Curso Básico de Programación Orientada a Objetos y Modelado de Datos.

Si ya tienes conocimientos más que básicos de programación orientada a objetos, hazte este módulo completo, te garantizo que vas a encontrar pepitas de oro que aun cuando no lo creas, te aportarán bastante conocimiento inclusive para explicar a otros de una manera mas entendible y hay par de truquitos que quizá no conozcas porque son nuevos en las diversas versiones.

Si tienes poco o ningún conocimiento de programación, te recomiendo que te saltes ciertos temas de este módulo y solo hagas los temas: “conceptos generales”, “Un breve repaso de programación” y “Un simple repaso de programación orientada a objetos”. Y los demás te los saltes hasta que yo te indique mas adelante que estas listo para verlos.

Lo mismo aplica con Base de Datos 101 y Base de Datos X, si tienes conocimientos básicos o nulos, solo hazte el primero y si ya conoces a un nivel un tanto superior, hazte los dos temas.

Es de vital importancia que sigas estas instrucciones, previamente indicadas, primero para que no te desanimes porque no entiendes nada y segundo para que puedas sorprenderte cuando en el curso yo desbarate por completo un método y lo rehaga de cero con un mejor principio.

Conceptos Generales

…Tema en construcción…

Abstracto

Podemos verlo como una representación imaginaria de ese algo que puede existir o no. Describe a la intención de no representar seres u objetos concretos; en cambio, se contemplan sólo elementos de forma, color, estructura o proporción, por ejemplo.

Entidad

Es la representación abstracta, de aquello del mundo real (o imaginario) que queremos emular, mediante algún software de computadoras.

Concepto

Representación abstracta que se refiere a un conjunto de entidades con características comunes. Un concepto es la representación mental de una categoría.

Es siempre una abstracción que los humanos hacemos para referirnos de forma comprensiva a una propiedad de las cosas o fenómenos que observamos. Los conceptos se inventan, no se descubren, se definen en términos de operaciones que los identifican en función de los hechos que conocemos y están relacionados. Los conceptos no pertenecen al dominio de las potencialidades sino al de las realidades individuales efectivas en un momento temporal concreto. Un niño puede tener aptitudes musicales pero su rendimiento con un determinado instrumento depende de haber aprendido su uso y de haber desarrollado las destrezas que le permitirán ejecutar una melodía o pieza musical.

Caracteres
Cammel Notation

Un breve repaso de programación.

Una variable no es más que una cajita de información donde introducimos un valor que debe ser del tipo que le hayamos especificado a la variable, en un cover de iphone no podemos meter una licuadora.

Los tipos de datos son los que me permiten saber qué tipo de información puedo almacenar en las cajitas, si son números, “float”, “doublé” o “decimal”, así como “int”, si es letras (cadenas de caracteres) “string”, si son verdadero o falso “boolean”, si son fechas “DateTime”.

Asumo que ustedes saben que para la computadora todos caracteres incluyendo los números no son más que dibujitos que ella debe interpretar y para poder interpretarlos esta necesita que se le asigne el tipo de datos para poder diferenciar un 1 de un “1”.

Un simple repaso de Programación Orientada a Objetos

Un objeto no es más que cualquier elemento sobre el cual puedo interactuar, normalmente son las representaciones abstractas de las cosas del mundo real, abstracto podemos verlo como una representación imaginaria de ese algo, mediante 0 y 1.

Un objeto en C# es una clase, la cual yo le voy a añadir atributos, y como dato curioso todo en c# es un objeto, incluyendo los tipos de datos primitivos.

Los atributos son esas características que debemos añadir a nuestro objeto para que sea ese objeto, es decir, es lo que hace que la cosa sea esa cosa que queremos representar, Ejemplo, si quiero representar un Potémon, el potemon, tiene ciertas características que lo hacen ser un potemon, como por ejemplo, quiero representar a pitachu, pitachu tiene cola, cachetes, ojos, boca, estas son sus características, estos son sus atributos; En el caso de un paciente, por ejemplo, tenemos que dentro de sus características están el record, el nombre, el apellido, el tipo de sangre, etc. Empleado de igual forma tenemos código de empleado, nombre, cedula, etc.

