Practica 1

Repaso de PL/SQL

Administración de Sistemas Gestores de Bases de Datos

Alejandro Rodriguez
Álvaro Camargo
Manuel Chacón
Carlos Hijano

Índice

Alejandro Rodriguez
ejercicio 2 ……3
ejercicio 8 ……3
ejercicio 3 ……3
Manuel Chacón
ejercicio 3 ……3
ejercicio 7 ……3
Carlos Hijano
ejercicio 2 ……4
ejercicio 4 ……4
ejercicio 5 ……4
Álvaro Camargo
ejercicio 2 ……4
ejercicio 7 ……4
Grupal
ejercicio 4 ……5
ejercicio 6 ……5
ejercicio 3 ……5
Tablas mutantes ……5
Diferencias entre PLSQL y pgPL/SQL ……5

Repositorio de GitHub con todo el trabajo:

git@github.com:hijano94/Practica1-BBDD.git

Alejandro Rodriguez

Ejercicio 2 de Carlos Hijano en Postgres

https://github.com/hijano94/Practica1-BBDD/blob/master/alejandrorodrigez-ej2

Ejercicio 8 de Carlos Hijano en ORACLE.

https://github.com/hijano94/Practica1-BBDD/blob/master/alejandrorodrigez-ej4

Ejercicio 4 de Álvaro Camargo en ORACLE.

https://github.com/hijano94/Practica1-BBDD/blob/master/alejandrorodrigez-ej8

Manuel Chacón

Ejercicio 3 de Álvaro Camargo en Postgres.

https://github.com/hijano94/Practica1-BBDD/blob/master/manuelchacon-ej3

Ejercicio 7 de Álvaro Camargo en ORACLE.

https://github.com/hijano94/Practica1-BBDD/blob/master/manuelchacon-ej7

Carlos Hijano

Ejercicio 2 de Álvaro Camargo en Postgres.

https://github.com/hijano94/Practica1-BBDD/blob/master/carloshijano-ej2

Ejercicio 4 de Álvaro Camargo en Postgres.

En este ejercicio voy a usar **py_gmail** una aplicación escrita en python para enviar correos electrónicos. Para usar esta aplicación necesitamos hacer una configuración previa.

1. Instalar la extensión de Postgres para interpretar código python.

    Apt install postgresql-plpython3

2. Descargar la aplicación del repositorio oficial en github

    git clone https://github.com/lcalisto/py_pgmail.git

3. Activar la extensión en Postgres

    create extension plpython3u;

4. Compilar en postgres la función py_pgmail/py_pgmail.sql

https://github.com/hijano94/Practica1-BBDD/blob/master/carloshijano-ej4

Ejercicio 5 de Álvaro Camargo en ORACLE.

https://github.com/hijano94/Practica1-BBDD/blob/master/carloshijano-ej5

Álvaro Camargo

Ejercicio 2 de Alejandro Rodríguez en Postgres.

https://github.com/hijano94/Practica1-BBDD/blob/master/alvarocamargo-ej2

Ejercicio 7 de Alejandro Rodríguez en ORACLE.

https://github.com/hijano94/Practica1-BBDD/blob/master/alvarocamargo-ej7

Grupal

Ejercicio 4 de Alejandro Rodríguez en ORACLE.

https://github.com/hijano94/Practica1-BBDD/blob/master/grupal-ej4

Ejercicio 6 de Alejandro Rodríguez en ORACLE.

https://github.com/hijano94/Practica1-BBDD/blob/master/grupal-ej6

Ejercicio 3 de Carlos Hijano en Postgres.

https://github.com/hijano94/Practica1-BBDD/blob/master/grupal-ej3

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

Realiza todos los módulos de programación y cambios necesarios para evitar que un mismo interprete actúe más de una vez por concierto.

https://github.com/hijano94/Practica1-BBDD/blob/master/grupal-tablasMutantes

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

La primera diferencia es en quien lo creó, PL/SQL fue patentado por Oracle, mientras que PL/PGSQL fue por PostgreSQL, despues PL/SQL es universal(con minimas modificaciones) por lo tanto es mas utilizado, mientras que PL/PGSQL al ser solo usado por PostgreSQL se utiliza menos.

Una vez hablado sobre la teoría, ahora vamos a ver las diferencias técnicas entre ellas.

Cuando hablamos de "pl/sql o pl/pgsql" pensamos en programación, una de las cosas que mas se usan son las funciones, aquí encontramos las siguientes diferencias:

PL/SQL

```
Create or replace function nombrefuncion(argumento,..)
return tipodedato
is
    variables a declarar
begin
    codigo
end;
```

PL/PGSQL

```
Create or replace function nombrefuncion(argumento,..)
returns tipodedato as $$
declare
    variables a declarar
begin
     codigo
end;
$$ LANGUAGE plpgsql;
```

Las diferencias son que PL/PGSQL cambia "is" por "AS $$" y añade la linea "declare" para diferenciar la seccion de declaracion sin contar con que PL/PGSQL acaba por "$$ LANGUAGE plpgsql;"

Lo mismo pasa al crear un procedimiento

PL/SQL

```
Create or replace procedure nombrefuncion(argumento,..)
is
    variables a declarar
begin
    codigo
end;
```

PL/PGSQL

```
Create or replace procedure nombrefuncion(argumento,..) as $$
declare
    variables a declarar
begin
     codigo
end;
$$ LANGUAGE plpgsql;
```

La forma de abrir y declarar los cursores también cambian:

declarar cursores

PL/SQL

```
CURSOR c_compositores IS argumentos;
```

PL/PGSQL

```
nombrecursor CURSOR FOR argumentos;
```

abrir cursores

PL/SQL

```
open nombrecursor;
```

PL/PGSQL

```
OPEN nombrecursor FOR consulta;
```

A la hora de crear un **trigger** en PL/PGSQL tenemos que separar el trigger de la ejecución del mismo, mientras que en oracle no es necesario separar el procedimiento del trigger

PL/SQL

```
CREATE OR REPLACE TRIGGER nombretrigger
(AFTER/BEFORE) (INSERT,UPDATE OR DELETE) ON tabla
FOR EACH (fila o sentencia)
BEGIN
    procedimiento/ejecucion
END;
```

PL/PGSQL

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

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