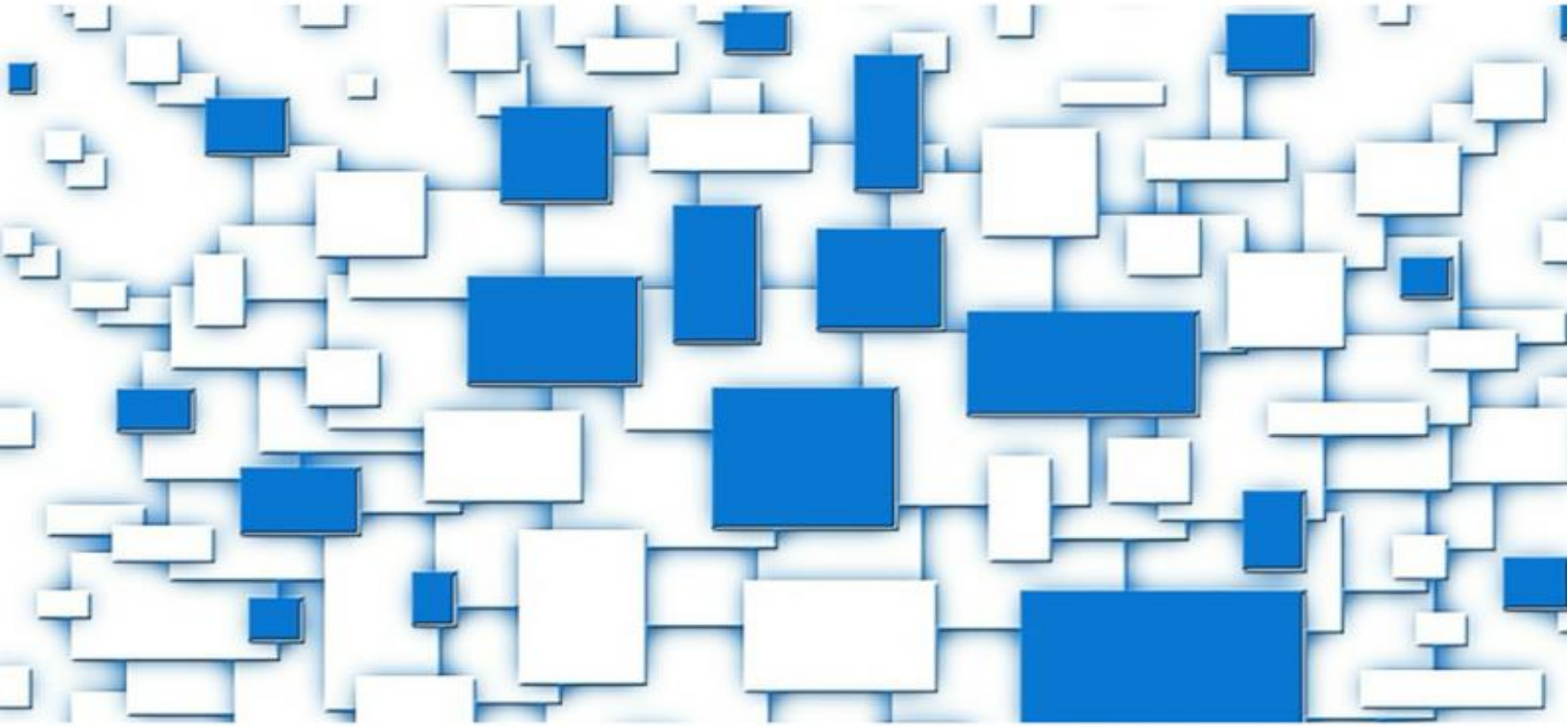
Hermosa Programación

METODOLOGÍA PARA EL DISEÑO CONCEPTUAL DE BASES DE DATOS



8 PASOS PARA NO COMPLICARSE

PROLOGO

El **diseño de base de datos** es una de las fases más importantes de una metodología de desarrollo de Software.

Para el diseñador es un gran desafío resumir e integrar todos los requerimientos del sistema en una vista conceptual que todo el equipo y cliente puedan entender.

Es por eso que es tan importante elegir las mejores prácticas para expresar forma objetiva y adaptable la realidad percibida.

Metodología Para Diseñar Bases De Datos

El diseño de base de datos generalmente está compuesto por tres etapas segmentadas por complejidad.

En primera instancia está el **Diseño Conceptual**, cuyo fin es representar modelos mentales del negocio en su extensión más general. En esta etapa se interpretan las necesidades del proyecto y se transforman en un modelo de datos que sea independiente de todo tipo de consideraciones físicas (Modelo Entidad Relación, diccionario de datos, validación de transacciones).

