPE(-4,5,NULL),
    NULL,
    NULL
  )
);

INSERT INTO Caceres VALUES(
numGeom.nextval,
'Farola2',
'Luz',
SDO_GEOMETRY(
    2001,
    NULL,
    SDO_POINT_TYPE(3,0,NULL),
    NULL,
    NULL
  )
);

INSERT INTO Caceres VALUES(
numGeom.nextval,
'Farola3',
'Luz',
SDO_GEOMETRY(
    2001,
    NULL,
    SDO_POINT_TYPE(6,5,NULL),
    NULL,
    NULL
  )
);

INSERT INTO Caceres VALUES(
numGeom.nextval,
'Farola4',
'Luz',
SDO_GEOMETRY(
    2001,
    NULL,
    SDO_POINT_TYPE(3,-1,NULL),
    NULL,
    NULL
  )
);
```

Los campos que forman esta geometría serían el código identificador (utilizamos una secuencia para ir incrementando los valores), el nombre (farola+nº de farolas insertadas), el tipo (Luz) y las características de la nueva geometría que en este caso sería 2001:

- El 2 representa el número de dimensiones.
- El primer 0 significa que no tiene LRS (sistema de referencia lineal).
- El 01 corresponde a la geometría de tipo punto.

Procedimiento farolaMasCercana

Este método muestra la distancia entre una persona específica y una farola de Cáceres. Además, en el caso de que una persona y su farola más cercana estén distanciados más de 5 metros, se mostrará un mensaje de advertencia. Recordar que se ha creado una geometría principal llamada Cáceres y sobre ella se insertan los elementos de tipo persona, barrios y farolas. Este sería el método:

```
  PROCEDURE farolaMasCercana(pnombre IN VARCHAR2) IS
    geomEntrada SDO_GEOMETRY;
    dist NUMBER;
    distMin NUMBER;
    tupla Caceres%ROWTYPE;
    dim SDO_DIM_ARRAY;
    nomFarola caceres.Nombre%TYPE;
    -- Cursor para recuperar las farolas
    CURSOR cursor_farola IS
      SELECT *
      FROM caceres
      WHERE tipo = 'Luz';

  BEGIN
  DBMS_OUTPUT.PUT_LINE('-----------------------------------------
-------------------------------------');
    distMin := 9999;

    -- Recuperar la geometria del parametro de entrada
    SELECT Geom INTO geomEntrada
    FROM Caceres
    WHERE Nombre = pnombre;

    -- Obtener dim
    SELECT DIMINFO INTO dim
    FROM USER_SDO_GEOM_METADATA
    WHERE table_name='CACERES'; --Ojo, CACERES en maysculas

    -- Recorrer todos las farolas
    OPEN cursor_farola;
    LOOP
      FETCH cursor_farola INTO tupla;
      EXIT WHEN cursor_farola%NOTFOUND;
      dist := SDO_GEOM.SDO_DISTANCE(geomEntrada,dim,tupla.Geom,dim);
      IF (dist < distMin) and (tupla.Nombre <> pnombre) THEN
        distMin := dist;
        nomFarola := tupla.Nombre;
      END IF;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('La farola m□cercana a '||pnombre||' es
'||nomFarola||', a una distancia de '||distMin);
    IF distMin<5.0 THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Tienes luz suficiente para seguir tu
camino.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('¡Escóndete de los posibles vampiros de
la noche!');
    END IF;
    DBMS_OUTPUT.PUT_LINE('----------------------------------------
----------------------------------------');
  END farolaMasCercana;
```

Explicación de método: He decidido llamar al método farolaMasCercana, en este caso, tiene de parámetro de entrada el nombre de la persona que quiero tomar de referencia para encontrar la farola más cercana a ella.

En primer lugar creo un cursor llamado `cursor_farola` en el que selecciono todas las tuplas insertadas en la geometría Cáceres que sean de tipo Luz (serían todas las farolas).

```
    -- Cursor para recuperar las farolas
    CURSOR cursor_farola IS
      SELECT *
      FROM caceres
      WHERE tipo = 'Luz';
```

Ahora imprimiría por pantalla con la función `DBMS_OUTPUT.PUT_LINE` una serie de guiones para separar este método de los demás en la consola de la ejecución cuando se ejecute y verlo mejor.

Creo una variable auxiliar distMin de tipo numérico y le asigno un valor inicial de 9999 (es un valor muy alto para evitar problemas a la hora de comparaciones).

Se almacena la geometría de la persona que entramos el nombre por parámetro (pnombre) en la variable auxiliar y local llamada geomEntrada de tipo `SDO_GEOMETRY`.

```
    SELECT Geom INTO geomEntrada
    FROM Extremadura
    WHERE Nombre = pnombre;
```

Ahora almacenamos la dimensión de la geometría Cáceres sobre las que se encuentran todas las personas y farolas en una variable local y auxiliar llamada “dim”.