La herencia no es más que yo heredar características de mi padre de forma implícita, lo cual hace que yo no necesite implementar esas características, ejemplo yo tengo un Potémon que se llama Pitachu y otro que se llama vamo a' calmano, ambos son Potémon, por lo cual, yo puedo perfectamente crear una clase llamada Potémon, con características base que todos los Potémons tengan, tales como: Nombre, Tipo1, Tipo2, Ataque, Defensa, Velocidad y Puntos de vida.
En el caso de un paciente, podríamos tener una clase padre que se llame Persona y ese padre tenga Nombre, Apellido y Cedula.

Luego tendríamos los hijos que si le especificamos que son nuestros hijos entonces no necesitamos repetir esas características adicionales.

El Potémon Pitachu hereda de la clase base Potémon y vamo a' calmano también, por lo que solo cabría añadir esas características únicas que hacen que Pitachu sea un Pitachu, como cola, cachetes rojos, etc, y en Encuero añadiríamos otras como caparazón, color azul, etc.

En el caso de pacientes y empleados seria solo añadir el Récord y el tipo de sangre si es paciente, y como empleado el salario.

Un NO tan simple repaso de Programación

…Tema en Construcción…

User Defined Types
-Clase

Lista de elementos

Values Versus Reference Types
Arrays And Collections
LINQ
Program Flow Introduction
Comparando valores,
IfThenElse
Getting Loopy
Switch
Debugging
-breakpoints, inspect elements, change values line by line, f10 and f11

Un no tan simple repaso de POO

-Encapsulamiento
Herencia
-Polimorfismo
Niveles de acceso
Case sensitive
Metodos
Constructor,
Classes & Libraries
Using
namespaces
Access Modifiers

Base de Datos 101

Tabla

No es más que la representación “abstracta” **de la entidad**, es decir, aquello que queremos representar del mundo real o aquello que queremos conceptualizar mediante un objeto de base de datos.

Por ejemplo, si nosotros tenemos un proyecto (del mundo real) en el cual queremos eficientizar la lenta búsqueda de los records físicos de los pacientes, tenemos varias entidades, una de ellas es paciente, y otra entidad puede ser tipo de paciente, estas son nuestras entidades, entonces, la tabla es la representación en la base de datos de esas entidades, no necesariamente deben llamarse igual, pero dependiendo de cada caso, se recomienda que sean así.

Como buena práctica, las tablas se nombran según la entidad que desean representar, pero en plural, por lo que la entidad “Paciente” se representaría como “Pacientes” en nuestra tabla.

Para nombras las tablas, puedes usar toda la longitud necesaria, por el amor a la divinidad, no uses nomenclaturas extrañas disque porque viste en un tutorial de los años 80 que es bueno usar nombres complejos para evitar que un hacker sepa la información que esta robando, si el hacker llegó ahí, él va a averiguar qué significa cada tabla, sin embargo el dolor de cabeza y la falta de productividad que conlleva usar nomenclaturas, no compensa para nada la poca lucha que le vas a sumar a un eventual atacante. Usa nombres para tus tablas que sean lo suficientemente descriptivas. Por ejemplo, para la tabla “préstamos personales del paciente” puedes usar “PacientesPrestamos” o “Prestamos_Pacientes” pero nunca “ePrePacPer” o “xadmprepac”, **eso no son malas prácticas, eso son ridiculeces.**

Las tablas se pueden nombrar usando cualquiera de las reglas existentes, con las limitantes de que deben iniciar por una letra o el underline (_), y que no se pueden usar caracteres especiales, puedes investigarlas y usar la que más te guste, sin embargo, a mí me gusta la Cammel Notation y esta es la que veremos en el curso.

Campos

No son más que esos atributos o características **de la cual deseamos almacenar información.**

Hay que recordar que un atributo son esos elementos que hacen que la entidad sea la entidad, por ejemplo, lo que hace que un paciente sea un paciente, son su nombre, apellido, tiene características como color de piel, color de ojos, tiene otros atributos que pueden ser estado de salud, alergias, dirección, etc.

De todas esas características que vuelven a nuestra entidad, una entidad, nosotros solo queremos almacenar algunas, puede que no nos interese los gustos musicales del paciente, entonces, esos datos que deseamos conservar del paciente en nuestra tabla son los campos.

Tipos de datos

Son los que sirven para identificar el tipo de información que se puede almacenar en un campo. Son los mismos que en programación, nada que agregar, excepto que algunos cambian de nombre, lo que en programación es un string en base de datos es un varchar o nvarchar, lo que en programación es un DateTime, en base de datos puede ser un DateTime, o se puede almacenar solo un Date, así como tambien se puede guardar solo un Time.

