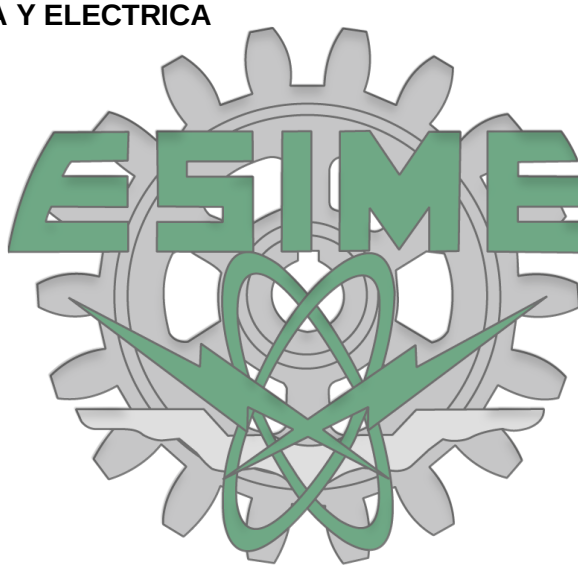
INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELECTRICA
UNIDAD CULHUACÁN



Bases de datos

RODRIGO EMMANUEL FLORES AVALOS

Grupo: 7CV3
Trabajo: Bases de datos: Unidad 3

ESIME Culhuacán, 22 de Septiembre de 2020.

Contenido

Clases, objetos e instancias

Estos son conceptos que se manejan principalmente en las bases de datos orientadas a objetos o documentos como lo son MongoDB o Cosmos DB

Clases: Es un patrón o plantilla en la que se basan objetos que son similares. Cuando un programa crea un objeto de una clase, proporciona datos para sus variables y el objeto puede entonces utilizar los métodos que se han escrito para la clase. Todos los objetos creados a partir de la misma clase comparten los mismos procedimientos para sus métodos, también tienen los mismos tipos para sus datos, pero los valores pueden diferir. Una clase también es un tipo de datos. De hecho, una clase es una implementación de lo que se conoce como un tipo abstracto de datos. El que una clase sea también un tipo de datos significa que una clase se puede utilizar como tipo de datos de un atributo.

Instancias: Una clase es una especificación del comportamiento abstracto y del estado abstracto de un tipo de objeto. Las clases son instanciables, por lo que a partir de ellas se pueden crear instancias de objetos individuales (es el equivalente a una clase concreta en los lenguajes de programación). El estándar propuesto soporta la herencia simple y la herencia múltiple mediante las interfaces. Ya que las interfaces no son instanciables, se suelen utilizar para especificar operaciones abstractas que pueden ser heredadas por clases o por otras interfaces. A esto se le denomina herencia de comportamiento y se especifica mediante el símbolo “:”. La herencia de comportamiento requiere que el súpertipo sea una interface, mientras que el subtipo puede ser una clase o una interface.

Objeto: Es un elemento autocontenido utilizado por el programa. Los valores que almacena un objeto se denominan atributos, variables o propiedades. Los objetos pueden realizar acciones, que se denominan métodos, servicios, funciones, procedimientos u operaciones. Los objetos tienen un gran sentido de la privacidad, por lo que solo dan información sobre sí mismos a través de los métodos que poseen para compartir su información. también ocultan la implementación de sus procedimientos, aunque es muy sencillo pedirles que los ejecuten. Los usuarios y los programas de aplicación no pueden ver que hay dentro de los métodos, solo pueden ver los resultados de ejecutarlos. A esto es a lo que se denomina ocultación de información o encapsulamiento de datos. Cada objeto presenta una interface publica al resto de objetos que pueden utilizarlo. Una de las mayores ventajas del encapsulamiento es que mientras que la interface publica sea la misma, se puede cambiar la implementación de los métodos sin que sea necesario informar al resto de objetos que los utilizan. Para pedir datos a un objeto o que este

realice una acción se le debe enviar un mensaje. Un programa orientado a objetos es un conjunto de objetos que tienen atributos y métodos. Los objetos interactúan enviando- se mensajes. La clave, por supuesto, es averiguar que objetos necesita el programa y cuales deben ser sus atributos y sus métodos

El lenguaje de descripción de objetos ODL

ODL es un lenguaje de especificación para definir tipos de objetos para sistemas compatibles con ODMG. ODL es el equivalente del DDL (lenguaje de definición de datos) de los SGBD tradicionales. Define los atributos y las relaciones entre tipos, y especifica la signatura de las operaciones. La sintaxis de ODL extiende el lenguaje de definición de interfaces (IDL) de la arquitectura CORBA (Common Object Request Broker Architecture).

```
class Persona
(extent personas key dni)
{
/* Definicion de atributos */
attribute struct Nom Persona {string nombre pila, string apellido1,
string apellido2} nombre;
attribute string dni;
attribute date fecha nacim;
attribute enum Genero{F,M} sexo;
attribute struct Direccion {string calle, string cp, string ciudad}
direccion;
/* Definicion de operaciones */
float edad();
}
class Profesor extends Persona
(extent profesores)
{
/* Definicion de atributos */
attribute string categoria;
attribute float salario;
attribute string despacho;
attributo string telefono;
/* Definicion de relaciones */
relationship Departamento trabaja en
```

```
inverse Departamento::tiene profesores;
relationship Set<EstudianteGrad> tutoriza
inverse EstudianteGrad::tutor;
relationship Set<EstudianteGrad> en comite
inverse EstudianteGrad::comite;
/* Definicion de operaciones */
void aumentar salario(in float aumento);
void promocionar(in string nueva categoria);
}
class Estudiante extends Persona
(extent estudiantes)
{
/* Definicion de atributos */
attribute string titulacion;
/* Definicion de relaciones */
relationship set<Calificacion> ediciones cursadas
inverse Calificacion::estudiante;
relationship set<EdicionActual> matriculado
inverse EdicionActual::estudiantes matriculados;
/* Definicion de operaciones */
float nota media();
void matricularse(in short num edic) raises(edicion no valida,
edicion llena);
void calificar(in short num edic; in float nota)
raises(edicion no valida, nota no valida);
};
```