IES Gonzalo Nazareno

Autores: Francisco Huzón
Jesús García
Ángel Almazán
Fernando Tirado
Profesor: Raúl Ruiz
Módulo: Administración de base de datos
Título: Repaso PL/SQL

Índice de contenido

1. RESPONSABILIDAD INDIVIDUAL FRÁN HUZÓN (CAPITÁN)

Ejercicio 4 Departamento Universitario

Realiza los módulos de programación necesarios para que cuando un proyecto de investigación reciba una nueva subvención con un importe mayor a 3000 euros se envíe un correo electrónico al profesor responsable del proyecto informándole del organismo que la ha concedido y el importe exacto de la misma.

Instalamos un complemento necesario para esta tarea:

```
sudo apt install postgresql-plpython3-11
```

Se crea la extensión para que plpgsql puede ejecutar python

```
create extension plpython3u;
```

Función que enviará el correo usando métodos de la extensión creada.

Se necesita activar el acceso de aplicaciones poco seguras de la cuenta a usar.

```
CREATE OR REPLACE FUNCTION py_pgmail(from_addr text, to_addr_list text[],
subject text, login text, password text, message_text text default '',
smtpserver text default 'smtp.gmail.com:587')
    RETURNS boolean
AS $$
    import smtplib
    from email.mime.multipart import MIMEMultipart
    from email.mime.text import MIMEText
    msg = MIMEMultipart('alternative')
    msg["Subject"] = subject
    msg['From'] = from_addr
    msg['To'] = ', '.join(to_addr_list)
    if message_text.replace(' ', '')!='':
        part1 = MIMEText(message_text, 'plain')
        msg.attach(part1)
    server = smtplib.SMTP(smtpserver)
    server.starttls()
    server.login(login, password)
    problems = server.sendmail(from_addr, to_addr_list,msg.as_string())
    server.quit()
    if len(problems)>0:
        plpy.info('Ha ocurrido un error: '+str(problems))
        return False
    else:
        return True
$$ LANGUAGE plpython3u;
```

Trigger que llama a la función subvencion3mil().

```
create trigger muchodinero
after insert on Subvenciones
for each row execute procedure subvencion3mil();
```

Función que consigue los datos pedidos y ejecuta el envío del correo.

```
create or replace function subvencion3mil()
returns trigger as $$
declare
v_correo VARCHAR(50);
v_organismo VARCHAR(50);
v_importe NUMERIC;
begin
    if new.importe > 3000 then
        select per.correo_electronico,sub.nombreorganismo,sub.importe into
v_correo,v_organismo,v_importe
        from personal per,subvenciones sub, profesores pro,
proyectos_de_investigacion proy
        where per.dni = pro.dni
        and pro.dni = proy.dniprofesordirector
        and proy.codigoproyecto = sub.codigoproyecto
        and sub.nombreorganismo = new.nombreorganismo;

        perform py_pgmail(
            'huzonvillarfranciscojesus@gmail.com',
            array[v_correo],
            array['huzonvillarfranciscojesus@gmail.com'],
            array['huzonvillarfranciscojesus@gmail.com'],
            'Nueva subvención',
            v_correo,
            '19julio1992',
            'Organismo: '||v_organismo||' '||'Importe: '||v_importe||'€',
            '',
            'smtp.gmail.com:587');
        return null;
    end if;
end;
$$ LANGUAGE plpgsql;
```

Ejercicio 5 Departamento Universitario

Añade un campo ImporteSubvencionado en la tabla Proyectos de Investigación, realiza las operaciones necesarias para rellenarlo y realiza los módulos de programación necesarios para mantener dicha columna actualizada de forma automática.

```
alter table proyectos_de_investigacion add ImporteSubvencionado NUMBER(7,2);

create or replace procedure totalsubvenciones
as
  cursor c_totalsubvenciones is
  select codigoproyecto,sum(importe) as importetotal
  from subvenciones
  group by codigoproyecto;
```

```
begin
    for i in c_totalsubvenciones LOOP
     update proyectos_de_investigacion
     set ImporteSubvencionado = i.importetotal
     where codigoproyecto = i.codigoproyecto;
    end loop;
end;
/

create or replace trigger actualizarimportestotales
after insert or delete or update of importe on subvenciones
for each row
begin
     if inserting then
            update proyectos_de_investigacion
            set ImporteSubvencionado = ImporteSubvencionado+:new.importe
            where codigoproyecto = :new.codigoproyecto;
     elsif updating then
            update proyectos_de_investigacion
            set ImporteSubvencionado = ImporteSubvencionado+
(:new.importe-:old.importe)
            where codigoproyecto = :old.codigoproyecto;
     elsif deleting then
            update proyectos_de_investigacion
            set ImporteSubvencionado = ImporteSubvencionado - :old.importe
            where codigoproyecto = :old.codigoproyecto;
     end if;
end;
/
```

Ejercicio 6 Departamento Universitario

Realiza los módulos de programación necesarios para que un alumno no pueda matricularse de una asignatura que haya aprobado durante un curso académico anterior.

```
create or replace trigger nomatricular
before insert or update on matriculaciones
for each row
declare
    cursor c_matriculas is
    select codigoasignatura,dnialumno,nota
    from convocatorias;
begin
    for i in c_matriculas loop
        if ((:new.dnialumno=i.dnialumno) and
(:new.codigoasignatura=i.codigoasignatura)) and (i.nota >=5) then
            raise_application_error(-20001,'Este alumno ya ha aprobado esa
asignatura. No puede matricularse.');
        end if;
    end loop;
end;
/
```

2. RESPONSABILIDAD INDIVIDUAL JESÚS GARCÍA (PORTAVOZ)

Ejercicio 4 Inventario Informático (Postgres)