Luego sigue el **Diseño Lógico**, donde se transforma el modelo de datos conceptual en un modelo de datos lógico (modelo de datos relacional), con el fin de acercar a una definición relacional que satisfaga las transacciones de la base de datos (Normalización, restricciones de integridad, reglas del negocio, etc.).

La tercera etapa es el **Diseño Físico**. Aquí se traduce todo el modelo lógico a una versión preliminar a la implementación de la base de datos en SQL, independientemente del motor de base datos a usar. Normalmente se realizan tareas como definir la estructura de archivos, la nomenclatura a usar, la declaración de las consultas más importantes en papel, el manejo de índices, etc.

Modelo Entidad-Relación

El **Modelo Entidad-Relación** (frecuentemente abreviado *Modelo ER*) es un enfoque de abstracción usado para la comprensión de la vista conceptual de un

proyecto. No importa si usas un **ciclo de vida en cascada** o una **planeación ágil**. En ambos casos funciona.

Con el podemos identificar desde un alto nivel, aquellas entidades que participan en el proceso que se desea comprender, analizando las características y relaciones de cada una.

Para ello se usan diagramas y documentos que organicen las ideas para su interpretación global.

Precisamente en la primera parte veremos cómo usar el **Diagrama Entidad-Relación** para representar el modelo de datos con el fin de facilitar visualmente el dominio del negocio o necesidades del usuario.

Aunque existen una gran variedad de notaciones para crear diagramas que representen el modelo de datos, en este ebook usaremos la notación UML (*Unified Modeling Language*) o *Lenguaje de Modelado Unificado*, el cual nos permite acercar la estructura de la base de datos a un **enfoque orientado a objetos**.

Esta notación permitirá generar un modelo conceptual y uno lógico sin tener que crear diagramas diferentes.

No obstante también existen notaciones como la **Crow's Foot**, la cual es muy usada para proyectar el estado final del diseño lógico de una base de datos. Su propósito es representar la base de datos transformada y lista para implementarla al código.

> Puedes ver su notación en el archivo "*Notación Crow's Foot.png*"

Otra notación conocida es la **Chen**, cuyo propósito es el diseño conceptual, es decir, puede ser implementado en la primera fase para levantar los datos generales, pero se aleja un poco de la implementación.

> Puedes ver su notación en el archivo "*Notación CHEN.png*"

Plantillas Y Checklists

Para la aplicación del contenido de este ebook he creado contenidos extras como lo son plantillas, checklist y formatos donde puedes practicar el uso de las instrucciones.

Revisa la carpeta *Herramientas* para acceder a estos recursos. A lo largo del temario verás anotaciones que te indicarán el archivo a estudiar.

Errores

Para ***Hermosa Programación*** es importante mejorar hasta el máximo los contenidos que produce para sus seguidores. Por esa razón apreciaría que me contases sobre cualquier error, sugerencia de redacción o anomalía que encuentres al leer.

Este ebook tuvo varios filtros de corrección, pero como sabes, somos humanos y cualquier cosa se nos puede pasar por alto.

Envía tus comentarios correo *jamesreveu@gmail.com*. *¡Gracias!*

CONTENIDO

PARTE 1
EL MODELO ENTIDAD-RELACION

INTRODUCCIÓN

Como se venía hablando, el modelo entidad relación surge como el estudio formal de las características de la información que debe ser almacenada en una base de datos.

Este nos permite modelar de forma general con el fin de interpretar correctamente la situación que deseamos solucionar.

Para demostrar la construcción de un modelo de datos, se ha creado un caso de estudio que nos permitirá comprender el temario basándonos en una situación apegada a la realidad. El cual seguiremos a lo largo de todo el ebook con el fin de ejemplificar todos los conceptos.

Luego de ello veremos en detalle cada uno de los componentes de un modelo entidad-relación y como son representados en el diagrama correspondiente con la notación UML.

Partiremos definiendo el propósito de las entidades, las cuales representan a cada uno de los objetos que nos interesa estudiar en un negocio o situación. En seguida analizaremos como se relacionan con otras entidades y que atributos poseen.

Por último se estudiará cómo afrontar la clasificación de las entidades en superclases y subclases, un concepto muy usado en la *Programación Orientada a Objetos* (POO). Veremos cómo este enfoque puede refinar nuestros modelos para mejorar su organización.

CASO DE ESTUDIO

El *Instituto Educativo San Judas* es un centro de educación técnica fundado en el año 2000.

Inició con tan solo 40 estudiantes en sus instalaciones y las facultades de *Tecnologías de la información* y *Mercadeo*.

Desde entonces ha tenido un crecimiento enorme, abriendo otras 8 facultades más y contando con aproximadamente 350 estudiantes activos.