```
    SELECT DIMINFO INTO dim
    FROM USER_SDO_GEOM_METADATA
    WHERE table_name='CÁCERES';
```

Seguidamente habría que abrir el cursor e iniciar el bucle para recorrer todas las tuplas que hubiese en el cursor. Esas tuplas se almacenaran en una variable llamada tupla con la que podremos hacer las comparaciones necesarias para saber la distancia entre cada tupla de farola y la persona que se ha introducido por parámetros de entrada. Habrá tantas iteraciones en el bucle como tuplas haya en el cursor.

```
    -- Recorrer todos las farolas
    OPEN cursor_farola;
    LOOP
      FETCH cursor_farola INTO tupla;
      EXIT WHEN cursor_farola%NOTFOUND;
      dist := SDO_GEOM.SDO_DISTANCE(geomEntrada,dim,tupla.Geom,dim);
      IF (dist < distMin) and (tupla.Nombre <> pnombre) THEN
        distMin := dist;
        nomFarola := tupla.Nombre;
      END IF;
    END LOOP;
```

Para averiguar la distancia usamos la función `SDO_GEOM.SDO_DISTANCE` que tiene por parámetros las variables geomEntrada (geometría de la persona), la dimensión donde se encuentra la persona (dim), la geometría de la farola (tupla.Geom) y la dimensión donde se haya la geometría. Seguidamente se realiza una comprobación para averiguar la distancia menor hasta ahora que se irá actualizando a medida que vayan pasando las iteraciones del bucle. Almacenamos las distancia de la farola más cercana en la variable local distMin y su nombre en nomFarola.

Una vez realizadas todas las iteraciones, se utiliza la función `DBMS_OUTPUT.PUT_LINE` para indicar la distancia de la farola más cercana y el nombre de la misma.

```
DBMS_OUTPUT.PUT_LINE('La farola m□cercana a '||pnombre||' es
'||nomFarola||', a una distancia de '||distMin);
```

Ahora hago una comprobación con la distancia obtenida y muestro los siguientes mensajes en función de la distancia obtenida:

- Si la distancia es menor que 5 → Tienes luz suficiente para seguir tu camino.
- Si la distancia es mayor que 5 → ¡Escóndete de los posibles vampiros de la noche!

Código de la comprobación:

```
    IF distMin<5.0 THEN
        DBMS_OUTPUT.PUT_LINE('Tienes luz suficiente para seguir tu
camino.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Esc□te de los posibles vampiros de la
noche!');
    END IF;
```

Se muestra por pantalla una serie de guiones para indicar que se acaba el método y separarlo del siguiente. Se finaliza el método con `END farolaMasCercana;`

Esta sería la llamada usando este método:

```
  farolaMasCercana('Andrés);
  farolaMasCercana('Eva');
  farolaMasCercana('Lucía');
```

Y este el resultado:

```
--------------------------------------------------------------------------
--------------
La farola más cercana a Andrés es Farola3, a una distancia de 0
Tienes luz suficiente para seguir tu camino.
--------------------------------------------------------------------------
--------------
--------------------------------------------------------------------------
--------------
La farola más cercana a Eva es Farola2, a una distancia de
2,23606797749979
Tienes luz suficiente para seguir tu camino.
--------------------------------------------------------------------------
--------------
--------------------------------------------------------------------------
--------------
La farola más cercana a Lucía es Farola1, a una distancia de
2,82842712474619
Tienes luz suficiente para seguir tu camino.
--------------------------------------------------------------------------
--------------
```

Procedimiento farolasPorBarrio

Este método muestra el número de farolas que hay en un barrio específico de Cáceres que se introduce por parámetro de entrada. Recordar que se ha creado una geometría principal llamada Cáceres y sobre ella se insertan los elementos de tipo persona, barrios y farolas. Este sería el método:

```
  PROCEDURE farolasPorBarrio(pnombre IN VARCHAR2) IS
   geomEntrada SDO_GEOMETRY;
   tupla caceres%ROWTYPE;
   dim SDO_DIM_ARRAY;
   farolas NUMBER;
   -- Cursor para recuperar las farolas
   CURSOR cursor_farola IS
     SELECT *
     FROM caceres
     WHERE tipo = 'Luz';

 BEGIN
   farolas:=0;
   -- Recuperar la geometria del parametro de entrada
   SELECT Geom INTO geomEntrada
   FROM Caceres
   WHERE Nombre = pnombre AND tipo = 'Barrio';

   -- Obtener dim
   SELECT DIMINFO INTO dim
   FROM USER_SDO_GEOM_METADATA
   WHERE table_name='CACERES'; --Ojo, CACERES en maysculas