Realizar un trigger para que en el momento en que se produzca una incidencia se envíe un correo electrónico con toda la información al usuario responsable de la misma. Si el ordenador es uno de los servidores, el correo se enviará a todos los usuarios.

```
-- Parámetro: Número de serie de un ordenador.
-- Descripción: Comprueba si el número de serie pertenece a un servidor o no,
si es un servidor devuelve 1, si no, 0.
CREATE OR REPLACE FUNCTION ComprobarSiServidor (p_numeroserie
Servidores.NumeroSerie%type)
RETURNS boolean
AS $ComprobarSiServidor$
DECLARE
  c_servidores CURSOR FOR
    SELECT NumeroSerie
    FROM Servidores;

  elem RECORD;

BEGIN
  FOR elem IN c_servidores LOOP
    IF p_numeroserie = elem.NumeroSerie THEN
      RETURN True;
    END IF;
  END LOOP;

  RETURN False;

END;
$ComprobarSiServidor$ LANGUAGE plpgsql;




-- Parámetro: Nombre de usuario.
-- Descripción: Devuelve la dirección de correo del usuario indicado.
CREATE OR REPLACE FUNCTION ObtenerCorreo (p_usuario Usuarios.Nombre%type)
RETURNS VARCHAR
AS $ObtenerCorreo$
DECLARE
  v_correo VARCHAR(30);

BEGIN
  SELECT CorreoElectronico INTO v_correo
  FROM Usuarios
  WHERE Nombre = p_usuario;

  RETURN v_correo;

END;
$ObtenerCorreo$ LANGUAGE plpgsql;
```

```
-- Descripción: Función que envía el mensaje.
CREATE OR REPLACE FUNCTION py_pgmail(from_addr text, to_addr text, subject
text, login text, password text, message_text text default '', smtpserver text
default 'smtp.gmail.com:587')
    RETURNS boolean
AS $$
    import smtplib
    from email.mime.multipart import MIMEMultipart
    from email.mime.text import MIMEText
    msg = MIMEMultipart('alternative')
    msg["Subject"] = subject
    msg['From'] = from_addr
    msg['To'] = to_addr
    if message_text.replace(' ', '')!='':
        part1 = MIMEText(message_text, 'plain')
        msg.attach(part1)
    server = smtplib.SMTP(smtpserver)
    server.starttls()
    server.login(login, password)
    problems = server.sendmail(from_addr, to_addr,msg.as_string())
    server.quit()
    if len(problems)>0:
        plpy.info('Ha ocurrido un error: '+str(problems))
        return False
    else:
        return True
$$ LANGUAGE plpython3u;


-- Descripción: Tras insertar una nueva incidencia, comprueba si el ordenador
afectado es un servidor,
-- si lo és, envía un correo a todos los usuarios, si no solo al creador de la
incidencia.
CREATE OR REPLACE FUNCTION CorreoIncidencia()
RETURNS TRIGGER AS $TriggerCorreoIncidencia$
DECLARE
  c_usuarios CURSOR FOR
    SELECT nombre
    FROM usuarios;

BEGIN
  IF ComprobarSiServidor(NEW.NumeroSerieOrdenador) THEN
    FOR elem IN c_usuarios LOOP
      PERFORM py_pgmail('incidencias@iesgn.org', -- Cuenta de correo que envía
el mensaje.
                        ObtenerCorreo(elem.nombre), -- Dirección a la que se
envía.
                        'Incidencia', -- Asunto del mensaje.
                        'incidencias@iesgn.org', -- Cuenta de correo que envía
el mensaje.
                        'password', -- Contraseña de la cuenta que envía el
mensaje.
                        NEW.Descripcion);
    END LOOP;
  ELSE
    PERFORM py_pgmail('incidencias@iesgn.org', -- Cuenta de correo que envía el
mensaje.
                      ObtenerCorreo(NEW.NombreUsuarioResponsable), -- Dirección
a la que se envía.
                      'Incidencia', -- Asunto del mensaje.
                      'incidencias@iesgn.org', -- Cuenta de correo que envía el
mensaje.
                      'password', -- Contraseña de la cuenta que envía el
mensaje.
```

```
                      NEW.Descripcion);
  END IF;

  RETURN NULL;

END;
$TriggerCorreoIncidencia$ LANGUAGE plpgsql;



CREATE TRIGGER TriggerCorreoIncidencia
AFTER INSERT ON Incidencias
FOR EACH ROW
EXECUTE PROCEDURE CorreoIncidencia();
```

Ejercicio 5 Inventario Informático

Añade una columna TiempodeDesconexión en la tabla Servidores. Haz un procedimiento que la rellene a partir de los datos que aparecen en la tabla PeriodosdeApagado y realiza un trigger que mantenga la columna actualizada cada vez que termine un periodo de apagado.

```
-- Añadir la columna "TiempodeDesconexion" a la tabla "Servidores".
ALTER TABLE Servidores ADD TiempodeDesconexion NUMBER(15);

-- Parámetros:
--    Fecha inicio.
--    Fecha fin.
-- Descripción: Devuelve el tiempo transcurrido en segundos entre las dos
fechas.
CREATE OR REPLACE FUNCTION CalcularDiferencia (p_inicio DATE,
                                               p_fin DATE)
RETURN NUMBER
AS
  v_diferencia NUMBER := 0;

BEGIN
  v_diferencia := (p_fin - p_inicio) * 86400;
  RETURN v_diferencia;

END CalcularDiferencia;
/

-- Descripción: Rellena la columna "TiempodeDesconexion" a través de los datos
de la tabla "Periodosdeapagado".
CREATE OR REPLACE PROCEDURE RellenarTiempoDesconexion
AS
  CURSOR c_periodosapagado IS
    SELECT NumeroSerieServidor, sum(CalcularDiferencia(FechaHoraInicio,
FechaHoraFin)) AS tiempodesconexion
    FROM Periodosdeapagado
    GROUP BY NumeroSerieServidor;

BEGIN
  FOR elem IN c_periodosapagado LOOP
    UPDATE Servidores
      SET TiempodeDesconexion = elem.tiempodesconexion
      WHERE NumeroSerie = elem.NumeroSerieServidor;
  END LOOP;

END RellenarTiempoDesconexion;
/
```

```
-- Descripción: Llama al procedimiento "RellenarTiempoDesconexion" cuando se
inserta o actualiza el campo "FechaHoraFin".
CREATE OR REPLACE TRIGGER ActualizarTablaServidores
AFTER INSERT OR UPDATE OF FechaHoraFin ON Periodosdeapagado
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    UPDATE Servidores
      SET TiempodeDesconexion = TiempodeDesconexion +
CalcularDiferencia(:new.FechaHoraInicio, :new.FechaHoraFin)
      WHERE NumeroSerie = :new.NumeroSerieServidor;
  ELSIF UPDATING THEN
    UPDATE Servidores
      SET TiempodeDesconexion = TiempodeDesconexion +
CalcularDiferencia(:new.FechaHoraInicio, :new.FechaHoraFin)
      WHERE NumeroSerie = :old.NumeroSerieServidor;
  END IF;
END;
/
```

