TEMA 089. ANÁLISIS DEL DOMINIO DE LOS SISTEMAS: MODELADO DE DOMINIO, MODELO ENTIDAD RELACIÓN Y MODELOS DE CLASES.

Actualizado a 09/04/2023

1. ANÁLISIS Y MODELADO

El análisis de un sistema de información representa el "qué" y el diseño el "cómo". Para establecer las características que debe cumplir el sistema para cubrir las necesidades de los usuarios se describen requisitos.

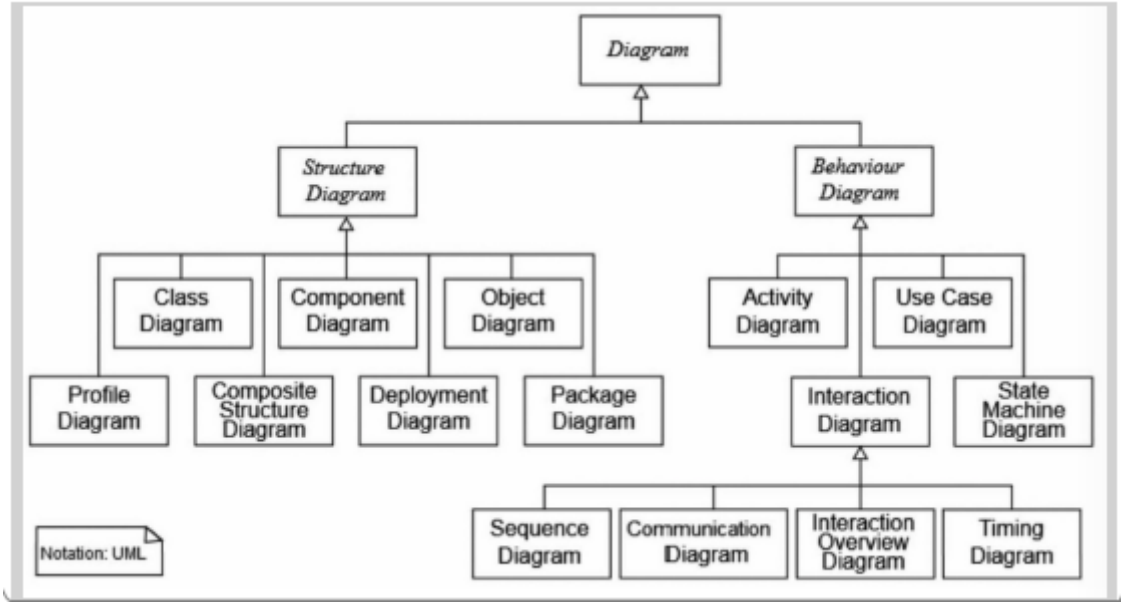
El enfoque de análisis puede ser **estructurado** u **orientado a objetos (OO)**. Estándares relacionados: ISO/IEC/IEEE 29148; ISO/IEC/IEEE 15288; ISO/IEC 24766.

El análisis del sistema se realiza en varios niveles:

- **Estático**: modelo de datos → técnicas: modelo E/R (estructurado) vs diagrama de objetos/clases (OO)
- **Funcional**: modelo de procesos →DFD[1] y DFC (estructurado) vs diagrama de casos de uso (OO)
- **Dinámico**: comportamiento → DTE [2]y HVE[3] (estructurado) vs diagrama de transición de estados o diagramas de interacción (secuencia y colaboración) (OO).

1.1. UML

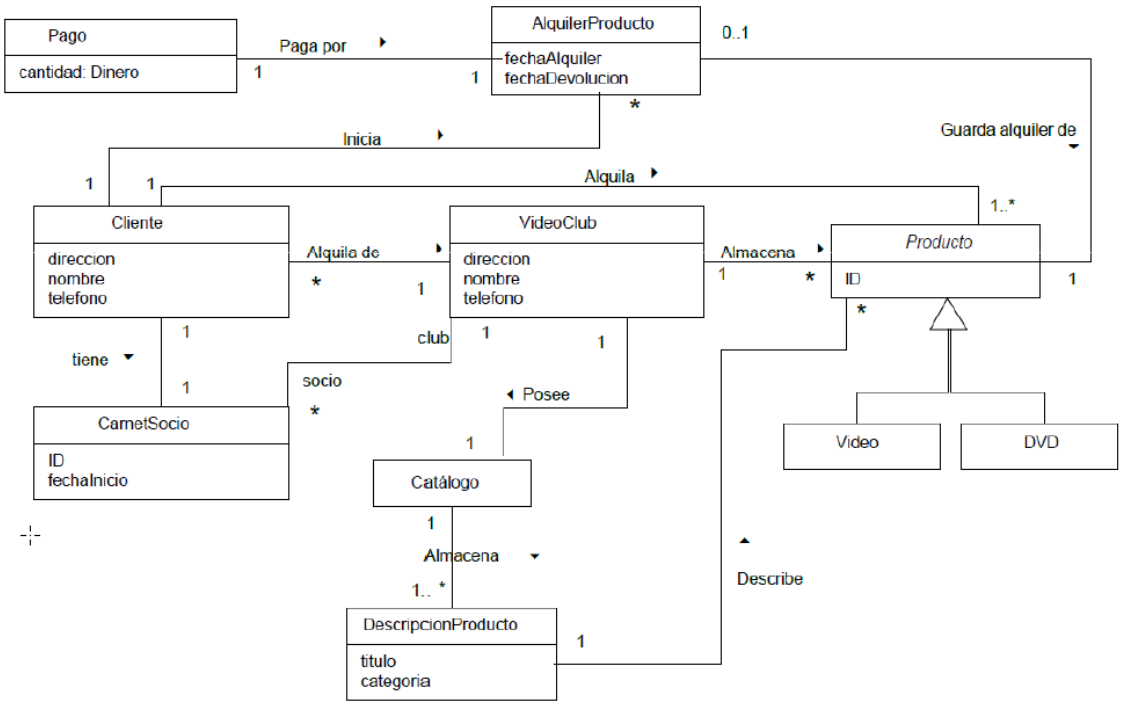
Es un **lenguaje de modelado** estandarizado por el OMG. Permite modelar, construir y documentar software, independientemente del proceso desarrollo. La versión actual es la 2.5.1 de diciembre 2017 y define los siguientes tipos de diagramas.