   -- Recorrer todos los barrios
   OPEN cursor_farola;
   LOOP
```

```
      FETCH cursor_farola INTO tupla;
      EXIT WHEN cursor_farola%NOTFOUND;
      --si hay interaccion entre ambas
      IF
(SDO_GEOM.RELATE(geomEntrada,dim,'anyinteract',tupla.Geom,dim)='TRUE')
THEN
        farolas:= farolas + 1;
      END IF;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('El nmero de farolas en el '||pnombre||' es
de '||farolas||'.');
  END farolasPorBarrio;
```

Explicación de método: He decidido llamar al método farolasPorBarrio, en este caso, tiene de parámetro de entrada el nombre del barrio que quiero tomar de referencia para encontrar el número de farolas que contiene.

En primer lugar creo un cursor llamado `cursor_farola` en el que selecciono todas las tuplas insertadas en la geometría Cáceres que sean de tipo Luz (serían todas las farolas).

```
    -- Cursor para recuperar las farolas
    CURSOR cursor_farola IS
      SELECT *
      FROM caceres
      WHERE tipo = 'Luz';
```

Creo una variable local para contar el número de farolas que hay en el barrio y la inicializo en 0.

```
farolas:=0;
```

Se almacena la geometría del barrio que entramos el nombre por parámetro (pnombre) en la variable auxiliar y local llamada geomEntrada de tipo `SDO_GEOMETRY`.

```
    SELECT Geom INTO geomEntrada
    FROM Extremadura
    WHERE Nombre = pnombre AND tipo = 'Barrio';
```

Ahora almacenamos la dimensión de la geometría Cáceres sobre las que se encuentran todas las personas y farolas en una variable local y auxiliar llamada “dim”.

```
    SELECT DIMINFO INTO dim
    FROM USER_SDO_GEOM_METADATA
    WHERE table_name='CÁCERES';
```

A continuación, abro el cursor y hago tantas iteraciones como tuplas haya en él. Cada valor de las tuplas del cursor los almaceno en la variable local y auxiliar **tupla** y en cada iteración compruebo con la función `SDO_GEOM.RELATE` si la geometría del barrio (geomEntrada) con su dimensión (dim) interacciona con la geometría de la farola (tupla.Geom) y su dimensión.

Se usa la palabra clave ANYINTERACT que devuelve **VERDADERO** si las dos geometrías no están separadas. En caso de que no estén separadas aumento en uno el valor de farolas en el barrio.

```
    -- Recorrer todos los barrios
    OPEN cursor_farola;
    LOOP
      FETCH cursor_farola INTO tupla;
      EXIT WHEN cursor_farola%NOTFOUND;
      --si hay interaccion entre ambas
      IF
(SDO_GEOM.RELATE(geomEntrada,dim,'anyinteract',tupla.Geom,dim)='TRUE')
THEN
        farolas:= farolas + 1;
      END IF;
    END LOOP;
```

Una vez finalizado el bucle muestro un mensaje con el número de farolas en el barrio.

```
DBMS_OUTPUT.PUT_LINE('El nmero de farolas en el '||pnombre||' es de
'||farolas||'.');
```

Se muestra por pantalla una serie de guiones para indicar que se acaba el método y separarlo del siguiente. Se finaliza el método con `END farolasPorBarrio;`

Esta sería la llamada usando este método:

```
  farolasPorBarrio('Parque del Príncipe');
  farolasPorBarrio('Cánovas');
  farolasPorBarrio('Parte Antigua');
  farolasPorBarrio('Parque El Rodeo');
```

Y este el resultado:

```
El número de farolas en el Parque del Príncipe es de 1.
El número de farolas en el Cánovas es de 0.
El número de farolas en el Parte Antigua es de 1.
El número de farolas en el Parque El Rodeo es de 1.
```

SALIDA DE SQL

Salida del scrip Espacial_1.sql

```
Table CACERES creado.

Sequence NUMGEOM creado.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

El amigo más cercano a Andrés es Eva, a una distancia de
6,40312423743285
El amigo más cercano a Eva es Marta, a una distancia de
2,82842712474619
El amigo más cercano a Lucía es Eva, a una distancia de
3,60555127546399
El barrio más cercano a Andrés es Parte Antigua, a una distancia de 0
```