Debido a este crecimiento, el personal administrativo y el cuerpo docente se ha incrementado proporcionalmente. Con ello los procesos de gestión de la información se han tornado en un flujo de papelería tediosa e innecesaria. Existen formatos físicos para diversas actividades, lo que hace retardar las actividades de registro y apertura de cursos.

Su Director Administrativo *Carlos Cifuentes*, está preocupado por las complicaciones que se están generando con todos los trámites que se deben llevar a cabo.

El cree que la solución está en la implementación de un sistema de información basado en bases de datos, ya que ha escuchado los increíbles resultados que han obtenidos los institutos cercanos con este modelo de operación.

Al contratar nuestro servicio, Carlos nos describe la forma en que funciona el instituto San Judas y como fluye la información:

> *"En primera instancia llega el estudiante con toda su documentación al área de Registro y Control, donde se reciben sus documentos y se le autoriza el pago del semestre actual, entregándole un recibo de pago.*
>
> *Luego de pagar se matricula financieramente y académicamente.*
>
> *La matrícula académica se gestiona en la faculta a la que pertenece el nuevo alumno, aquí se le entrega información detallada sobre los horarios diurnos y nocturnos de los cursos que puede ver , además*

de la cantidad de créditos que dispone para la configuración de su horario.

Luego se revisa el plan académico del estudiante para asegurarse de que ha cumplido los prerrequisitos de cada curso incluido en su matrícula académica.

Los horarios son creados de acuerdo a la disponibilidad de los instructores que pertenecen a la facultad. Cada instructor tiene un horario de trabajo registrado en los informes de actividades que nos deben entregar.

En cuanto las evaluaciones, actualmente contamos con tres cortes de evaluación, cuya ponderación se divide en 30, 30 y 40 por ciento respectivamente.

Cada profesor carga una lista de estudiantes con sus notas asociadas, la cual se entrega diligenciada al final de cada corte en la facultad correspondiente."

Luego nos comenta sus objetivos con la implementación del Software:

> *"Necesito que el proceso completo de Registro de Estudiantes, Matrícula Académica, Gestión de cursos y la configuración de horario de nuestros profesores sea implementada digitalmente. Creo que esto agilizará los procesos institucionales y pondrá en segunda plano los aspectos que nos complican las operaciones"*

Luego de ello Carlos nos entrega *diagramas de procesos*, *formatos de inscripción*, *planillas de cursos*, *reportes de estudiantes* y toda la documentación necesaria para identificar lo que vamos a sistematizar.

Adicionalmente se realizan una serie de reuniones con el equipo desarrollador y las personas interesadas, para determinar qué características debe tener el software.

Al finalizar esta exploración se obtienen los siguientes ítems:

Requerimientos del Software

Facultades

El Instituto *San Judas* divide a su personal por saberes en facultades, por ejemplo la facultad de *Ingeniería*. Se requiere administrar los datos de cada facultad como lo son su *identificador de facultad*, su *nombre*, la *oficina* donde está ubicada en las instalaciones, el *nombre de la secretaria* que brinda el servicio de registro y el *número* para comunicarse vía telefónica.

Instructores

Es necesario almacenar la información de todos los instructores para optimizar la consulta de sus datos, averiguar su disponibilidad y la cantidad de cursos que tienen asignados.

Los datos más importantes de un instructor son: el *identificador* brindado por la instituto, sus *nombres*, *apellidos*, *dirección*, un *teléfono principal* para ubicarlo y máximo *2 números alternativos*. También se desea almacenar la *oficina* en que se ubicó y su *salario*.

Cabe destacar que de todos los instructores de una facultad, se elige uno para coordinar las actividades de esta. Adicionalmente debe supervisar el trabajo de sus compañeros.

Se debe considerar que a cada instructor se le pueden asignar varios cursos dependiendo de su horario semanal.

El salario del instructor puede variar dependiendo de su desempeño, por lo que se necesita realizar un seguimiento a su variación.

Estudiantes

Para los estudiantes es necesario identificar a que cursos están inscritos para crear un historial académico de sus rendimientos.

Los datos de los estudiantes a almacenar son muy parecidos a los de los instructores. Estos son: el *identificador* que se le asigna una vez haga parte de la institución, los datos de sus *nombres*, *apellidos*, un *número telefónico principal* (como máximo 3 números telefónicos) y la *dirección* de residencia.

Por cada curso en que se encuentre el estudiante, deben guardarse todas las calificaciones que el profesor le asigne por su desempeño evaluado.

Cursos
Por cada curso se requiere almacenar el *código institucional* asignado, el *nombre* del curso, la *descripción* del curso y los *créditos* que tiene. Adicionalmente se necesita saber la fecha en la que dio *inicio al curso*, el *semestre* en que se presentó, el *horario* que tuvo cuando se dictó junto al *salón* donde se llevó a cabo.