Ejercicio 7 Inventario Informático

Realiza los módulos de programación necesarios para garantizar que un servidor siempre proporciona al menos un servicio.

```
-- Creación de un paquete donde almacenar los servicios de cada servidor.
CREATE OR REPLACE PACKAGE Pkg_Servicios
AS
  TYPE reg_servicios IS RECORD
  (
    numero_serie_servidor servidores.NumeroSerie%type,
    numero_servicios      NUMBER
  );

  TYPE tab_servicios IS TABLE OF reg_servicios
  INDEX BY VARCHAR2(15);

  v_servicios tab_servicios;

END;
/

-- Descripción: Rellena el paquete Pkg_Servicios indicando el número de
servicios que tiene cada servidor.
CREATE OR REPLACE TRIGGER RellenarServicios
BEFORE INSERT OR UPDATE OR DELETE ON ServiciosInstalados
DECLARE
  CURSOR c_serviciosinstalados IS
    SELECT NumeroSerieServidor, count(NombreServicio) AS NumeroServicios
    FROM ServiciosInstalados
    WHERE FechaFin IS NULL
    GROUP BY NumeroSerieServidor;

BEGIN
  FOR elem IN c_serviciosinstalados LOOP

Pkg_Servicios.v_servicios(elem.NumeroSerieServidor).numero_serie_servidor :=
elem.NumeroSerieServidor;
    Pkg_Servicios.v_servicios(elem.NumeroSerieServidor).numero_servicios :=
elem.NumeroServicios;
  END LOOP;

END;
/
```

```
-- Descripción: Se asegura de que el número de servicios del servidor afectado
siempre sea mayor que 1.
CREATE OR REPLACE TRIGGER ComprobarNumeroServicios
BEFORE INSERT OR UPDATE OR DELETE ON ServiciosInstalados
FOR EACH ROW
BEGIN
-- Si FechaFin no es null, significa que el servicio está desinstalado por lo
que el servidor no ofrecería ningún servicio.
  IF INSERTING AND :new.FechaFin IS NOT NULL THEN
    RAISE_APPLICATION_ERROR(-20001, 'Con la carga de datos indicada el servidor
'||:new.NumeroSerieServidor||' no ofrece ningún servicio.');
  END IF;


-- Comprueba si FechaFin es null, indicando así que el servicio está en
funcionamiento, después de eso, levanta una excepción si es el único servicio
del servidor o
-- resta 1 al número de servicios si tiene varios.
  IF DELETING AND :old.FechaFin IS NULL THEN
    IF Pkg_Servicios.v_servicios(:old.NumeroSerieServidor).numero_servicios = 1
THEN
      RAISE_APPLICATION_ERROR(-20002, 'Todos los servidores deben tener al
menos 1 servicio, al ejecutar la instrucción indicada el servidor
'||:old.NumeroSerieServidor||' no cumpliría esta regla.');
    ELSE
      Pkg_Servicios.v_servicios(:old.NumeroSerieServidor).numero_servicios :=
Pkg_Servicios.v_servicios(:old.NumeroSerieServidor).numero_servicios - 1;
    END IF;
  END IF;

-- Comprueba si el número de serie del servidor antiguo es distinto al nuevo,
indicando así que el servicio se ha cambiado de servidor, levanta una excepción
si es el único servicio del
-- servidor antiguo, si tiene varios, resta 1 al número de servicios del
servidor antiguo y suma 1 al número de servicios del servidor nuevo.
  IF UPDATING AND :old.NumeroSerieServidor != :new.NumeroSerieServidor THEN
    IF Pkg_Servicios.v_servicios(:old.NumeroSerieServidor).numero_servicios = 1
THEN
      RAISE_APPLICATION_ERROR(-20002, 'Todos los servidores deben tener al
menos 1 servicio, al ejecutar la instrucción indicada el servidor
'||:old.NumeroSerieServidor||' no cumpliría esta regla.');
    ELSE
      Pkg_Servicios.v_servicios(:old.NumeroSerieServidor).numero_servicios :=
Pkg_Servicios.v_servicios(:old.NumeroSerieServidor).numero_servicios - 1;
      Pkg_Servicios.v_servicios(:new.NumeroSerieServidor).numero_servicios :=
Pkg_Servicios.v_servicios(:new.NumeroSerieServidor).numero_servicios + 1;
    END IF;

-- Comprueba si FechaFin no es NULL, indicando así que el servicio se ha
desinstalado, levanta una excepción si es el único servicio del servidor o
resta 1 al
-- número de servicios si tiene varios.
  ELSIF UPDATING AND :new.FechaFin IS NOT NULL THEN
    IF Pkg_Servicios.v_servicios(:new.NumeroSerieServidor).numero_servicios = 1
THEN
      RAISE_APPLICATION_ERROR(-20002, 'Todos los servidores deben tener al
menos 1 servicio, al ejecutar la instrucción indicada el servidor
'||:new.NumeroSerieServidor||' no cumpliría esta regla.');
    ELSE
      Pkg_Servicios.v_servicios(:new.NumeroSerieServidor).numero_servicios :=
Pkg_Servicios.v_servicios(:new.NumeroSerieServidor).numero_servicios - 1;
    END IF;
  END IF;

END;
/
```

3. RESPONSABILIDAD INDIVIDUAL ÁNGEL ALMAZÁN (FACILITADOR)

Ejercicio 2 Campeonato de Ajedrez (Postgres)

Realiza un procedimiento que realice informes sobre la clasificación del campeonato gestionando las excepciones oportunas, el primer parámetro recibido será el tipo de informe.

