**Procedural Language + postgres + SQL = PL/pgSQL**

1. ¿Qué es un lenguaje procedimental o procedural?
2. ¿Qué es PL/pgSQL?
3. Revisa la liga referente a lenguajes procedimentales en la documentación oficial de postgres y contesta:

* ¿Qué otros lenguajes procedimentales existen?
* ¿En qué se diferencian de PL/pgSQL?

1. Partiendo de los recursos referentes a la sección ventajas/desventajas de PL/pgSQL y las frases que refieren a la Perspectiva de Género en relación con la autonomía económica de las mujeres, completa el siguiente cuadro (pueden realizarse breves búsquedas en internet sobre los temas tratados):

|  |  |  |
| --- | --- | --- |
| ***Enunciado*** | ***V/F*** | ***Justificación*** |
| * Es un lenguaje portable, de manera que el mismo código funciona en otros gestores * No existe desigualdad sobre la autonomía económica entre mujeres y hombres. |  |  |
| * PL/pgSQL es el único lenguaje procedimental que existe en postgres * Las políticas de igualdad de género se refieren unicamente al combate contra la violencia y discriminación. |  |  |
| * Con PL/pgSQL se pueden usar todos los tipos de datos, columnas, operadores y funciones de SQL * El marco normativo de la autonomía económica de las mujeres toma en cuenta el trabajo remunerado y no remunerado. |  |  |
| * Debido a que las funciones PL/pgSQL corren dentro de postgres; éstas, operan en cualquier plataforma donde postgres corra * La apertura del comercio internacional permite mayores oportunidades de empleabilidad para las mujeres. |  |  |
| * Es fácil manejar versiones y depurar el código * El comercio internacional es un espacio igualitario que no discrimina el sexo. |  |  |
| * Con PL/pgSQL agrupamos cálculo y consultas dentro del servidor de bases de datos, ahorrando tiempo porque no tiene la sobrecarga de una comunicación cliente/servidor. * En México, existen mecanismos que favorecen la inserción en el mercado internacional de empresas que cumplan e implementen políticas que persiguen objetivos igualitarios para las mujeres dentro de la misma. |  |  |
| * No se necesitan conocimientos de programación para generar código con PL/pgSQL * La autonomía económica de las mujeres trata de beneficios otorgados a las mujeres para recibir apoyos económicos y mejores puestos laborales. |  |  |
| * Imagina que tienes una base de datos de la que consumen datos varias aplicaciones escritas en diferentes lenguajes como Pascal, Java, PHP o Python y tienes la necesidad de realizar operaciones de inserción y actualización de datos.   Gracias a PL/pgSQL no se necesitaría escribir el código en cada uno de los lenguajes especificados, únicamente se indicaría su ejecución.   * La revolución digital de las últimas décadas ha resaltado las brechas digital y de género., imposibilitando la integración de mujeres en espacios digitales como espacios laborales. |  |  |
| * PL/pgSQL envuelve varias declaraciones en un objeto y las almacena en el servidor de base de datos * La tecnología crece y se desarrolla a pasos agigantados, generando nuevos empleos no concebibles en otra época, dichos espacios deberán ser igualitarios para lograr la participación de las mujeres en estos nuevos escenarios tecnológicos y de producción. |  |  |
| * Siempre es mejor ejecutar sentencias individualmente a agruparlas con PL/pgSQL * Las políticas públicas dirigidas para la inserción de mujeres en el entorno digital, tanto para su formación como para la práctica de su autonomía económica imposibilita a los hombres situarse en las mismas posibilidades. |  |  |

1. Clasifica los enunciados identificados en el punto anterior en ventajas o desventajas

|  |  |
| --- | --- |
| ***Ventajas*** | ***Desventajas*** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. Para cada uno de los siguientes enunciados, indica cuándo sería recomendable usar una consulta y cuando PL/pgSQL:

|  |  |
| --- | --- |
| ***Enunciado*** | ***¿SQL o PL/pgSQL?*** |
| Marisol cuenta con una base de datos para su despacho de abogados y necesita obtener el nombre de los trámites más solicitados por los clientes en el último mes |  |
| Rosa necesita mantener un control en el inventario de su restaurante. Así que, en caso de que algún ingrediente de su tabla INGREDIENTE llegué a 0, solicita que este se agregue a una nueva tabla llamada INGREDIENTE\_AGOTADO.  De esta forma Rosa puede compartir sólo la información de la segunda tabla con sus proveedores |  |
| Margarita necesita calcular la cantidad de días vacacionales para los empleados de su empresa.  Debido a que esta información depende del número de años que el empleado lleva laborando, deberá basarse en los criterios establecidos en la *Ley Federal del Trabajo* para ajustar los periodos de cada individuo y almacenarlos en su base de datos |  |