Debe notificarse también que cursos son prerrequisito de otros para evitar que los estudiantes sigan un flujo incorrecto de su carrera. También se debe saber si un curso es presencial o es online, o si está creado para las dos modalidades.

Consultas Identificadas

1. Obtener una lista todos los cursos vistos por un estudiante.
2. Generar el horario de clases para cada estudiante.
3. Generar el horario de los instructores.
4. Generar listado de todos los estudiantes en un curso y su cantidad total.
5. Generar una lista del historial de notas (historial académico) de un estudiante.
6. Calcular la cantidad de créditos matriculados por un estudiante.
7. Calcular la cantidad de estudiantes por facultad.
8. Generar una lista de los instructores a cargo de un supervisor.
9. Listar los cursos menos aprobados por los estudiantes.
10. Calcular cuántos cursos online existen.
11. Listar los estudiantes que tienen que recuperar materias.

Visión general

El objetivo de este caso de estudio no es abordar temas de ingeniería de requerimientos ni el análisis de personas. Solo hemos usado un contexto simple acomodado a la vida real para tener una guía ilustrativa y seguir una secuencia clara para los temas que se desarrollarán a lo largo de todo el temario.

Como ves no se han usado formatos especiales de especificación de requerimientos o para la creación de historias de usuario. Tampoco se ha definido

la arquitectura del sistema, ni la experiencia de usuario, o el sistema de roles, ni se determinaron las vistas de usuario.

El fin de este ebook es especializarse simplemente en el **diseño conceptual de una base de datos** y no dejar temas inconclusos.

El Modelo ER del Instituto San Judas diagramado
A continuación puedes ver el modelo de datos representado con UML que resulta del caso de estudio:

> Si deseas verlo aisladamente busca el archivo *"Diagrama ER San Judas.png"*

Puedes consultarlo frecuentemente para contextualizar los temas que trataremos en la primera parte.