Informe Tipo 1: El segundo parámetro será un código de partida y el tercer parámetro se ignorará.

Informe Tipo 2: El segundo parámetro será un código de jugador, el tercer parámetro será un nombre de ronda. Se mostrarán todas las partidas correspondientes a esa ronda en las que haya participado el jugador especificado, incluyendo oponente, color de piezas y resultado final de la partida. Al final, se añadirá una línea con la puntuación total del jugador en esa ronda.

Informe Tipo 3: El segundo parámetro será un código de país y el tercero será ignorado. Se mostrará un informe con los resultados de todos los jugadores de ese país en todas las rondas.

Ejercicio 2 Departamento Universitario (Postgres)

Realiza un procedimiento llamado InformesCalificaciones que reciba tres parámetros. El primero será el tipo de informe deseado, el segundo y el tercero dependerán del tipo de informe. En todos los casos debes contemplar las excepciones que consideres necesarias.

Informe Tipo 1: El segundo parámetro será un curso académico. El tercer parámetro será el nombre de una asignatura. El listado mostrará la nota más alta conseguida por cada alumno que haya cursado la asignatura durante ese curso académico junto con el nombre del profesor que le impartió la asignatura y la fecha de la convocatoria en la que logró dicha nota. Finalmente se mostrará la nota media de la asignatura con tres decimales.

Informe Tipo 2: El segundo parámetro será un curso académico. El tercer parámetro será el nombre de un profesor. El listado mostrará para cada asignatura impartida por el citado profesor un listado de los alumnos que la han cursado durante el curso académico recibido como parámetro, junto con la nota más alta que consiguieron. Al final de cada asignatura mostrará la nota media en la asignatura.

Informe Tipo 3: El segundo parámetro valdrá NULL. El tercer parámetro será el nombre de una asignatura. Se mostrará información de las calificaciones en dicha asignatura a lo largo de los distintos cursos académicos, ordenados cronológicamente.

Ejercicio 8 Campeonato de Ajedrez

Realiza un trigger que no permita insertar una partida si uno de los dos participantes ha alcanzado ya los doce puntos en la ronda correspondiente a la misma.

4. RESPONSABILIDAD INDIVIDUAL FERNANDO TIRADO (SECRETARIO)

Ejercicio 4 Investigadores (Postgres)

Añade una columna “Dirección e-mail” a la tabla Profesores y rellénala con datos consistentes. A continuación realiza un trigger que cada vez que se inserte una nueva valoración envíe un correo electrónico a todos los profesores que están trabajando ahora mismo en ese proyecto con la siguiente información: Nombre y Apellidos del Evaluador, Fecha de la valoración, proyecto, objetivo y criterio valorado y valoración realizada.

Instalamos un complemento necesario para esta tarea:

```
sudo apt install postgresql-plpython3-11
```

Para hacer esto, también es necesario habilitar que tu cuenta de google pueda recibir peticiones. Para esto debemos de habilitarlo en el siguiente link:

https://support.google.com/accounts/answer/6010255?hl=en

Ahora creamos la extensión en postgres:

```
investigadores=# create extension plpython3u;
```

Creamos la función:

```
CREATE OR REPLACE FUNCTION py_pgmail(from_addr text, to_addr_list text[],
subject text, login text, password text, message_text text default '',
smtpserver text default 'smtp.gmail.com:587')

     RETURNS boolean
AS $$
     import smtplib
     from email.mime.multipart import MIMEMultipart
     from email.mime.text import MIMEText
     msg = MIMEMultipart('alternative')
     msg["Subject"] = subject
     msg['From'] = from_addr
     msg['To'] = ', '.join(to_addr_list)
     if message_text.replace(' ', '')!='':
            part1 = MIMEText(message_text, 'plain')
            msg.attach(part1)
     server = smtplib.SMTP(smtpserver)
     server.starttls()
     server.login(login, password)
     problems = server.sendmail(from_addr, to_addr_list,msg.as_string())
     server.quit()
     if len(problems)>0:
            plpy.info('An error ocurred: '+str(problems))
            return False
     else:
            return True

$$ LANGUAGE plpython3u;
```

Realizamos el ejercicio

```
create or replace function MandarCorreo()
returns trigger as $$
declare

     c_correo cursor for
     select pr.direccion_mail as mailprofesor, e.nombre as nombrevaluador,
e.apellido as apellidoevaluador, v.fecha as fechavaloracion, p.nombre as
nombreproyecto, o.nombre as nombreobjetivo, c.descripcion as
descripcioncriterio, v.valoracion as valoracion
     from evaluadores e, valoraciones v, proyectos_de_investigacion p,
objetivos o, criterios c, profesores pr, colaboraciones co
     where e.codigo = v.codevaluador
     and e.codigo = NEW.codevaluador
     and p.codigo = o.codigoproyecto
     and o.codigo = v.codobjetivo
     and o.codigo = NEW.codobjetivo
     and c.codigo = v.codcriterio
     and c.codigo = NEW.codcriterio
     and pr.dni = co.dnicolaborador
     and co.codigoproyecto = p.codigo;

begin

     for v_correo in c_correo loop
            perform py_pgmail(
     'fernando.tb.95@gmail.com',
     array[v_correo.mailprofesor],
     'Información valoración',
     'pruebaftirado@gmail.com',
     'fernando-24',
     'Nombre y apellidos evaluador/a: '||v_correo.nombrevaluador||' '||
v_correo.apellidoevaluador||chr(10)||
     'Fecha de la valoración: '||v_correo.fechavaloracion||chr(10)||
     'Nombre proyecto: '||v_correo.nombreproyecto||chr(10)||
     'Nombre objetivo: '||v_correo.nombreobjetivo||chr(10)||
     'Criterio valorado: '||v_correo.descripcioncriterio||chr(10)||
     'Valoración: '||v_correo.valoracion,
     'smtp.gmail.com:587');
     end loop;

     return null;
end;
$$ LANGUAGE plpgsql;


create trigger TriggerCorreo
after insert on valoraciones
for each row
execute procedure MandarCorreo();
```