1.2. MODELO DE DOMINIO

Modelo conceptual para describir un problema específico representando el vocabulario y los conceptos clave a través de entidades, atributos, papeles y relaciones, así como restricciones. Se representa mediante un diagrama de clases con clases conceptuales (sin métodos), para comprender el problema sin contemplar detalles en cuanto a implementación.

[1] Diagrama de Flujo de Datos
[2] Diagrama de Transición de Estados
[3] Historia de Vida de las Entidades



2. ORIENTACIÓN A OBJETOS

- ✓ Sketchpad, Simula, Smalltalk, Oberon, Eiffel, C++ → Híbridos, Lenguajes visuales de 4ª generación. Arquitecturas Java y .NET

Análisis Orientado a Objetos "es un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos encontrados en el vocabulario de del dominio del problema".

El **diseño orientado a objetos** es un refinamiento del análisis, un refinamiento de los modelos generados para su posterior implementación.

Programación Orientada a Objetos se define como: "un método de implementación en el cual los programas son organizados como grupos cooperativos de objetos, cada uno de los cuales representa una instancia de alguna clase, y estas clases son miembros de una jerarquía de clases unidas vía relaciones de herencia".

Ventajas OO:

- ✓ Suministra modelos similares a los del mundo real
- ✓ Facilita el desarrollo de sistemas complejos, generando sistemas más preparados al cambio
- ✓ **Facilita la reutilización de software y diseños -> aumento de la productividad**
- ✓ Permite el desarrollo iterativo de aplicaciones
- ✓ Permite la interoperabilidad entre aplicaciones, aislando las dependencias de las plataformas
- ✓ Los sistemas orientados a objetos son generalmente más pequeños que su equivalente no orientado a objetos, lo que supone menos código y más reutilización
- ✓ Vale para aplicaciones de pequeño y gran tamaño

Características del diseño orientado a objetos:

- ✓ Modularidad
- ✓ Extensibilidad
- ✓ Ocultación de implementación (information hiding)

- ✓ Integrable
- ✓ Acoplamiento débil
- ✓ Reusabilidad
- ✓ Cohesión fuerte

Metodologías de Diseño Orientadas a Objetos (según Jacobson)

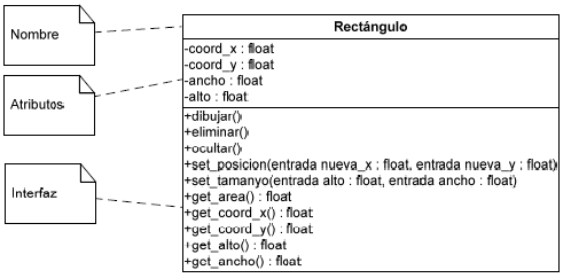
- ✓ Object-Oriented Design (OOD), Booch.
- ✓ Object Modeling Technique (OMT), Rumbaugh.
- ✓ Object Oriented Analysis (OOA), Coad/Yourdon.
- ✓ Hierarchical Object Oriented Design (HOOD), ESA.
- ✓ Object Oriented Structured Design (OOSD), Wasserman.
- ✓ Object Oriented Systems Analysis (OOSA), Shaler y Mellor
- ✓ UML: Unified Modelling Language. Realmente UML no es una metodología (en todo caso se debería considerar RUP)

SIGLAS	NOMBRE	FASES	DESCRIPCIÓN
GOOD	General Object-Oriented Design	D	Utilizada fundamentalmente para entornos en tiempo real y ADA
HOOD	Hierarchical Objet-Oriented Design	D	Se basa en un diseño descendente jerárquico
OOSD	Object-Oriented Structured Design	D	Notación derivada de Booch y de los diagramas de estructura de Constantine que soporta concurrencia
OMT	Object Modelling Technique	A/D/i	Parte de la idea de utilizar los mismos conceptos y notación a lo largo de todo el ciclo de vida
OODLE	Object-Oriented Design Language	D	Notación gráfica independiente del lenguaje para OO de programas, librerías y entornos
OOSE	Object-Oriented Software Engineering	A/D/I	Versión reducida de Objetory ("proceso" industrial para el desarrollo OO de sistemas)
SQMA	Object-Oriented Modelling Approach	A	Extensión de la metodología OOA de Coad y Yourdon, que incorpora algunas ideas de IA y modelización semántica de datos
OOD/ BOOCH	Object-Oriented Design	D/I	Se basa en un descripción muy completa de los sistemas, tanto lógica como física, apoyándose en diagramas
OOA/ OOD	Object-Oriented Analysis & Design	A/D	Elabora 5 "niveles" (clases/objetos, estructuras, sujetos, atributos, servicios) durante el análisis y 4 "componentes" durante el diseño (gestores de tareas, datos e interacción humana)
OOSA	SCHALER y MELLOR	A	Resulta más bien una colección de técnicas estructuradas que de OO

2.1. CLASES, OBJETOS Y MÉTODOS

Las **clases** son un **concepto estático** definido en el programa fuente, una abstracción de la esencia de un objeto. Es un tipo abstracto de datos.