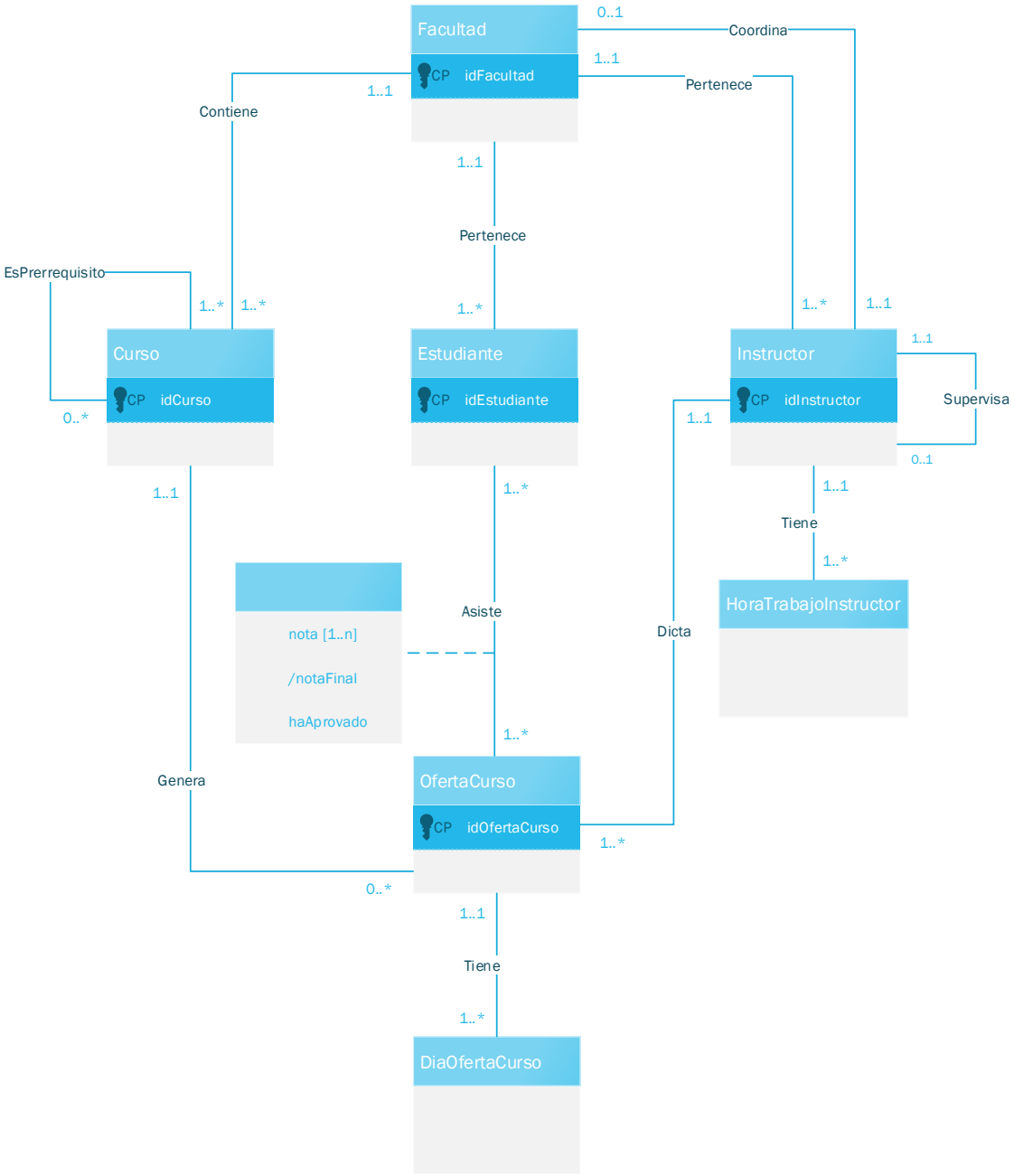


Figura 1 *Diagrama E-R final del Instituto San Judas*

1. ENTIDADES

La base de los modelos de datos es la **entidad** (Este concepto es análogo a las **clases** en POO). Una entidad es la representación general de un conjunto de objetos, ya sea físicos o abstractos, que se caracterizan por tener los mismos atributos y pertenecer a una misma familia.

Por lo general el cliente manifiesta qué entidades desea almacenar con facilidad para tener un registro de estas. Sin embargo cuando entiendas los procesos de la empresa encontrarás entidades ocultas necesarias para la derivación de información.

A cada elemento único perteneciente a la entidad se le llama **instancia**, es decir, que el estudiante *Francisco Maldonado* es una instancia de la entidad Estudiante, ya que se habla del objeto en particular como un **sustantivo propio** como se ve en la Figura 1.

Figura 2 *Ejemplo de instancias que componen la entidad Estudiante*

En el caso del colegio San Judas se pueden identificar varios ejemplos, como Estudiante, Instructor, Curso, Horario, etc. Pero también pueden ser entidades que no tengan forma física, como Tarea, Evaluación, Estadística, Venta, etc.

Para representar una entidad en notación UML se usa un rectángulo con el nombre de la entidad en singular. La convención para la escritura del nombre es usar la primera letra en mayúscula, y una letra capital por cada palabra que siga.

Veamos el ejemplo de la entidad Estudiante:

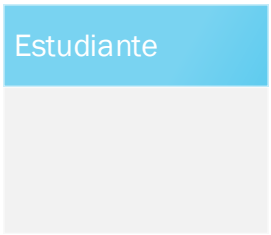


Figura 3 *Notación UML para la entidad Estudiante*

2. RELACIONES

Una relación es un conjunto de asociaciones entre dos o más entidades. Por lo que se entiende que las instancias de una entidad se relacionan con las instancias de la otra. Este vínculo se puede basar en cualquier tipo de afección, cálculo, dependencia, etc. Es decir, todo lo que se refiera a un verbo.

Por ejemplo, *a una facultad pertenecen uno o varios instructores*. En este caso el verbo **pertenecer** es identifica una relación entre las entidades Facultad e Instructor.

Al igual que con las entidades, una relación tiene instancias para darle trato particular a los elementos. Veamos una representación en redes semánticas entre las instancias de las facultades y los instructores:

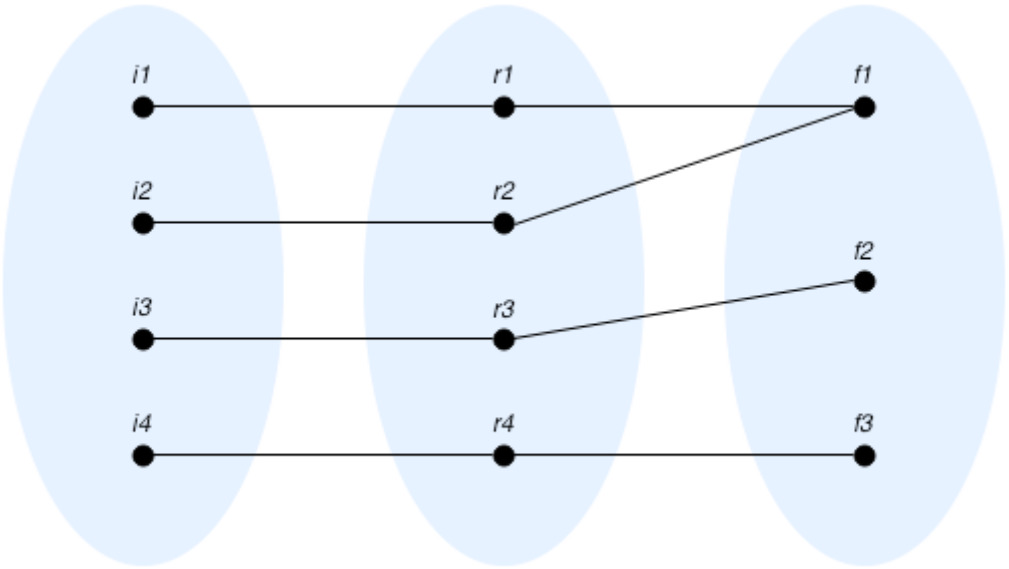


Figura 4 *Ejemplo de Redes Semánticas*

Los elementos denominados con la letra i representan instancias de la entidad Instructor, los elementos del tipo r representan una instancia de relación y los elementos f representan a las facultades.

Matemáticamente podríamos decir que una **instancia de relación** para el caso anterior se representa con el par r= { i , f }.

Generalmente se sigue la misma nomenclatura para el nombramiento que se usó con las entidades. Solo que usaremos como nombre un verbo en presente simple.

Opcionalmente puedes usar una flecha que indique la dirección desde donde se inicia la relación.

Veamos como representar la relación anteriormente descrita:

Figura 5 *Representación UML de una Relación*

2.1 Grado de una Relación

El grado de una relación es el número de entidades que participan en la relación. Por ejemplo, la relación anterior es de tipo **binaria**, ya que solo participan dos entidades.

Si participasen tres entidades en la relación, entonces sería una relación **ternaria**, si participan cuatro sería **cuaternaria** y si participan muchas más se generaliza con la notación **n-aria**.

Las relaciones que tienen grado mayor a 2 son llamadas también **relaciones complejas**. Un ejemplo podría ser la relación: *Un instructor dicta una de un curso a un estudiante*.

La notación UML para representar las relaciones complejas se realiza con un diamante conector, del cual desprenden líneas desde sus extremos hacia las entidades que intervienen en la relación. El nombre de la relación se ubica en el interior del diamante.

Veamos:

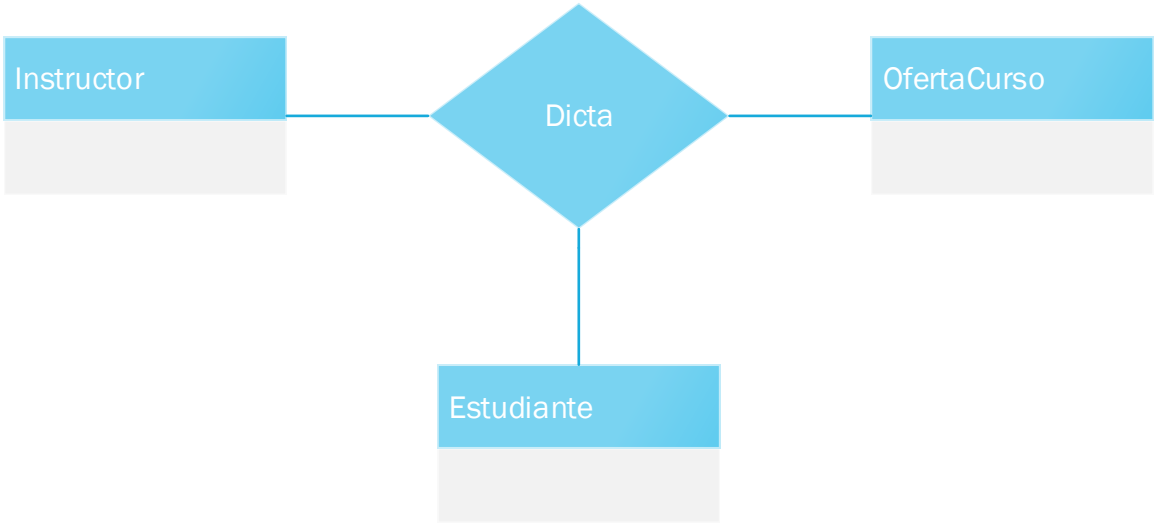


Figura 6 *Relación Compleja entre Instructor, OfertaCurso y Estudiante con UML*

2.2 ¿Cómo identificar si es pertinente usar una relación compleja?

Esta decisión depende de las vistas de los usuarios, los reportes que necesitan los clientes y de si la información no es posible obtenerla desde una relación binaria. Por ejemplo, el diagrama anterior se observa que un instructor puede impartir una oferta de un curso a un estudiante.