Ejercicio 5 Investigadores

Añade una columna DineroGestionado a la tabla Doctores. Rellena dicha columna con la suma de los presupuestos de los proyectos de los que es responsable cada doctor. Realiza los triggers necesarios para que se mantenga actualizada automáticamente.

```
alter table doctores
add dinerogestionado number(15);
```

```
create or replace procedure RellenarDinero
as
        cursor c_suma
        is
        select dni_responsable, sum(presupuesto) as dinero
        from proyectos_de_investigacion
        group by dni_responsable;
begin
        for i in c_suma loop
                update doctores
                set dinerogestionado = i.dinero
                where dni = i.dni_responsable;
        end loop;
end;
/

create or replace trigger ActualizacionDinero
after insert or update or delete of presupuesto on proyectos_de_investigacion
for each row
begin
        if inserting then
                update doctores
                set dinerogestionado = dinerogestionado+:new.presupuesto
                where dni = :new.dni_responsable;
        elsif updating then
                update doctores
                set dinerogestionado = dinerogestionado+
(:new.presupuesto-:old.presupuesto)
                where dni = :old.dni_responsable;
        elsif deleting then
                update doctores
                set dinerogestionado = dinerogestionado-:old.presupuesto
                where dni = :old.dni_responsable;
        end if;
end;
/
```

Ejercicio 8 Aerogeneradores

Realiza los módulos de programación necesarios para evitar que en una misma central se monten aerogeneradores de más de tres modelos diferentes.

```
create or replace package PkgModelos
as
        type tRegModelos IS RECORD
        (
                NombreCentral aerogeneradores.nombre_central%type,
                NombreModelo aerogeneradores.nombre_modelo%type
        );

        type tTablaModelos is table of tRegModelos
        index by binary_integer;
        v_TabMd tTablaModelos;
end;
/
```

```
create or replace trigger RellenarNumModelos
before insert or update on aerogeneradores
declare
        cursor c_model
        is
        select nombre_central, nombre_modelo
        from aerogeneradores;

        i number:=0;
begin
        for v_model in c_model loop
                PkgModelos.v_TabMd(i).NombreCentral:=v_model.nombre_central;
                PkgModelos.v_TabMd(i).NombreModelo:=v_model.nombre_modelo;
                i:=i+1;
        end loop;
end;
/

create or replace trigger ComprobarNumModelos
before insert or update on aerogeneradores
for each row
declare
        v_NumModelos number:=0;
begin
        for i in PkgModelos.v_TabMd.FIRST .. PkgModelos.v_TabMd.LAST loop
                if PkgModelos.v_TabMd(i).NombreCentral = :new.nombre_central
AND PkgModelos.v_TabMd(i).NombreModelo != :new.nombre_modelo then
                        v_NumModelos:=v_NumModelos+1;
                end if;
        end loop;
        if v_NumModelos > 3 then
                raise_application_error(-20001,'Esa central no admite más
modelos diferentes de aerogenerador');
        end if;
end;
/
```

5. RESPONSABILIDAD GRUPAL

Ejercicio 2 Investigadores (Postgres)

Realiza un procedimiento que recibirá dos parámetros. El primero determinará el tipo de informe y el significado del segundo dependerá del tipo de informe.

Informe Tipo 1:

En este caso, el segundo parámetro será el nombre de un criterio y el informe mostrará las valoraciones recibidas de ese criterio.

Informe Tipo 2:

En este caso, el segundo parámetro será el nombre de un objetivo y el informe mostrará las

valoraciones recibidas de ese objetivo.

Informe Tipo 3:

En este caso, el segundo parámetro será el nombre de un proyecto y el informe mostrará las