- ✓ Una clase define a un conjunto de objetos que comparten una misma estructura y comportamiento común.
- ✓ Es el conjunto de métodos y atributos que resumen las características comunes de todos los objetos que la componen
- ✓ Objeto es un modelo o Instancia de una clase.
- ✓ No todas las clases tienen por qué tener instancias; a una clase que no se puede instanciar se denomina **clase abstracta** (es una clase base para conceptos generales, tiene al menos un método declarado pero no definido para que las clases derivadas proporcionen implementaciones. Puede tener atributos y/o métodos con implementación)
- ✓ El **interfaz** de la clase de objetos estará definido por el conjunto de métodos que soporta y los mensajes que es capaz de tratar (es un contrato. Todos los métodos declarados tienen que ser implementados en una subclase. No confundir con las clases abstractas)
- ✓ Al conjunto de clases utilizadas para una tarea determinada de programación se le denomina **biblioteca de clases** (diferencia entre biblioteca de clases (xa un sistema en concreto) y de componentes (N sistemas distintos)).



Los atributos y operaciones pueden tener los siguientes tipos de acceso.
+ Público: Se pueden acceder desde cualquier clase.
- Privado: Sólo se pueden acceder desde operaciones de la clase.
Protegido: Sólo se pueden acceder desde operaciones de la clase o de clases derivadas.

Se recomienda, para estar alineado con los conceptos de encapsulamiento y abstracción

Un **objeto** es una entidad que se caracteriza porque tiene:

- ✓ Estado (propiedades + valores): Este estado se representa con un conjunto de atributos del objeto
- ✓ Comportamiento (acciones y reacciones a mensajes): métodos
- ✓ Identidad (propiedad que lo restringe de los demás objetos)
- ✓ Concepto dinámico del objeto como una entidad que existe en tiempo de ejecución, ocupa memoria y tiene direcciones asociadas a él.

Un **objeto no es una clase**, sin embargo, una **clase puede ser un objeto.**

Los objetos contienen un conjunto de procedimientos compartidos llamados **métodos** → funciones que determinan el comportamiento de los objetos. Definen la forma en que los datos contenidos en los objetos son manipulados.

Los métodos de un objeto se invocan exclusivamente con el **mensaje adecuado**. Los objetos se comunican e interaccionen entre sí por medio de mensajes. Si un objeto desea que otro objeto ejecute un método le envía un mensaje que puede tener información adicional en forma de parámetros. (Un mensaje es una solicitud que se le pide a un objeto para que se comporte de una manera determinada)

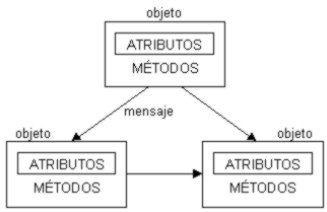
Un método tiene dos partes claramente diferenciadas:

- ✓ **la cabecera**-→ tipo de accesibilidad del método, tipo de valor a devolver, nombre del método, parámetros si es que los contiene y el tipo de excepción si el método puede lanzar excepciones
- ✓ **el cuerpo**→ conjunto de instrucciones que se ejecutarán cuando se invoque.

Métodos comunes: constructor; destructor; selector (get); modificador (set).

Los **objetos contienen atributos**. Éstos son los elementos que definen el estado de un objeto

- ✓ Hay 2 tipos: atributos de clase y atributos de instancia (o de objeto)



2.1.1. CRITERIOS PARA DEFINIR UNA CLASE

- ✓ Cuando aparezca un concepto nuevo, bien del problema a solucionar, bien de la solución del problema
- ✓ Cada vez que se requiera un tipo de datos nuevo, se diseña una clase con unos atributos que modelen el rango del tipo de datos y unos métodos que modelen las operaciones soportadas.

2.1.2. MÉTODOS

- ✓ **Análisis léxico** o nominal: Partiendo de un enunciado lo más preciso posible del problema, sobre él se estudian los nombres que aparecen. Los mismos constituyen posibles clases del modelo (parecen entidades/abstracciones existentes en el problema). Los verbos son operaciones candidatas de las clases con que aparecen o posibles relaciones de éstas con otras.
- ✓ Tarjetas CRC: También conocido por los nombres de 'programación por contrato' o diseño dirigido por responsabilidades, intenta definir claramente las responsabilidades (qué información posee, qué operaciones se le pueden solicitar) de cada clase, así como cuáles son las otras clases a las que puede pedir colaboración para cumplir su misión (Collaboration Responsibilities Cards)

2.2. PRINCIPIOS DEL MODELO ORIENTADO A OBJETOS

Son **identidad**, **abstracción, encapsulación, clasificación, modularidad, jerarquía, herencia**, **y polimorfismo** fundamentalmente, y en menor grado concurrencia, persistencia.

Principios SOLID:

- ✓ Single responsability: responsabilidad única contenida en la clase
- ✓ Open/closed: clases abiertas para extensión, cerradas para modificación
- ✓ Liskov substitution: objetos reemplazables por instancias de sus subtipos sin alterar el funcionamiento
- ✓ Interface segregation: interfaces cliente específicas
- ✓ Dependency inversión: depender de abstracciones, no de implementaciones. IoC (inversión de control) y DI (inyección de dependencias)

Identidad

- ✓ Un objeto puede ser cualquier cosa (un fichero, un proceso, …) pero tiene su propia identidad inherente, es decir, dos objetos con los mismos valores de sus atributos son dos objetos diferentes.
- ✓ El handle es el identificador único de cada objeto con el que puede ser referenciado unívocamente (su referencia).

Abstracción

- ✓ Consiste en la generalización conceptual del comportamiento de un determinado grupo de objetos y de sus atributos.
- ✓ Las clases abstractas dividen los problemas complejos en módulos sencillos y ocultan los detalles de la realización (**information hiding).).**
- ✓ La herencia mantiene automáticamente las relaciones entre las clases siguiendo la jerarquía de la biblioteca.

