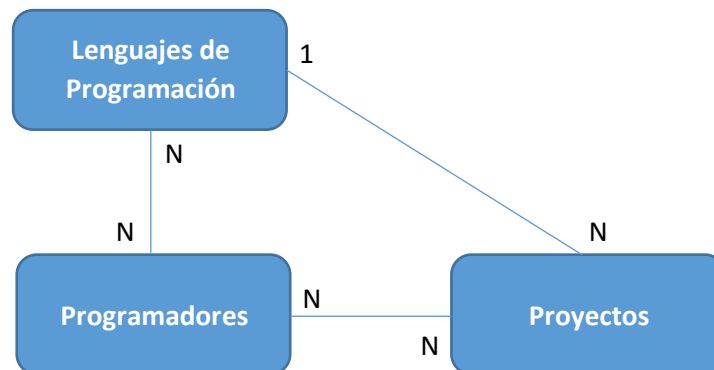


OLSoftware Soluciones Tecnológicas Efectivas	PRUEBA PRÁCTICA PL/SQL		
	CODIGO: FM-GH-50	VERSION: 01	Página 1 de 4
	F. EMISION: 24-07-2020		F. ACTUALIZACION: N/A

Descripción del Problema

A nuestra empresa OLSoftware SAS se le solicito un desarrollo de software basado en el siguiente modelo de datos:



Como se puede observar el modelo de datos no se encuentra normalizado, lo que es un grave problema para alcanzar los objetivos definidos por cliente. Tras varias reuniones con el cliente se logró entender la necesidad del cliente y consolido la siguiente información:

Modelo de Datos


Los lenguajes de Programación debe tener los siguientes atributos: Un Identificador único, Nombre o Descripción (Máximo 100 caracteres), Estado (Solo puede ser A-Activo o I-Inactivo).

Los programadores deben tener un identificador único, Primer Nombre (Obligatorio), Segundo Nombre (Opcional), Primer Apellido (Obligatorio), Segundo Apellido (Opcional), Edad (Solo enteros positivos), Años de Experiencia (Solo enteros positivos), Sexo (M-Masculino o F-Femenino).

Los proyectos deben tener un identificador único, Nombre o Descripción (Máximo 100 caracteres), Valor (solo se aceptan 2 decimales), Fecha de inicio, Fecha de finalización (no puede ser menor a la fecha de inicio), Cantidad de Programadores necesarios (Solo enteros positivos), Lenguaje de Programación Requerido, Estado (Solo puede ser V-Vigente o T-Terminado).

Reglas

- Un programador puede estar en varios proyectos
- En un proyecto pueden estar varios programadores
- Un programador puede dominar varios lenguajes de programación
- Un lenguaje de programación puede ser dominado por varios programadores

 Soluciones Tecnológicas Efectivas	PRUEBA PRÁCTICA PL/SQL		
	CODIGO: FM-GH-50	VERSION: 01	Página 2 de 4
	F. EMISION: 24-07-2020		F. ACTUALIZACION: N/A

- Un proyecto puede tener solo un lenguaje de programación base
- Un lenguaje de programación puede estar vinculado a varios proyectos

Consideraciones


- Cada entidad debe contener dos campos adicionales (Usuario y Fecha de Actualización) de auditoria para tener control del usuario que modifique cualquier registro en particular
- Cada identificador único de cada tabla debe ser generado por una secuencia de Base de Datos
- Cada secuencia de Base de Datos debe ser gestionada por un trigger o disparador, además cada vez que se realice una inserción o actualización en cada tabla, el trigger debe actualizar la información de los campos de auditoria (Usuario y Fecha de Actualización).
- Cada Entidad debe tener su respectiva llave primaria y llaves foráneas (las que considere necesarias).
- Utilizar Check Constraint donde sea necesario (Validaciones a nivel de tabla).
- Definir los índices que considere necesarios.
- No utilizar cursores implícitos.
- Todos los procedimientos, funcionalidades, tipo record y tablas PL/SQL que se construyan deben estar contenidas o pertenecer a un paquete de base de datos.
- Las sentencias DML y DDL utilizadas deben ser consignadas en un archivo con extensión .sql.