valoraciones recibidas de ese proyecto.

```
CREATE OR REPLACE FUNCTION INFORMES(P_INF NUMERIC,
                                    P_PARAM2 VARCHAR) RETURNS VOID AS $$
DECLARE
BEGIN
CASE
    WHEN P_INF = 1 THEN
        PERFORM INFORME1(P_PARAM2);
    WHEN P_INF = 2 THEN
        PERFORM INFORME2(P_PARAM2);
    WHEN P_INF = 3 THEN
        PERFORM INFORME3(P_PARAM2);
    WHEN P_INF > 3 THEN
     RAISE EXCEPTION 'Número de informe incorrecto';
END CASE;
END;
$$ language plpgsql;



---------------------------------


INFORME 1
SELECT INFORMES(1,'DEBEN SER DE LITIO');
```

```
CREATE OR REPLACE FUNCTION INFORME1(P_2 VARCHAR) RETURNS VOID AS $$
DECLARE
    IN1 CURSOR FOR
    SELECT NOMBRE
    FROM EVALUADORES
    WHERE CODIGO IN (SELECT CODEVALUADOR
                     FROM VALORACIONES
                     WHERE CODCRITERIO IN (SELECT CODIGO
                                           FROM CRITERIOS
                                           WHERE DESCRIPCION = P_2));
    I NUMERIC:=0;

BEGIN

    RAISE NOTICE 'DESCRIPCION CRITERIO: %', P_2;
    FOR V_IN1 IN IN1 LOOP
        PERFORM INFORME12(V_IN1.NOMBRE, P_2, I);
    END LOOP;
END;
$$ language plpgsql;

CREATE OR REPLACE FUNCTION INFORME12(P_PROFE VARCHAR,
                                     P_CRITERIO VARCHAR,
                                     P_NUMVAL NUMERIC) RETURNS VOID AS $$
DECLARE
    IN12 CURSOR FOR
    SELECT VALORACION, FECHA
    FROM VALORACIONES
    WHERE CODEVALUADOR IN (SELECT CODIGO
                           FROM EVALUADORES
                           WHERE NOMBRE = P_PROFE)
    AND CODCRITERIO = (SELECT CODIGO
                       FROM CRITERIOS
                       WHERE DESCRIPCION = P_CRITERIO);

BEGIN

    RAISE NOTICE '%', P_PROFE;
    FOR V_IN12 IN IN12 LOOP
        RAISE NOTICE '% % %', CHR(9), V_IN12.FECHA, V_IN12.VALORACION;
        P_NUMVAL:=P_NUMVAL+1;
    END LOOP;
    RAISE NOTICE 'Número Total de Valoraciones: %', P_NUMVAL;
END;
$$ language plpgsql;




-------------------------


INFORME 2
SELECT INFORMES(2,'ESTUDIO GENERAL');
```

```
CREATE OR REPLACE FUNCTION INFORME2(P_OBJETIVO VARCHAR) RETURNS VOID AS $$
DECLARE
    IN2 CURSOR FOR
    SELECT C.DESCRIPCION, O.DESCRIPCION AS HOLA
    FROM CRITERIOS C, OBJETIVOS O
    WHERE C.CODIGO IN (SELECT CODCRITERIO
                    FROM VALORACIONES
                    WHERE CODOBJETIVO IN (SELECT CODIGO
                                          FROM OBJETIVOS
                                          WHERE NOMBRE = P_OBJETIVO))
    AND O.NOMBRE = P_OBJETIVO;

    V_NUMVALOBJ NUMERIC:=0;
    CONTADOR NUMERIC :=0;

BEGIN

    RAISE NOTICE 'NOMBRE OBJETIVO: %', P_OBJETIVO;
    FOR V_IN2 IN IN2 LOOP
     RAISE NOTICE '%', chr(10);
     RAISE NOTICE 'DESCRIPCIÓN OBJETIVO: %', V_IN2.HOLA;
        PERFORM INFORME21(V_IN2.DESCRIPCION,V_NUMVALOBJ);
     CONTADOR:=CONTADOR+1;
    END LOOP;
    RAISE NOTICE 'Número total de valoraciones del objetivo %: %',
P_OBJETIVO ,CONTADOR;
END;
$$ language plpgsql;

CREATE OR REPLACE FUNCTION INFORME21(P_CRITERIO VARCHAR,
                                     P_NUMVALOBJ NUMERIC) RETURNS VOID AS $$
DECLARE
    IN1 CURSOR FOR
    SELECT NOMBRE
    FROM EVALUADORES
    WHERE CODIGO IN (SELECT CODEVALUADOR
                     FROM VALORACIONES
                     WHERE CODCRITERIO IN (SELECT CODIGO
                                           FROM CRITERIOS
                                           WHERE DESCRIPCION = P_CRITERIO));
    V_NUMVAL NUMERIC:=0;

BEGIN

    RAISE NOTICE 'DESCRIPCION CRITERIO: %', P_CRITERIO;
    FOR V_IN1 IN IN1 LOOP
        PERFORM INFORME12(V_IN1.NOMBRE, P_CRITERIO, V_NUMVAL);
    END LOOP;
END;
$$ language plpgsql;




-------------------



INFORME 3
SELECT INFORMES(3,'BIOINGENIERIA DE ORGANISMOS UNICELULARES');
```

```
CREATE OR REPLACE FUNCTION INFORME3(P_NOMPROYECTO VARCHAR) RETURNS VOID AS $$
DECLARE
    IN3 CURSOR FOR
    SELECT O.NOMBRE AS OBJNOM, P.NOMBRE AS PROFENOM, P.APELLIDO  AS PROFEAPE
    FROM OBJETIVOS O, PROYECTOS_DE_INVESTIGACION PP, PROFESORES P
    WHERE O.CODIGOPROYECTO=PP.CODIGO
    AND PP.DNI_RESPONSABLE=P.DNI
    AND O.CODIGOPROYECTO = ( SELECT CODIGO
                             FROM PROYECTOS_DE_INVESTIGACION
                             WHERE NOMBRE = P_NOMPROYECTO);
    V_NUMVALPRO NUMERIC:=0;
    CONTADOR1 NUMERIC:=0;

BEGIN

    RAISE NOTICE 'NOMBRE PROYECTO: %', P_NOMPROYECTO;
    FOR V_IN3 IN IN3 LOOP
        RAISE NOTICE 'Investigador responsable: D. % %', V_IN3.PROFENOM,
V_IN3.PROFEAPE;
        PERFORM INFORME31(V_IN3.OBJNOM,V_NUMVALPRO);
     CONTADOR1:=CONTADOR1+1;
    END LOOP;
    RAISE NOTICE 'Número Total de Valoraciones del Proyecto %:%',
P_NOMPROYECTO, CONTADOR1;
END;
$$ language plpgsql;

CREATE OR REPLACE FUNCTION INFORME31(P_OBJETIVO VARCHAR,
                                P_NUMVALPRO NUMERIC) RETURNS VOID AS $$
DECLARE
    IN2 CURSOR FOR
    SELECT DESCRIPCION
    FROM CRITERIOS
    WHERE CODIGO IN (SELECT CODCRITERIO
                     FROM VALORACIONES
                     WHERE CODOBJETIVO IN (SELECT CODIGO
                                           FROM OBJETIVOS
                                           WHERE NOMBRE = P_OBJETIVO));
    V_NUMVALOBJ NUMERIC:=0;
    CONTADOR2 NUMERIC:=0;

BEGIN

    RAISE NOTICE 'NOMBRE OBJETIVO: %', P_OBJETIVO;
    FOR V_IN2 IN IN2 LOOP
        PERFORM INFORME21(V_IN2.DESCRIPCION,V_NUMVALOBJ);
     CONTADOR2:=CONTADOR2+1;
    END LOOP;
    P_NUMVALPRO:=P_NUMVALPRO+V_NUMVALOBJ;
    RAISE NOTICE 'Número Total de Valoraciones del Objetivo %:%', P_OBJETIVO,
CONTADOR2;
END;
$$ language plpgsql;
```