Encapsulación

- ✓ La encapsulación es el proceso de ocultar todos los detalles de un objeto que no contribuyen a sus características esenciales, es decir, separar el aspecto externo del objeto accesible por otros objetos, del aspecto interno del mismo que será inaccesible para los demás.
- ✓ **Los datos de un objeto solamente pueden ser manipulados vía los mensajes y métodos predefinidos**
- ✓ **La encapsulación es un mecanismo para conseguir la abstracción de datos**

Clasificación

Los objetos que tienen la misma estructura de datos (atributos) y el mismo comportamiento (operaciones), están agrupados en la misma clase

Modularidad

- ✓ es la propiedad de un sistema que ha sido descompuesto en un conjunto de módulos coherentes e independientes.

Herencia (es-un)

- ✓ La programación orientada a objetos introduce la posibilidad de extender clases, produciendo nuevas definiciones de clases que heredan todo el comportamiento y código de la clase extendida.
- ✓ La clase original se denomina clase padre, base o superclase. La nueva clase que se define como una extensión se denomina clase hija, derivada o subclase.
- ✓ Herencia simple si solo tiene una superclase
- ✓ La herencia es un mecanismo importante de extensión que permite la reutilización de código, pero tal extensión jamás provocará la modificación de la clase base.
- ✓ La relación de herencia es transitiva y define una jerarquía de herencia.
- ✓ La herencia puede ser simple o múltiple, según una clase herede los métodos y los datos de una sola clase o de más de una.
- ✓ La **herencia de implementación** implica que la clase hija hereda la implementación de métodos de la clase padre.
- ✓ La **herencia de interfaz** implica que la clase hija hereda el interfaz (pero no la implementación de las operaciones).
- ✓ La **herencia de clase** combina la herencia de implementación y de interfaz

Polimorfismo

- ✓ Es la propiedad por la cual un mismo mensaje puede originar conductas completamente diferentes al ser recibido por diferentes objetos.
- ✓ El polimorfismo está fomentado por el mecanismo de herencia.
- ✓ Las funciones de una clase padre o superclase pueden ser sustituidas en una subclase si se realiza una duplicación de su declaración en esta última clase hija.
- ✓ Es por ello que los objetos de las dos clases, tanto de la superclase como de la subclase, puedan reaccionar a los mismos mensajes, pero lo harán de diferentes maneras

También se puede dar el polimorfismo variando los parámetros que definen un método. Es decir, una clase puede tener dos o más métodos que se llaman igual, y tendrían o podrían tener comportamientos distintos. Este tipo de polimorfismo se denomina "**Sobrecarga**".

Las diferencias fundamentales entre la sobrecarga y el polimorfismo son:

- ✓ La sobrecarga es estática y se resuelve en tiempo de compilación, y el polimorfismo en general es dinámico y se resuelve en tiempo de ejecución.
- ✓ La sobrecarga es más eficiente (veloz) en tiempo de ejecución.
- ✓ La sobrecarga afecta a una única clase, y el "polimorfismo" a varias clases, normalmente relacionadas por una relación de herencia, aunque no en todas las situaciones es necesaria la herencia.

- ✓ Polimorfismo estático o ad-hoc (sobrecarga): es aquél en el que los tipos a los que se aplica el polimorfismo deben ser explícitos y declarados uno por uno antes de poder ser utilizados
- ✓ Polimorfismo dinámico (o polimorfismo paramétrico) es aquél en el que el código no incluye ningún tipo de especificación sobre el tipo de datos sobre el que se trabaja. Así, puede ser utilizado a todo tipo de datos compatible
    - Subtipos: Paramétrico, Subtipado Multitipo o Politipo (también conocido como de genericidad de tipo de datos)

Ligadura La ligadura se encarga de ligar o relacionar la llamada a un método con el cuerpo del método que se ejecuta finalmente.

- ✓ Ligadura estática: Consiste en realizar el proceso de ligadura en tiempo de compilación según el tipo declarado del objeto al que se manda el mensaje.
- ✓ Ligadura dinámica: Consiste en realizar el proceso de ligadura en tiempo de ejecución (ya que no puede resolverse en tiempo de compilación) siendo la forma dinámica del objeto la que determina la versión del método a ejecutar.

Reusabilidad

- ✓ Es la capacidad de producir componentes reutilizables para otros diseños o aplicaciones.
- ✓ Dos clases de reusabilidad:
    - la que comparte código escrito dentro de un proyecto, descubriendo secuencias de código redundantes en el diseño,
    - la que reutiliza código previamente escrito en nuevos proyectos.

Concurrencia

Es la propiedad que distingue un objeto que está activo de uno que no lo está.

Persistencia

- ✓ Es la propiedad de un objeto a través de la cual su existencia trasciende el tiempo (es decir, el objeto continúa existiendo después de que su creador ha dejado de existir) y/o el espacio (es decir, la localización del objeto se mueve del espacio de dirección en que fue creado). → EJB EntitiesBeans
- ✓ Uno de los problemas más habituales en programación es la gestión de la memoria y evitar que se produzcan "memory Leaks", o pérdida paulatina de memoria hasta que el equipo se queda sin recursos. → garbage collection

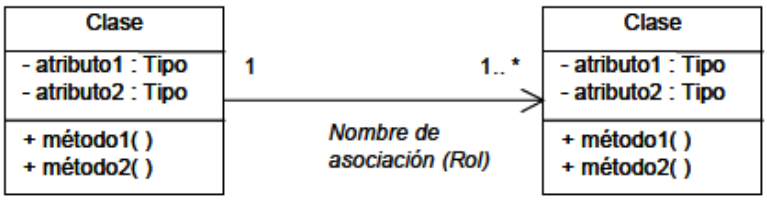
Extensibilidad

Es la capacidad de un programa o un sistema para ser fácilmente alterado de forma que pueda tratar con nuevas clases de entrada.