Si la *Institución San Judas* expresa explícitamente que desea generar un listado de los contactos presenciales que ha tenido un profesor con los estudiantes de un curso, entonces la relación ternaria tendría cabida en el modelo de datos.

De lo contrario el diseñador de la base de datos puede asumir que en realidad no existe una relación directa entre un instructor y un estudiante, ya que ambos interactúan por medio de la entidad curso. Esto permite modelar solo relaciones binarias entre Estudiante y Curso e Instructor y Curso.

2.3 Relaciones Recursivas

Una relación **recursiva** (también llamada **unaria**) es aquella donde una entidad participa consigo misma, es decir, sus instancia se relacionan entre sí.

Por ejemplo, en el *Instituto San Judas* a los instructores se les asigna un coordinador, el cual debe ser uno de sus mismos compañeros de facultad. Lo quiere decir que existe un instructor “especial” ante las demás instancias.

Pero… *¿Cómo modelar dicha situación?*

Sencillo, se deben establecer dos **roles** que diferencien a las instancias de la entidad. Uno para los coordinadores y otro para los que son coordinados.

Gráficamente las relaciones recursivas se representan con una línea de asociación, cuyos extremos están conectados a la misma entidad. Adicionalmente se debe ubicar el rol de la entidad dependiendo de la dirección de la relación.

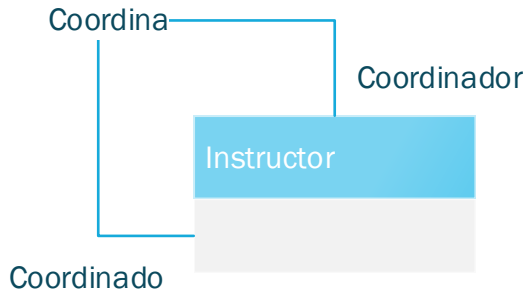


***Figura** 7 Ejemplo de una Relación Recursiva en Instructor*

Como ves en la ilustración la relación indica que *"Un instructor (Coordinador) coordina/supervisa a otros instructores (Coordinados)"*.

2.4 Más de una Relación entre dos Entidades

Los roles también pueden usarse cuando existen varias relaciones entre dos entidades. Por ejemplo, el coordinador de los instructores no solo tiene la responsabilidad de guiarlos si no también de asumir toda el área administrativa de la facultad.

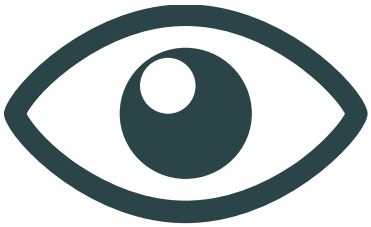
Esto significa existe otra relación de Instructor y Facultad que tiene como propósito indicar el coordinador de la facultad.

Pero modelar esto es sencillo, simplemente usamos otra relación pero estableciendo roles de coordinador e instructor:

Figura 8 *Dos relaciones entre las Entidades Facultad y Coordinador*

Ubicamos solo el rol de Coordinador, ya que los demás roles se mantienen igual.

3. ATRIBUTOS



Los atributos son propiedades de las entidades. Un ejemplo claro serían el nombres, apellidos, fecha de nacimiento y teléfono de un estudiante. Cada uno de estos datos es considerado un atributo, los cuales son el insumo necesario para realizar análisis y gestionar ocurrencias en los sistemas de información.

3.1 Dominio de un Atributo

¿Crees que es permitido agregar números al nombre de un estudiante? Obviamente no es bien visto un modelo que permita este tipo de situaciones irregulares, sin embargo los diseñadores y programadores pueden dar sorpresas.

Para prevenir estas anomalías, es necesario hacer énfasis en el **dominio de los atributos** que tendrán las entidades de tu base de datos.

El dominio comprende el conjunto de aquellos valores que son permitidos en un atributo. Si recuerdas cuando veías conjuntos o funciones matemáticas, ya sabrás a que me refiero.

Por ejemplo, el atributo sexo de la entidad Estudiante, tiene un dominio al cual solo pertenecen los valores '**F**' (Femenino) y '**M**' (Masculino), así que debe restringirse al atributo para que acate esa indicación.

Incluso podemos expresar el dominio a través de la teoría de conjuntos. La siguiente notación por extensión establece que el atributo salario de un instructor debe ser mayor a 2000USD:

$$D = \{\, salario | salario \in N \ \wedge salario \geq 2000 \,\}$$

El anterior dominio se describe como *todos los valores de salario que pertenezcan a los números naturales y sean mayor o igual a 2000*.

3.2 Atributos Atómicos y Compuestos

Un **atributo atómico** o **simple** es aquel que no puede ser divisible en más atributos y su existencia es independiente de toda restricción. Como por ejemplo el email y teléfono de la entidad Estudiante.