Ejercicio 7 Departamento Universitario

Realiza los módulos de programación necesarios para que el número de alumnos matriculados de una asignatura durante un curso académico esté entre 1 y 10.

```
create or replace package PkgCurso
as
    type tRegCurso IS RECORD
    (
        CursoAca matriculaciones.cursoacademico%type,
        CodAsig asignaturas.codigoasignatura%type,
        NumAlumMatri number
    );

    type tTablaCurso is table of tRegCurso
    index by binary_integer;

    v_TabCr tTablaCurso;

end;
/

create or replace trigger RellenarNumAlumnos
before insert or update on matriculaciones
declare
    cursor c_mat
    is
    select cursoacademico, codigoasignatura, count(*) as num
    from matriculaciones
    group by cursoacademico, codigoasignatura;

    i number:=0;
begin
    for v_mat in c_mat loop
        PkgCurso.v_TabCr(i).CursoAca:=v_mat.cursoacademico;
        PkgCurso.v_TabCr(i).CodAsig:=v_mat.codigoasignatura;
        PkgCurso.v_TabCr(i).NumAlumMatri:=v_mat.num;
        i:=i+1;
    end loop;
end;
/

create or replace trigger ComprobarNumAlumnos
before insert on matriculaciones
for each row
declare
    v_NumAlActual number:=0;
begin
    v_NumAlActual:=DevolverNumAl(:new.cursoacademico, :new.codigoasignatura);
    if v_NumAlActual < 0 or v_NumAlActual >= 10 then
        raise_application_error(-20003,'El número de alumnos matriculados en
ese curso académico debe estar entre 1 y 10');
    else
        CrearFilaNueva(:new.cursoacademico, :new.codigoasignatura);
    end if;
end;
/

create or replace function DevolverNumAl(p_curso matriculaciones.cursoacademico
%type,
                    p_asig asignaturas.codigoasignatura%type)
return NUMBER
is
begin
    for i in PkgCurso.v_TabCr.FIRST .. PkgCurso.v_TabCr.LAST loop
        if p_curso=PkgCurso.v_TabCr(i).CursoAca AND
p_asig=PkgCurso.v_TabCr(i).CodAsig then
            return PkgCurso.v_TabCr(i).NumAlumMatri;
        end if;
    end loop;
    return 0;
end;
/
```

```
create or replace procedure CrearFilaNueva(p_curso
matriculaciones.cursoacademico%type,
                      p_asig asignaturas.codigoasignatura%type)
is
begin
    PkgCurso.v_TabCr(PkgCurso.v_TabCr.LAST+1).CursoAca:=p_curso;
    PkgCurso.v_TabCr(PkgCurso.v_TabCr.LAST).CodAsig:=p_asig;
    PkgCurso.v_TabCr(PkgCurso.v_TabCr.LAST).NumAlumMatri:=1;
end CrearFilaNueva;
/
```

Ejercicio 8 Inventario Informático

Realiza los módulos de programación necesarios para evitar que los periodos de apagado de un mismo servidor se solapen entre sí.

```
-- Descripción: Guarda los datos necesarios de un periodo de apagado.

CREATE OR REPLACE PACKAGE Pkg_PeriodosApagado

AS

  TYPE reg_periodo IS RECORD

  (

       FechaHoraInicio  periodosdeapagado.FechaHoraInicio%type,

       FechaHoraFin     periodosdeapagado.FechaHoraFin%type,

       NumeroSerie      periodosdeapagado.NumeroSerieServidor%type

  );


  TYPE tab_periodos IS TABLE OF reg_periodo

  INDEX BY BINARY_INTEGER;


  v_periodos tab_periodos;

END;
/

-- Descripción: Rellena el paquete Pkg_PeriodosApagado con los datos actuales
de la tabla Periodosdeapagado.

CREATE OR REPLACE TRIGGER RellenarPkg_PeriodosApagado

BEFORE INSERT OR UPDATE ON Periodosdeapagado

DECLARE

  CURSOR c_datos IS

    SELECT FechaHoraInicio, FechaHoraFin, NumeroSerieServidor

    FROM Periodosdeapagado;


    indice NUMBER := 0;
```

```
BEGIN

  FOR elem IN c_datos LOOP

    Pkg_PeriodosApagado.v_periodos(indice).FechaHoraInicio :=
elem.FechaHoraInicio;

    Pkg_PeriodosApagado.v_periodos(indice).FechaHoraFin := elem.FechaHoraFin;

    Pkg_PeriodosApagado.v_periodos(indice).NumeroSerie :=
elem.NumeroSerieServidor;

  END LOOP;

END;
/


-- Descripción: Recorre el paquete, comprobando si el registro coincide con el
número de serie insertado, si coincide, se comprueba que la FechaHoraInicio
insertada

-- se encuentre entra los registros FechaHoraInicio y FechaHoraFin del paquete
o que la FechaHoraFin insertada se encuentre entra los registros
FechaHoraInicio y

-- FechaHoraFin del paquete levanta un error si se cumple alguna de dichas
condiciones.

CREATE OR REPLACE TRIGGER NoSolapan

BEFORE INSERT OR UPDATE ON Periodosdeapagado

FOR EACH ROW

DECLARE

BEGIN

  FOR elem IN Pkg_PeriodosApagado.v_periodos.FIRST ..
Pkg_PeriodosApagado.v_periodos.LAST LOOP

    IF :new.NumeroSerieServidor =
Pkg_PeriodosApagado.v_periodos(elem).NumeroSerie THEN

      IF (:new.FechaHoraInicio BETWEEN
Pkg_PeriodosApagado.v_periodos(elem).FechaHoraInicio AND
Pkg_PeriodosApagado.v_periodos(elem).FechaHoraFin)

         OR (:new.FechaHoraFin BETWEEN
Pkg_PeriodosApagado.v_periodos(elem).FechaHoraInicio AND
Pkg_PeriodosApagado.v_periodos(elem).FechaHoraFin) THEN

           RAISE_APPLICATION_ERROR(-20001, 'Los tiempos de apagado se solapan
entre sí.');

      END IF;

    END IF;

  END LOOP;

END;

/
```

Inventar, describir y resolver un problema nuevo de tablas mutantes en cualquiera de los proyectos.