La herencia, el polimorfismo y la composición (fuerte, desaparición de clase compuesta implica supresión de componentes) y agregación (clase compuesto por otras, composición débil) son **mecanismo de extensibilidad del lenguaje.**

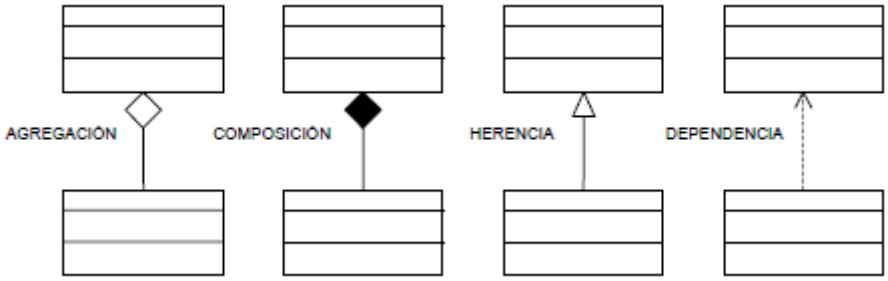
3. DIAGRAMA DE CLASES

Es un tipo de diagrama de estructura estática recogido en UML que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones, y las relaciones entre los objetos.

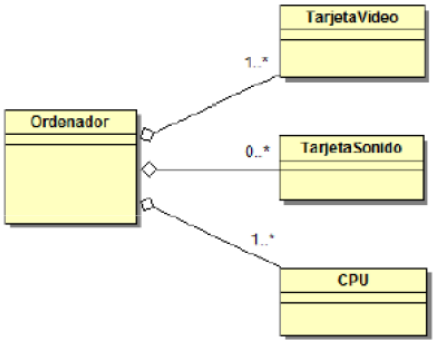


Elementos:

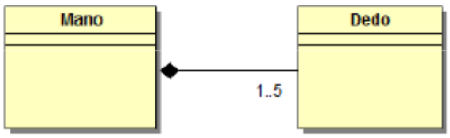
- Clases: elemento principal del diagrama para representar los conceptos o entidades.
- Relaciones: identifican una dependencia. Se representa mediante una línea que une las clases relacionadas y puede llevar nombre y multiplicidad o cardinalidad (número de elementos que participan en una relación).



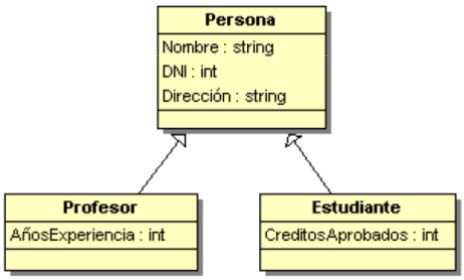
  - **Asociación**: tipo de relación más común, representa dependencia semántica y se dibuja con una línea simple.
  - **Agregación:** relación dinámica. Se representa mediante un diamante vacío colocado en el extremo de la clase "todo o base". El objeto base puede tener multiplicidad diferente de 1.



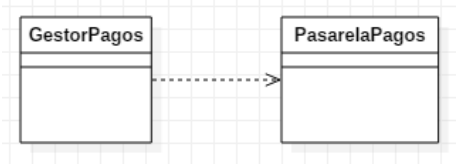
  - **Composición**: relación estática parte/todo. Se representa mediante un diamante sombreado. Objeto base con multiplicidad 1.



- **Herencia:** se representa con una flecha vacía que apunta a la clase padre.



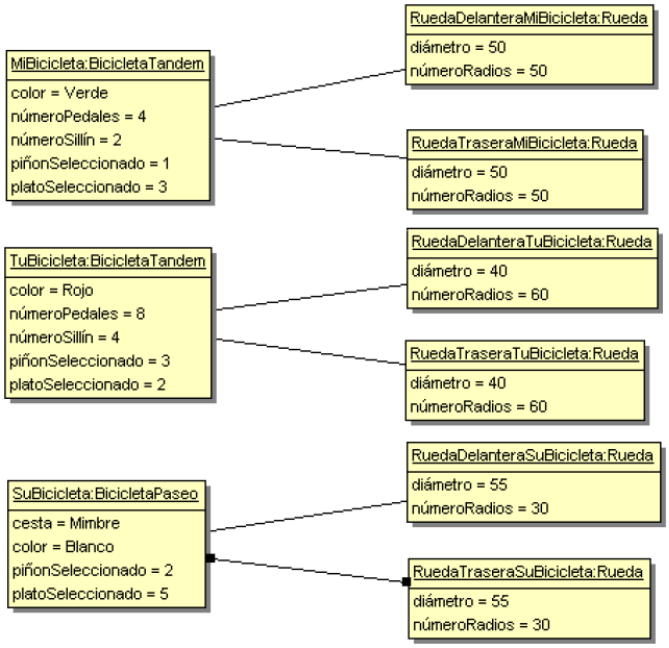
- **Dependencia:** una clase requiere de otra para ofrecer sus funcionalidades. Se representa con una flecha discontinua desde la clase que necesita la utilidad.



Más información en Doc Adicional y documento de Técnicas Métrica v3.

3.1. DIAGRAMA DE OBJETOS

Caso especial del Diagrama de Clases que muestra las instancias específicas de las clases involucradas en un instante determinado previamente decidido.



4. MODELO E/R

Modelo Entidad – Relación (E/R): Representa el **Modelo conceptual de datos**. Es único para cada sistema de información y se corresponde con el nivel conceptual de la Arquitectura ANSI a tres niveles.