Requerimientos

1. Normalizar el Modelo de Datos (Si lo considera viable y necesario).
2. Construcción del Modelo de Datos (Entidades, Secuencias y Triggers).
3. Poblar el Modelo de Datos (Generar los Insert correspondiente para cada entidad base: Proyecto, Programador y Lenguaje de Programación)
 - a. Lenguajes de programación: Crear 10 Registros
 - b. Programadores: Crear 25 Registros
 - c. Proyectos: Crear 15 Registros (Un lenguaje de programación diferente para proyecto)
4. Construir un procedimiento o funcionalidad que realice una asignación aleatoria de **3 lenguajes de programación** a cada **programador** (No se puede repetir un mismo lenguaje de programación para el mismo programador).

Reglas

- Adicionar un campo en la entidad que considere correcta para almacenar puntos de experiencia del programador en cada lenguaje de programación.

 Soluciones Tecnológicas Efectivas	PRUEBA PRÁCTICA PL/SQL		
	CODIGO: FM-GH-50	VERSION: 01	Página 3 de 4
	F. EMISION: 24-07-2020		F. ACTUALIZACION: N/A

- Los puntos de experiencia del programador deben estar en un rango de 0 a 100 (Generar este valor aleatoriamente).
- Implementar la cláusula MERGE para el control de las sentencias DML involucradas.
- Generar datos semilla

5. Construir un procedimiento o funcionalidad que realice la asignación de la cantidad de programadores requerida por cada proyecto (No se puede repetir un mismo programador para el mismo proyecto).

Reglas


- La asignación de cada programador depende del lenguaje de programación necesario en cada proyecto y la cantidad de puntos de experiencia (Prioridad de mayor a menor) de cada programador en el lenguaje de programación requerido por el proyecto.
- Un programador solo puede estar en dos proyectos al mismo tiempo de acuerdo a la fecha de inicio y la fecha de finalización de cada proyecto.
- Generar datos semilla

6. Construir una función que retorne un cursor con el promedio aritmético de los puntos de experiencia de los programadores en los lenguajes de programación que dominan.

Reglas

- El cursor que se debe retornar solo debe tener los campos: Nombre del Programador (Concatenar los campos: Primer Nombre, Segundo Nombre, Primer Apellido, Segundo Apellido) y el promedio aritmético de los puntos de experiencia de los programadores en los lenguajes de programación que dominan.
- El cursor resultante debe estar ordenado por el promedio aritmético de forma descendente.

7. Construir un procedimiento que implemente o invoque la función creada en el **requerimiento 6**, luego recorrer el cursor retornado por la función e imprimir en consola (Salida DBMS) la información que contiene el cursor, con el siguiente formato: Nombre del Programador -> Promedio.
8. Construir un **tipo record** con los siguientes campos: Nombre Del Programador, Años de Experiencia, Puntos de Experiencia en el lenguaje del proyecto, Nombre del Proyecto, Valor del Proyecto y el identificador de fila de la entidad – Proyecto (donde se almacenará el campo **ROWID** de la tabla).
9. Construir una tabla PLSQL del tipo record creado en el requerimiento anterior.

 Soluciones Tecnológicas Efectivas	PRUEBA PRÁCTICA PL/SQL		
	CODIGO: FM-GH-50	VERSION: 01	Página 4 de 4
	F. EMISION: 24-07-2020	F. ACTUALIZACION: N/A	

10. Construir un procedimiento que implemente la **tabla PLSQL** creada en el requerimiento anterior, la **tabla PLSQL** debe ser alimentada con los registros obtenidos de un cursor explícito que obedezca a la estructura del tipo record creado en el requerimiento 8, seguidamente recorrer la **tabla PLSQL** y actualizar el estado de los proyectos a **T-Terminado** utilizando el campo **ROWID** en la cláusula **WHERE** para encontrar el proyecto respectivo.

Reglas

- El cursor explícito que se utiliza para alimentar la **tabla PLSQL** debe contener 5 registros, correspondientes a los proyectos con menor valor.
- Los 5 proyectos retornados en el cursor deben ser distintos.