Proyecto Investigadores

Realizar un trigger que impide que un profesor doctor no pueda ser responsable de más de 3 proyectos de investigación en el mismo año.

```
CREATE OR REPLACE PACKAGE PACK82
AS
        TYPE tFECHA IS RECORD
    (
        DNI_RESPONSABLE     DOCTORES.DNI%TYPE,
        FECHAINI        PROYECTOS_DE_INVESTIGACION.FECHAINI%TYPE
    );
    TYPE tTABLA IS TABLE OF tFECHA

    INDEX BY BINARY_INTEGER;
    V_TABLA tTABLA;
END;
/

CREATE OR REPLACE TRIGGER OBJSEN82
BEFORE INSERT OR UPDATE ON PROYECTOS_DE_INVESTIGACION
DECLARE
    CURSOR C8
    IS
    SELECT DNI_RESPONSABLE, FECHAINI
    FROM PROYECTOS_DE_INVESTIGACION
    WHERE DNI_RESPONSABLE IN ( SELECT DNI
                               FROM DOCTORES);
    I NUMBER:=0;
BEGIN
    FOR V_C8 IN C8 LOOP
        PACK82.V_TABLA(I).DNI_RESPONSABLE:=V_C8.DNI_RESPONSABLE;
        PACK82.V_TABLA(I).FECHAINI:=V_C8.FECHAINI;
        I:=I+1;
    END LOOP;
END;
/

CREATE OR REPLACE TRIGGER OBJFILA82
BEFORE INSERT OR UPDATE ON PROYECTOS_DE_INVESTIGACION
FOR EACH ROW
DECLARE
 V_CONTFECHA NUMBER:=0;
BEGIN
    FOR I IN PACK82.V_TABLA.FIRST..PACK82.V_TABLA.LAST LOOP
        IF PACK82.V_TABLA(I).DNI_RESPONSABLE = :NEW.DNI_RESPONSABLE AND
TO_NUMBER(TO_CHAR(PACK82.V_TABLA(I).FECHAINI,'YY')) =
TO_NUMBER(TO_CHAR(:NEW.FECHAINI,'YY')) THEN
             V_CONTFECHA:=V_CONTFECHA+1;
       END IF;
    END LOOP;
    IF V_CONTFECHA >= 3 THEN
        DBMS_OUTPUT.PUT_LINE(V_CONTFECHA);
        RAISE_APPLICATION_ERROR(-20001,'Los profesores DOCTORES no pueden
iniciar mas de 3 proyectos de investigacion en el mismo año.');
    END IF;
    V_CONTFECHA:=0;
END;
/
```

Documentar con formato de entrada de un blog técnico las diferencias encontradas entre PLSQL y pgPL/SQL.

En este documento vamos a hablar de las diferencias entre PL/SQL de Oracle y PL/PGSQL de Postgres.

- **TRIGGERS**

En PL/PGSQL, no existe el trigger como tal, sino que creamos una función que hará lo que deseemos que haga el trigger y posteriormente crearemos un trigger para llamar a esa función y le dirá cuando actuar.

La estructura sería de la siguiente manera:

Función

```
CREATE OR REPLACE FUNCTIO nombrefuncion RETURNS TRIGGER AS $nombretrigger$
DECLARE
    ...
BEGIN
    ...
END;
$nombretrigger$ LANGUAGE PLPGSQL
```

Trigger

```
CREATE OR REPLACE nombretrigger
(AFTER OR BEFORE)(INSERT,UPDATE OR DELETE) ON tabla
FOR EACH(FILA O SENTENCIA)
EXECUTE FUNCTION nombrefuncion;
```

- **INSERTING, UPDATING Y DELETING**

En PL/PGSQL, para hacer una condición de inserting, deleting o updating en triggers, tendríamos la siguiente estructura:

```
IF (TG_OP = 'INSERT')
IF (TG_OP = 'UPDATE')
IF (TG_OP = 'DELETE')
```

Que sería equivalente en Oracle a:

```
IF INSERTING
IF UPDATING
IF DELETING
```

- **EJECUTAR UNA FUNCIÓN DENTRO DE UNA FUNCIÓN Y QUE NO DEVUELVA NADA**

En PL/PGSQL, para usar una función dentro de una función usamos ‘perform’ y ‘return null’:

```
create or replace function nombre_funcion()
returns trigger as $$
declare
v_variable VARCHAR(50);
begin
        perform nombre_otra_funcion(p_parametros);
        return null;
end;
$$ LANGUAGE plpgsql;
```

- **DIFERENCIAS EN SEGUNDOS ENTRE DOS FECHAS**

En PL/SQL podemos averiguar la diferencia en segundos entre dos fechas de la siguiente forma:

```
 v_variable:=(p_fin - p_inicio) * 86400;
```

Mientras que en PL/PGSQL hay que usar:

```
v_diferencia := EXTRACT(EPOCH FROM (p_fin – p_inicio));
```

- **CURSORES**

Además, también cambia la forma de recorrer un CURSOR con un FOR. Mientras que en PL/SQL se crea el CURSOR y luego se recorre:

```
CURSOR c_cursor
IS
SELECT campo1, campo2
FROM tabla;
...
FOR i IN c_cursor LOOP
    ...
END LOOP;
```

En PL/PGSQL sería:

```
FOR c_cursor IN SELECT campo1, campo2 FROM tabla LOOP;
    ...
END LOOP;
```