- ✓ Tiene cierta correspondencia con el diagrama de clases (ciclo de vida orientado a objetos). Las clases contienen los métodos y los datos. En cambio, en el diagrama E/R sólo están los datos. En el DFD están los procesos.
- ✓ El diagrama E/R es un modelo que describe con un alto nivel de abstracción la distribución de datos almacenados en un sistema

- ✓ Enfatiza las relaciones entre almacenes de datos en el Diagrama de Flujo de Datos (DFD)
- ✓ El diagrama E/R representa problemas de "mente abierta", ya que es una aproximación del problema.

Diagrama de estructura de datos (DED): Representa el **Modelo lógico de datos**. Depende del Sistema Gestor de Base de Datos que se vaya a utilizar en el sistema y por tanto no es único. Se corresponde con el nivel lógico global o externo de la Arquitectura ANSI**.**

4.1. ENTIDADES

Una entidad es un objeto real o abstracto de interés en una organización y acerca del cual se puede y se quiere obtener una determinada información; personas, cosas, lugares, etc

ENTIDAD

- ✓ Tipo de entidad: estructura genérica
- ✓ Ocurrencia de entidad: realización concreta de una entidad

Características:

- ✓ Tiene que tener existencia propia.
- ✓ Cada ocurrencia de un tipo de entidad debe poder distinguirse de las demás.
- ✓ Todas las ocurrencias de un tipo de entidad deben tener los mismos tipos de características (atributos).
- ✓ Proceso iterativo

4.2. ATRIBUTOS

Cada una de las posibles propiedades o características que tiene un tipo de entidad. Son atómicos y el nº de atributos delimita al tipo de entidad.

Asociado al concepto de atributo surge el concepto de **dominio**. Un dominio es un conjunto nominado de valores homogéneos con independencia de cualquier entidad, relación o atributo. (recordar que en SQL estándar no existe el create domain)

Superclave a cualquier conjunto de atributos que permita distinguir a todas las entidades de cualquier instancia válida de un tipo de entidad.

Clave candidata es una superclave que no contiene ningún subconjunto que también sea superclave (Conjunto mínimo de atributos que forma una superclave). algunas de ellas pueden no servir por no darnos una única ocurrencia sobre la entidad

Clave primaria es una clave candidata seleccionada por el diseñador para distinguir entre las entidades de cada instancia. Dicho de otra forma, la clave principal es aquel atributo o conjunto de atributos cuyos valores identifican unívocamente cada entidad. La representación gráfica del atributo principal o clave es mediante una línea que lo subraye.

- ✓ En la PK debe regir el principio de unicidad, es decir los mínimos atributos posibles de la PK en el acceso. Para elegirlos deben ocupar la mínima capacidad de almacenamiento y la mínima demanda de procesamiento (principio de mínimos)

Tipos:

- ✓ Simple, son atributos que no están divididos en partes, es decir, representan un valor indivisible.
- ✓ Compuesto, atributo que puede ser subdividido en atributos más elementales.
- ✓ Univaluado, atributo que sólo puede tomar un valor para todas y cada una de las ocurrencias del tipo de entidad al que pertenece.
- ✓ Multivaluado ó multivalorado, atributo que puede tomar más de un valor para algunas de las ocurrencias del tipo de entidad al que pertenece

- ✓ Compuesto: Son aquellos que agrupan en sí mismos, por afinidad o por formo de uso, más de un atributo. Por ejemplo: Por su forma habitual de utilización, el atributo "dirección" engloba los atributos calle, numero, ciudad, provincia y código postal.
- ✓ Obligatorio, atributo que tiene que tomar al menos un valor para todas y cada una de las ocurrencias del tipo de entidad al que pertenece.
- ✓ Derivado, atributo cuyo valor se obtiene a partir de los valores de otros atributos de la misma o diferente tipo de entidad. Es un tipo de atributo que se debe evitar, en el pie del E/R se debe explicar el procedimiento que lo calcula y las restricciones.

(Atrib 1, Atrib 2 ……..)

4.3. RELACIONES O INTERRELACIONES

Una relación es básicamente una asociación entre entidades y se caracterizará por unas determinadas restricciones que determinarán qué entidades participan en la misma. Se representan a través de verbos que impliquen acción.

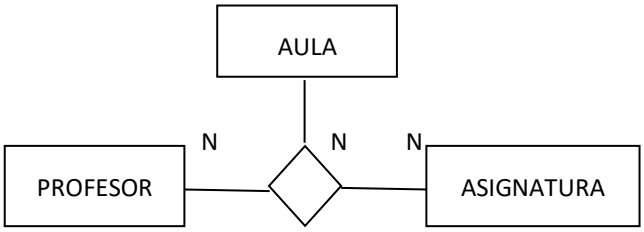
Las relaciones pueden tener atributos propios (PERO NUNCA DEBEN SER PK)

Grado: número de tipos de entidad sobre las que se establece la relación. En función del grado las relaciones se clasifican en:

i) UNITARIAS ó UNARIAS: Una entidad se relaciona consigo misma.

ii) BINARIAS: Entidades relacionadas dos a dos.

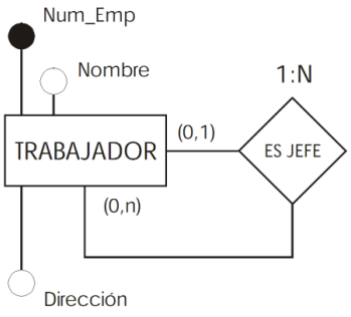
iii) TERNARIAS: Relación entre tres entidades / N-arias (entre N entidades). Para hallar el tipo de correspondencia se fijan 2 ocurrencias y se pregunta por la 3ª.



IV) REFLEXIVAS

Tipo de Correspondencia: Número máximo de ocurrencias de cada tipo de entidad que pueden intervenir en una ocurrencia del tipo de relación.