**Respuestas: Procedural Language + postgres + SQL = PL/pgSQL**

4.

|  |  |  |
| --- | --- | --- |
| ***Enunciado*** | ***V/F*** | ***Justificación*** |
| Es un lenguaje portable, de manera que el mismo código funciona en otros gestores |  | La sintaxis utilizada para la construcción de funciones, procedimientos y disparadores no es la misma en todos los gestores. |
| PL/pgSQL es el único lenguaje procedimental que existe en postgres |  | Existen más lenguajes procedimentales que pueden ser utilizados en postgres,a menudo, llamados lenguajes de desconfianza |
| Con PL/pgSQL se pueden usar todos los tipos de datos, columnas, operadores y funciones de SQL |  | Como el nombre de la actividad lo indica,PL/pgSQL suma la potencia de un lenguaje procedural y el uso de SQL. |
| Debido a que las funciones PL/pgSQL corren dentro de postgres; éstas, operan en cualquier plataforma donde postgres corra |  | PL/pgSQL corre del lado del servidor, por lo que esta afirmación es correcta |
| Es fácil manejar versiones y depurar el código |  | En caso de que la sintaxis del lenguaje cambie en una nueva versión, se necesitaría reescribir todo el código |
| Con PL/pgSQL agrupamos cálculo y consultas dentro del servidor de bases de datos, ahorrando tiempo porque no tiene la sobrecarga de una comunicación cliente/servidor |  | Cada comando SQL debe ser ejecutado individualmente por el servidor de bases de datos.  Esto significa que la aplicación cliente envía cada consulta al servidor de bases de datos, espera a que se procese, recibe el resultado, realiza cálculo, y luego envía otras consultas al servidor. Todo esto incurre en una comunicación entre procesos, el cual puede sobrecargar la red, si el cliente se encuentra en una máquina distinta al servidor de bases de datos |
| No se necesitan conocimientos de programación para generar código con PL/pgSQL |  | PL/pgSQL incorpora estructuras de control y ciclos, así que, se necesita estar familiarizado o entender la lógica de programación |
| Imagina que tienes una base de datos de la que consumen datos varias aplicaciones escritas en diferentes lenguajes como Pascal, Java, PHP o Python y tienes la necesidad de realizar operaciones de inserción y actualización de datos.  Gracias a PL/pgSQL no se necesitaría escribir el código en cada uno de los lenguajes especificados, únicamente se indicaría su ejecución. |  | Como PL/pgSQL, se ejecuta del lado del servidor, se podría escribir sólo una vez el código a utilizar y mandarlo a llamar desde las diversas aplicaciones |
| PL/pgSQL envuelve varias declaraciones en un objeto y las almacena en el servidor de base de datos |  | PL/pgSQL, nos permite agrupar sentencias y en vez de enviar una por una, realizar un sólo envío mediante los ‘objetos’ definidos |
| Siempre es mejor ejecutar sentencias individualmente a agruparlas con PL/pgSQL |  | Enviar una a una cada sentencia, conlleva un mayor tiempo de ejecución, en el caso de tener que esperar algún resultado y después realizar operaciones con ese resultado |

|  |  |
| --- | --- |
| ***Ventajas*** | ***Desventajas*** |
| Se pueden usar todos los tipos de datos, columnas, operadores y funciones de SQL | No es un lenguaje portable |
| Existen más lenguajes procedimentales que pueden ser utilizados en postgres | Es difícil manejar versiones y depurar el código |
| Las funciones PL/pgSQL operan en cualquier plataforma donde postgres corra | Se necesitan conocimientos de programación para generar código con PL/pgSQL |
| Con PL/pgSQL agrupamos cálculo y consultas dentro del servidor de bases de datos, ahorrando tiempo porque no tiene la sobrecarga de una comunicación cliente/servidor |  |
| Permite el ahorro de código |  |