Longitud y restricciones

Algunos campos, se les puede indicar la longitud máxima de caracteres que soportan, por ejemplo, nombre varchar(50) significa que no pueden guardar un nombre mayor a 50 caracteres

Las restricciones son aquellas limitantes que ponemos sobre ciertos campos, existen restricciones por defecto, como que por ejemplo el Id nunca puede contener un valor nulo o que un campo solo admita ciertos caracteres, para el número telefónico, por ejemplo, se puede configurar para que solo se admitan números, guiones y paréntesis. Se puede especificar que un campo como nombre, no se pueda dejar vacío, entre otras restricciones.

Registros

Son el conjunto de informaciones que insertamos sobre la entidad en el grupo de campos que representan a una entidad en concreto dentro de nuestra tabla.

Es decir, es el conjunto de información que tenemos sobre un elemento de una entidad, por ejemplo, nosotros tenemos la información medica del paciente Starling Germosen que pesa 250 libras y tiene 31 años, esto es un registro, pero en la misma tabla tenemos también la información de Anilda Rodriguez, que pesa 160 libras y tiene 27 años, este entonces es otro registro.
Los registros normalmente se identifican por un campo único que no se puede repetir entre ninguno de los registros, para poder identificarlos rápidamente, por lo regular este campo se nombre como “ID” que es la llave primaria de la tabla.

Primera Forma Normal

Las reglas de Cood, son unas reglas que establecen la estructura de como deberían normalizarse ciertas tablas, en una base de datos relacional, a los fines de que la información sea lo mas segregada posible y de esa forma evitemos la duplicidad innecesaria de información.

Por ejemplo, tenemos los siguientes registros de pacientes.

PacienteId	**Nombre**	**Apellido**	**TipoSangre**	**Sexo**	**Edad**
1	Starling	Germosen	A+	Masculino	31
2	Ana	Valdez	A+	Femenino	25
3	Anilda	Rodriguez	O-	Femenino	27
4	Albert	Recio	B+	NA	20

Si nos fijamos bien, hay información que se repite, como el tipo de sangre y el sexo, la primera forma normal, nos dice que deberíamos separar esto en tablas que nos permitan evitar la duplicidad de información, tendríamos algo como esto

SexoId	**Nombre**
1	Masculino

2	Femenino
3	NA

TipoSangreId	Descripcion
1	Desconocido
2	A+
3	A-
4	B+
5	B+
6	AB+
7	AB-
8	O+
9	O-

PacienteId	Nombre	Apellido	TipoSangreId	SexoId	Edad
1	Starling	Germosen	2	1	31
2	Ana	Valdez	2	2	25
3	Anilda	Rodriguez	9	2	27
4	Albert	Recio	5	3	20

Llaves

Son las que permiten establecer relaciones entre tablas, constan de dos vías por lo regular, **una llave primaria que es el identificador único que no puede repetirse en la que viene a ser la tabla padre o tabla maestra**, (por algo es único), y **una llave foránea que es la que apunta hacia una llave primaria de otra tabla y es hacia donde se hace la relación.** En el ejemplo anterior la llave primaria en la tabla Paciente es PacienteId, mientras que las llaves foráneas son TipoSangreId y SexoId.

Relaciones

Las relaciones son las que me permiten establecer interconexiones entre las informaciones que deseo saber de una tabla pero que la información adicional de ese dato se encuentra en otra tabla. Como vimos en tópico de la primera forma normal.

Por ejemplo, yo tengo mi tabla “Pacientes” donde tengo al paciente Starling Germosen, con el Id 1, sin embargo, la información de su tipo de sangre es solo un valor numérico que apunta a la información que está en otra tabla.

Existen relaciones de uno a uno, uno a mucho, mucho a mucho (no recomendado y técnicamente no soportado por Entity Framework).

Tener pendiente

Recuerden que no tiene sentido para fines prácticos tener una entidad persona con datos que no siempre van a ser llenados, por eso se realiza una normalización, recordando que normalización no es más que hacer que nuestra base de datos o más bien las tablas de nuestra base de datos, contenga la menor cantidad de incongruencia y datos repetidos o nulos, en una normalización muy estricta, campos como el teléfono, correo y dirección van en tablas separadas, pero ese no es el tema.

Base de Datos X

…..Tema en construcción….