```
El barrio más cercano a Eva es Cánovas, a una distancia de 0
El barrio más cercano a Lucía es Parque del Príncipe, a una distancia
de 1
--------------------------------------------------------------------
--------------
La farola más cercana a Andrés es Farola3, a una distancia de 0
Tienes luz suficiente para seguir tu camino.
--------------------------------------------------------------------
--------------
--------------------------------------------------------------------
--------------
La farola más cercana a Eva es Farola2, a una distancia de
2,23606797749979
Tienes luz suficiente para seguir tu camino.
--------------------------------------------------------------------
--------------
--------------------------------------------------------------------
--------------
La farola más cercana a Lucía es Farola1, a una distancia de
2,82842712474619
Tienes luz suficiente para seguir tu camino.
--------------------------------------------------------------------
--------------
El amigo más cercano a Andrés es Eva, a una distancia de
1,4142135623731
El barrio más cercano a Andrés es Cánovas, a una distancia de 0
Andrés NO está en Parque del Príncipe
Andrés está en Cánovas
Andrés NO está en Parte Antigua
Andrés NO está en Parque El Rodeo
Marta NO está en Parque del Príncipe
Marta NO está en Cánovas
Marta NO está en Parte Antigua
Marta está en Parque El Rodeo
El número de farolas en el Parque del Príncipe es de 1.
El número de farolas en el Cánovas es de 0.
El número de farolas en el Parte Antigua es de 1.
El número de farolas en el Parque El Rodeo es de 1.

Procedimiento PL/SQL terminado correctamente.

Table CACERES borrado.


Sequence NUMGEOM borrado.


1 fila eliminado
```

Salida del scrip Espacial_2.sql

```
Table EXTREMADURA creado.

Sequence NUMGEOM creado.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.
```

```
1 fila insertadas.


1 fila insertadas.


1 fila insertadas.


1 fila insertadas.


1 fila insertadas.


1 fila insertadas.


1 fila insertadas.


1 fila insertadas.


1 fila insertadas.

La autovía A66 pasa por Cáceres
La autovía A66 pasa por Mérida
La autovía A66 pasa por Plasencia
La autovía A66 pasa por Zafra
--------------------------------------------------------------------------
--------------
La autovía A5 pasa por Miajadas
La autovía A5 pasa por Mérida
La autovía A5 pasa por Navalmoral
La autovía A5 pasa por Trujillo
--------------------------------------------------------------------------
--------------
Las localidades contenidas en un radio 3 a la localidad Badajoz son:
Cáceres con distancia: 2,23606797749979
Mérida con distancia: 2,82842712474619
ValenciaAlcántara con distancia: 1,4142135623731
--------------------------------------------------------------------------
--------------
Las localidades contenidas en un radio 5 a la localidad Badajoz son:
Alcántara con distancia: 3
Coria con distancia: 3,80788655293195
Cáceres con distancia: 2,23606797749979
DonBenito con distancia: 4,47213595499958
Miajadas con distancia: 4,12310562561766
Moraleja con distancia: 4,12310562561766
Mérida con distancia: 2,82842712474619
Olivenza con distancia: 3,16227766016838
Trujillo con distancia: 4,12310562561766
ValenciaAlcántara con distancia: 1,4142135623731
Villanueva con distancia: 4,92442890089805
Zafra con distancia: 4,47213595499958
--------------------------------------------------------------------------
--------------
```

```
Las localidades contenidas en un radio 3 a la localidad Cáceres son:
Alcántara con distancia: 2,82842712474619
Badajoz con distancia: 2,23606797749979
Coria con distancia: 2,54950975679639
Miajadas con distancia: 2,82842712474619
Trujillo con distancia: 2
--------------------------------------------------------------------------
--------------
Las localidades contenidas en un radio 5 a la localidad Cáceres son:
Alcántara con distancia: 2,82842712474619
Badajoz con distancia: 2,23606797749979
Coria con distancia: 2,54950975679639
DonBenito con distancia: 3,60555127546399
Miajadas con distancia: 2,82842712474619
Moraleja con distancia: 3,16227766016838
Mérida con distancia: 3
Navalmoral con distancia: 4,24264068711928
Plasencia con distancia: 3,16227766016838
Trujillo con distancia: 2
ValenciaAlcántara con distancia: 3
Villanueva con distancia: 3,90512483795333
--------------------------------------------------------------------------
--------------
El embalse más cercano a la localidad de Cáceres es EAlcántara con
distancia 1,56524758424985
El embalse más cercano a la localidad de Badajoz es EAlcántara con
distancia 2,06155281280883
El embalse más cercano a la localidad de Mérida es EOrellana con
distancia 4
El embalse más grande es EOrellana con area 2
--------------------------------------------------------------------------
--------------
Procedimiento 1
El perimetro del embalse   EAlcántara es de 5,23606797749979
El perimetro del embalse   EOrellana es de 6,47213595499958
El perimetro del embalse   ESerena es de 7,08276253029822
--------------------------------------------------------------------------
--------------
Procedimiento 2
La distancia entre la localidad Alcántara y el embalse EOrellana es de
8,00347146902864
--------------------------------------------------------------------------
--------------
Procedimiento 3
Existe interseccion entre Zona1 y EAlcántara.
--------------------------------------------------------------------------
--------------

Procedimiento PL/SQL terminado correctamente.

Table EXTREMADURA borrado.


Sequence NUMGEOM borrado.


1 fila eliminado
```