1. Para cada uno de los siguientes enunciados, indica cuándo sería recomendable usar una consulta y cuando PL/pgSQL:

|  |  |
| --- | --- |
| ***Enunciado*** | ***¿SQL o PL/pgSQL?*** |
| Marisol cuenta con una base de datos para su despacho de abogados y necesita obtener el nombre de los trámites más solicitados por los clientes en el último mes | SQL, ya que Margarita únicamente necesita consultar la información y no realizar una operación o acción con ella |
| Rosa necesita mantener un control en el inventario de su restaurante. Así que, en caso de que algún ingrediente de su tabla INGREDIENTE llegué a 0, solicita que este se agregue a una nueva tabla llamada INGREDIENTE\_AGOTADO.  De esta forma Rosa puede compartir sólo la información de la segunda tabla con sus proveedores | PL/pgSQL, los requerimientos de Rosa involucran una inserción en una tabla, por lo que no podría realizarse únicamente con una consulta |
| Margarita necesita calcular la cantidad de días vacacionales para los empleados de su empresa.  Debido a que esta información depende del número de años que el empleado lleva laborando, deberá basarse en los criterios establecidos en la *Ley Federal del Trabajo* para ajustar los periodos de cada individuo y almacenarlos en su base de datos | PL/pgSQL, Margarita necesita realizar comparaciones para establecer en que rango de años se encuentra cada uno de sus empleados, por lo que, no es posible lograr lo que se pide utilizando consultas |

**Bloques anónimos, pero no desconocidos**

* Analiza los códigos que se presentan a continuación e indica cuales se han construido de manera correcta, partiendo de la sintaxis:

**<<primer\_bloque>>**

**DECLARE**

**BEGIN**

**END primer\_bloque;**

**<<primer\_bloque>>**

**BEGIN**

**END primer\_bloque;**

**DECLARE**

**BEGIN**

**END primer\_bloque;**

**<<primer\_bloque>>**

**DECLARE**

**BEGIN**

**END;**

**DECLARE**

**BEGIN**

**END;**

**BEGIN**

**END;**

**<<primer\_bloque>>**

**DECLARE**

**nombre VARCHAR(20);**

**BEGIN**

**END primer\_bloque;**

**<<primer\_bloque>>**

**DECLARE**

**VARCHAR(20) nombre := "Escribe tu nombre:";**

**BEGIN**

**END primer\_bloque;**

**<<primer\_bloque>>**

**DECLARE**

**nombre CONSTANT VARCHAR(20) := "Escribe tu nombre:";**

**es\_navidad BOOLEAN;**

**BEGIN**

**END primer\_bloque;**

**<<primer\_bloque>>**

**DECLARE**

**nombre CONSTANT VARCHAR(20) := "Escribe tu nombre:";**

**es\_navidad BOOLEAN;**

**BEGIN**

**<<segundo\_bloque>>**

**BEGIN**

**END segundo\_bloque;**

**END primer\_bloque;**

Nota: Dentro del cuerpo de un bloque, podemos definir otros bloques, veáse el esquema en la sección [PL/pgSQL Subblocks](https://www.postgresqltutorial.com/plpgsql-block-structure/)

**Respuestas: Bloques anónimos, pero no desconocidos**

* Analiza los códigos que se presentan a continuación e indica cuales se han construido de manera correcta, partiendo de la sintaxis:

**<<primer\_bloque>>**

**DECLARE -> Si no existen declaraciones, se omite la palabra DECLARE**

**BEGIN**

**END primer\_bloque;**

**<<primer\_bloque>>**

**BEGIN**

**END primer\_bloque;**

**DECLARE -> Si no existen declaraciones, se omite la palabra DECLARE**

**BEGIN**

**END primer\_bloque; -> No existe etiqueta de inicio**

**<<primer\_bloque>> -> No existe etiqueta de cierre**

**BEGIN**

**END;**

**DECLARE -> Si no existen declaraciones, se omite la palabra DECLARE**

**BEGIN**

**END;**

**BEGIN**

**END;**

**<<primer\_bloque>>**

**DECLARE**

**nombre VARCHAR(20);**

**BEGIN**

**END primer\_bloque;**

**<<primer\_bloque>>**

**DECLARE**

**VARCHAR(20) nombre := "Escribe tu nombre:";**

**BEGIN**

**END primer\_bloque;**

**<<primer\_bloque>>**

**DECLARE**

**nombre CONSTANT VARCHAR(20) := "Escribe tu nombre:";**

**es\_navidad BOOLEAN;**

**BEGIN**

**END primer\_bloque;**

**<<primer\_bloque>>**

**DECLARE**

**nombre CONSTANT VARCHAR(20) := "Escribe tu nombre:";**

**es\_navidad BOOLEAN;**

**BEGIN**

**<<segundo\_bloque>>**

**BEGIN**

**END segundo\_bloque;**

**END primer\_bloque;**

**Representación de cadenas y escape de caracteres**

* Completa el ejercicio de práctica propuesto para cada caso, tomando como base las reglas y ejemplos presentados