- ✓ **1:1** Una ocurrencia de la entidad A se asocia como máximo con una única ocurrencia de la entidad B y viceversa. Tienen poca carga semántica. Marcan un instante en el tiempo.
- ✓ **1:N** Una ocurrencia de la entidad A se asocia con un número indeterminado de ocurrencias de la entidad B, pero una ocurrencia de la entidad B se asocia como máximo con una ocurrencia de A
- ✓ **N:M** Una ocurrencia de la entidad A se asocia un número indeterminado de ocurrencias de la entidad B y viceversa
- ✓ **Reflexivas**: Son relaciones unarias y, por tanto, en este tipo de relación sólo participa un único tipo de entidad.
  - **Rol de la entidad:** dependiendo de si se pregunta de una manera u otra, existe el rol "ser jefe" y "empleado".
  - Ojo al transformarla a tabla hay que renombrar la PK (Num_jefe y Num_emp)



4.3.1. ENTIDADES DÉBILES

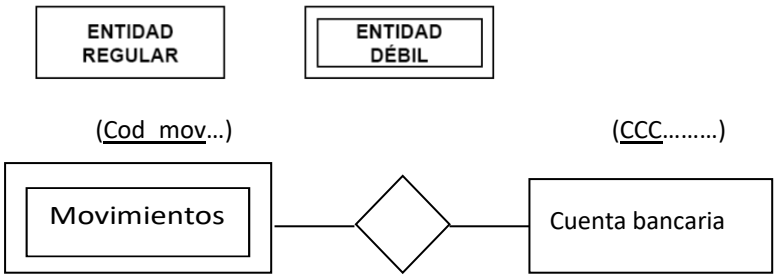
Son siempre binarias y su tipo de correspondencia es siempre 1:N.

- ✓ DEPENDENCIA POR IDENTIFICADOR. Este tipo especial de relación surge por la existencia de entidades que no tiene suficientes atributos para formar su clave primaria, es decir, la restricción de existencia con dependencia en identificación se produce cuando una entidad no es identificable por el valor de sus atributos, pero sí por su relación con otra entidad que aporta la parte de clave que le falta a la primera o entidad débil.
  - No garantiza el principio de unicidad en el acceso ya que es necesario incorporar la PK en la entidad fuerte.

- ✓ DEPENDENCIA POR EXISTENCIA. En este caso la entidad débil dispone de una clave primaria suficiente como para garantizar el acceso único a cada una de sus ocurrencias de entidad. Es la mejor solución porque me aseguro que la entidad débil está siempre con un PK válido

La dependencia en existencia no implica una dependencia en identificación, hecho que si sucede en el caso inverso.

En una interrelación con cardinalidad N:M nunca habrá entidades débiles



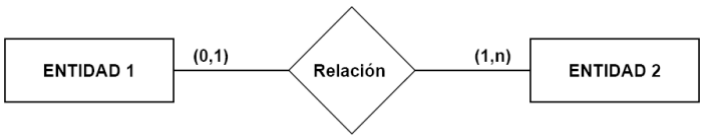
d.p.e / d.p.i

Si los movimientos de cuenta se secuencian globalmente, es una d.p.e, si se secuencian a partir de la cuenta corriente a las que están vinculadas con el cod_mov no es suficiente, se necesita la CCC también para formar la clave.

4.4. MODELO E/R EXTENDIDO

4.4.1. CARDINALIDAD

La cardinalidad de un tipo de Entidad se define como el número mínimo y máximo de ocurrencias de un tipo de entidad que pueden estar relacionadas con una ocurrencia del otro, u otros tipos de entidad que participan en el tipo de relación



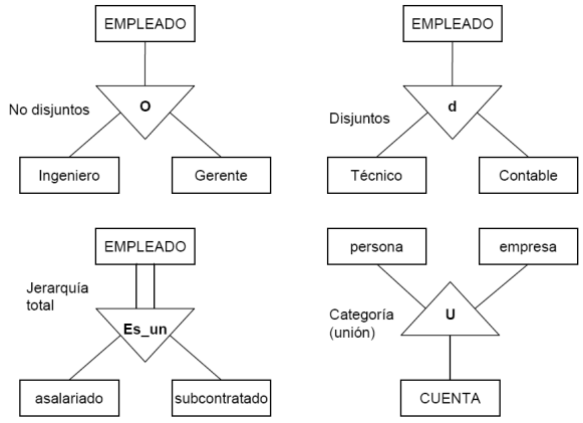
También denominado “clase de pertenencia”, permite especificar si todas las ocurrencias de una entidad participan o no en la relación establecida con otra(s) entidad(es):

- ✓ Obligatoria: Si toda ocurrencia de la entidad A debe estar asociada con al menos una ocurrencia de la entidad B a la que está asociada por una determinada interrelación, la cardinalidad mínima es 1.
- ✓ Opcional: Por el contrario, si no toda ocurrencia de la entidad A necesita estar asociada con alguna ocurrencia de la entidad B asociada, la cardinalidad mínima es 0.

4.4.2. RELACIONES DE JERARQUÍA

- ✓ **Generalización** De lo concreto a lo abstracto, método inductivo, va de lo particular a lo general. El supertipo incluye atributos de los subtipos.
- ✓ **Especialización** De lo general a lo particular, método deductivo. Se da cuando no se representan todos los subtipos posibles (p. ej un empleado puede ser técnico o gerente, puede haber más tipos de empleados pero no se representan)

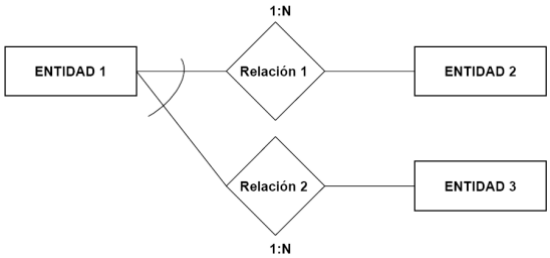
JERARQUÍA DE GENERALIZACIÓN/ESPECIALIZACIÓN



Un **subconjunto** es un caso particular de generalización con una sola entidad como subentidad. Un subconjunto siempre es una **jerarquía parcial y exclusiva**

Cada jerarquía es total o parcial, y exclusiva o superpuesta.