|  |  |  |
| --- | --- | --- |
| ***Regla*** | ***Ejemplo*** | ***Ejercicio*** |
| Se usan comillas simples para definir cadenas | Hola soy una cadena => ‘Hola soy una cadena’ | Propón otro ejemplo de cadena |
| Cuando una cadena contiene una comilla simple ('), se debe escapar duplicando la comilla simple | I’m also a string =>  ‘I’’m also a string’ | Representa la cadena:  ‘Soy’ ‘una’ ‘cadena’ ‘con’ ‘muchas’ ‘comillas’ ‘simples’ |
| Si se usa una versión anterior de PostgreSQL, se puede anteponer la cadena con E y usar la barra invertida \ para escapar de una comilla simple | I’m also a string => E’I\’m also a string’ | Representa la cadena:  ‘Soy’ ‘una’ ‘cadena’ ‘con’ ‘muchas’ ‘comillas’ ‘simples’ |
| Si una cadena contiene una barra invertida, se debe escapar usando otra barra invertida | \(n\_n)/ => E‘\\(n\_n)/’ | Representa la cadena: ‘Esta’\’es’\’una’\’oración’\’dividida’ |
| PostgreSQL versión 8.0 introdujo la notación de doble signo de dólar para hacer las cadenas más legibles.  Sintaxis:  $etiqueta$ <cadena>$etiqueta$  \*La etiqueta es opcional | ‘Soy’ ‘una’ ‘cadena’ ‘con’ ‘muchas’ ‘comillas’ ‘simples’ =>   * $‘Soy’ ‘una’ ‘cadena’ ‘con’ ‘muchas’ ‘comillas’ ‘simples’ $ * $cadena1$‘Soy’ ‘una’ ‘cadena’ ‘con’ ‘muchas’ ‘comillas’ ‘simples’ $cadena1$ | Representa la cadena: ‘Esta’\’es’\’una’\’oración’\’dividida’ |

* Con base en lo que observaste a través del ejercicio, ¿cuál consideras que es la forma más sencilla de representar cadenas?

**Sintaxis lenguaje procedimental**

* **Anotaciones**
  + Todo lo que se encuentre entre corchetes (**[]**) será opcional
  + Las palabras en **azul,** deberán cambiarse por los elementos del lenguaje mencionados en su nombre
  + Las palabras en color **rosa** representan palabras reservadas del lenguaje, por lo que siempre se escribirán de la misma forma. Aunque es posible escribirlas en mayúsculas o minúsculas
  + Los elementos en color **verde** representan caracteres que conservan su valor
* **Declaración de un bloque**

**[ <<etiqueta>> ]**

**[ declare**

**declaraciones ]**

**begin**

**sentencias**

**...**

**end [ etiqueta ];**

* **Ejecución de un bloque anónimo**

**do $$**

**[ <<etiqueta>> ]**

**[ declare**

**declaraciones ]**

**begin**

**sentencias**

**...**

**end [ etiqueta ];**

**$$**

* **RAISE**

**RAISE [ nivel ] ‘formato’ [ , *expresión* [, ... ] ];**

* **Declaración de variables**

**nombre\_de\_la\_variable tipo\_de\_dato [ := expresión ];**

* **Declaración de constantes**

**nombre\_de\_la\_constante constant tipo\_de\_dato := expresión;**

* **SELECT INTO**

**SELECT**

**lista\_de\_columnas**

**INTO**

**[ TEMPORARY | TEMP | UNLOGGED ] [ TABLE ] nombre\_de\_las\_variables**

**FROM**

**tabla**

**WHERE**

**condición;**

* **Copiar el tipo de dato de una columna**

**nombre\_de\_la\_variable nombre\_de\_la\_tabla.nombre\_de\_la\_columna;**

* **%type**

**nombre\_de\_la\_variable variable%type;**

* **Declaración de una variable de tipo fila**

**nombre\_de\_la\_variable nombre\_de\_la\_tabla%rowtype;**

* **Acceso a una columna de tipo fila**

**nombre\_de\_la\_variable.nombre\_de\_la\_columna**

* **Declaración de una variable de tipo registro**

**nombre\_de\_la\_variable record;**

* **Acceso a una columna de tipo registro**

**nombre\_de\_la\_variable.nombre\_de\_la\_columna**