- ✓ Jerarquías (en general): Cada subentidad "hereda" todas las relaciones y atributos de la superentidad
- ✓ Jerarquías según cobertura.
  - Jerarquía con Cobertura Total. Cada elemento de la superentidad debe estar en al menos una de las subentidades.
  - Jerarquía con Cobertura Parcial. Puede haber elementos de la superentidad que no estén en ninguna de las subentidades.
- ✓ Jerarquías según solapamiento
  - Jerarquía Disjunta. Cada elemento de la superentidad está a lo sumo en una subentidad
  - Jerarquía con Solapamiento (no disjunto). Los elementos de la superentidad pueden estar en más de una subentidad.
- ✓ **Relaciones exclusivas:** Decimos que dos o más tipos de relación son exclusivos cuando cada ocurrencia de un tipo de entidad sólo puede pertenecer a un tipo de relación**.**



4.4.3. CATEGORÍAS

Parecido a la herencia múltiple.

Se denomina categoría al subtipo que aparece como resultado de la unión de varios tipos de entidad.

En este caso, hay varios supertipos y un sólo subtipo

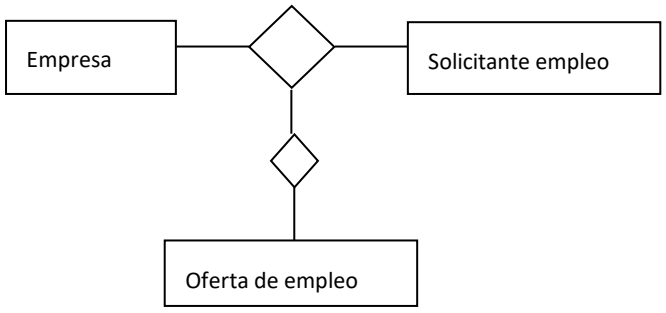
4.4.4. ASOCIACIONES

Consiste en relacionar dos tipos de entidades que normalmente son de dominios independientes, pero coyunturalmente se asocian

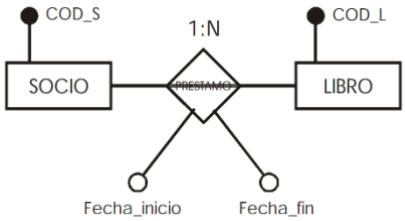
4.4.5. AGREGACIÓN

Consiste en construir un nuevo tipo de entidad como composición de otros y su tipo de relación y así poder manejarlo en un nivel de abstracción mayor.

Por ejemplo, se tienen los tipos de entidad empresa y solicitante de empleo relacionados mediante el tipo de relación entrevista; pero es necesario que cada entrevista se corresponda con una determinada oferta de empleo. Como no se permite la relación entre tipos de relación, se puede crear un tipo de entidad compuesto por empresa, entrevista y solicitante de empleo y relacionarla con el tipo de entidad oferta de empleo. El proceso inverso se denomina desagregación.



4.4.6. DIMENSIÓN TEMPORAL

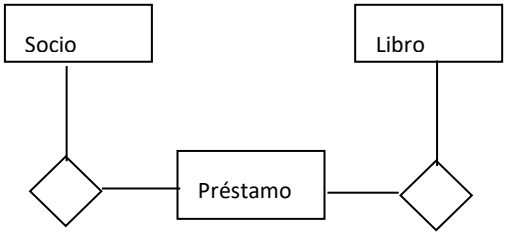


SOCIO (Cod_socio,……)
LIBRO (Cod_libro,…….Cod_socio (FK), fecha_inicio, fecha_fin)

1 libro puede ser prestado por el mismo socio varias veces, por lo tanto hay grupos repetitivos

Si ponemos la correspondencia a N:M
SOCIO (Cod_socio…)
LIBRO (Cod_libro….)
PRESTAMO (Cod_socio, Cod_libro, Fecha_ini, Fecha_fin) → se repite la PK
→Socio1-Libro1

Solución



(Cod_prestamo, Fecha_ini, Fecha_fin)

4.4.7. TRANSFORMACIÓN A TABLAS

- ✓ 1:N se propaga el id de la entidad de <u>cardinalidad máxima 1 a la que es N</u>
  - Si la relación es de asociación, la clave propagada es clave ajena en la tabla que se ha propagado
  - Si la relación es de dependencia, la PK de la entidad débil estará formada por la concatenación de los id de ambas entidades

- ✓ 1:1 es un caso particular de las 1:N y por tanto se propaga la clave en las dos direcciones

4.5. MODELO E/R EN MÉTRICA

Las ventajas de realizar un modelo de datos son, entre otras:

- ✓ Comprensión de los datos de una organización y del funcionamiento de la organización.
- ✓ Obtención de estructuras de datos independientes del entorno físico.
- ✓ Control de los posibles errores desde el principio, o al menos, darse cuenta de las deficiencias lo antes posible.
- ✓ Mejora del mantenimiento.

Este diagrama se centra en los datos, independientemente del procesamiento que los transforma y sin entrar en consideraciones de eficiencia. Por ello, es independiente del entorno físico y debe ser una fiel representación del sistema de información objeto del estudio, proporcionando a los usuarios toda la información que necesiten y en la forma en que la necesiten.

Más información en Doc Adicional y documento de Técnicas Métrica v3.