Un **atributo compuesto**, por el contario, es la composición de dos o más atributos atómicos. Un buen ejemplo sería la composición del *nombre* de una persona por su *primer nombre*, *segundo nombre*, *apellido paterno* y *apellido materno*.

La selección de la complejidad del atributo depende de los requerimientos del usuario.

3.3 Atributos Monovalorados y Multivariados

Los **atributos monovalorados** o de un solo valor son aquellos que solo tienen asignados un solo valor por cada instancia de la entidad. Por ejemplo, existe un solo valor para el atributo creditos de Curso, ya que no aplican variaciones.

Ahora, un **atributo multivariado** es aquel que puede recibir varios valores por cada instancia. Puedes detectarlos fácilmente al percibir que un atributo contiene una lista de varios valores.

Normalmente se encuentran en un rango considerado entre un valor *mínimo* y *máximo*. Un atributo multivariado muy popular es el número telefónico, por ejemplo, el instructor IA04 se le puede contactar al número 3451234 y al 4562213.

Aunque este instructor tiene dos números, recuerda que en la definición del caso de estudio se determinó que debe existir como mínimo un teléfono principal y máximo 3.

3.4 Atributos Derivados

Un **atributo derivado** es aquel que su valor se genera con respecto al valor de otros atributos. Esto significa que se el valor se puede originar de un cálculo matemático o una asignación condicionada de valores de otros atributos de la entidad u otras entidades.

Por ejemplo, la notaFinal de un estudiante es calculada a partir de las notas de los distintos cortes evaluados.

También se puede considerar un atributo derivado la cantidad de instancias que tiene la entidad Facultad. O por ejemplo la cantidad total en ventas de un vendedor.

3.5 Llaves

3.5.1 Llave Candidata

Una **llave candidata** es un atributo que puede llegar a comportarse como un identificador único de cada instancia de una entidad.

Por ejemplo, el atributo idEstudiante asignado por el instituto es una buena opción para diferenciar a las instancias de Estudiante, por lo que es una llave candidata.

Además cada estudiante como ciudadano de un país tiene asignada una identificación personal, por lo que este posible atributo también es una llave candidata.

3.5.2 Llave Primaria o Clave Primaria

Como ves, en una entidad pueden existir varias llaves candidatas, sin embargo debe elegirse una como la **llave primaria** o **principal**.

El objetivo es asegurar que cada instancia de una entidad se diferencie de las demás. Esta característica permite que las relaciones entre entidades se representen con claridad, evitando la informalidad.

Puede que por el momento existan entidades que no tengan definida su llave primaria completamente debido a que son entidades débiles (veremos este término en el siguiente apartado), así que no te preocupes por esa circunstancia.

3.5.3 Llave compuesta

No siempre una llave primaria está compuesta de un solo atributo. En ocasiones una sola llave primaria no es suficiente para asegurar la unicidad entre las instancias de una entidad.

En esos casos se implementa una **llave compuesta**, la cual se configura a partir de dos o más atributos llamados también **llaves primarias parciales**.

Cabe destacar que una llave compuesta puede generarse partir de llaves candidatas, atributos que no sean llaves candidatas o una combinación de ambos.

Un excelente ejemplo de llave compuesta se da en la entidad HoraTrabajoInstructor, la cual representa los horarios de trabajo de cada instructor.

Tiene los atributos diaSemana, horaInicio y horaFin. Pero individualmente ninguno es una llave candidata. Incluso, los tres juntos tampoco representarían una llave candidata.

Cuando veas el *Modelo Relacional* (Diseño lógico), verás que esta entidad recibe una copia de la llave primaria de Instructor como llave foránea y que su llave primaria es una llave compuesta entre idInstructor, diaSemana, horaInicio y horaFin.

Esta se representaría como el siguiente conjunto:

$$CP = \{idInstructor, diaSemana, horaInicio, horaFin\}$$

3.6 Representación de atributos en UML

Para diagramar los atributos relacionados a las entidades, dividiremos el rectángulo en dos partes. En la parte superior incluimos el nombre de la entidad y en la parte inferior la lista de atributos.

Dependiendo del tipo de atributo que sea usaremos las siguientes notaciones:

Tipo de Atributo	Notación
Llave Primaria	**{PK}** ó **{CP}**
Llave Primaria Parcial	**{PPK}** ó **{CPP}**
Atributo Compuesto	Una tabulación a la derecha
Atributo Multivalorado	Incluir rango entre llaves **[min…max]**
Atributo Derivado	Anteceder una barra inclinada **'/'**

Tabla 1 *Notación UML para atributos*

Veamos cómo hacerlo para la relación entre Instructor y Facultad:

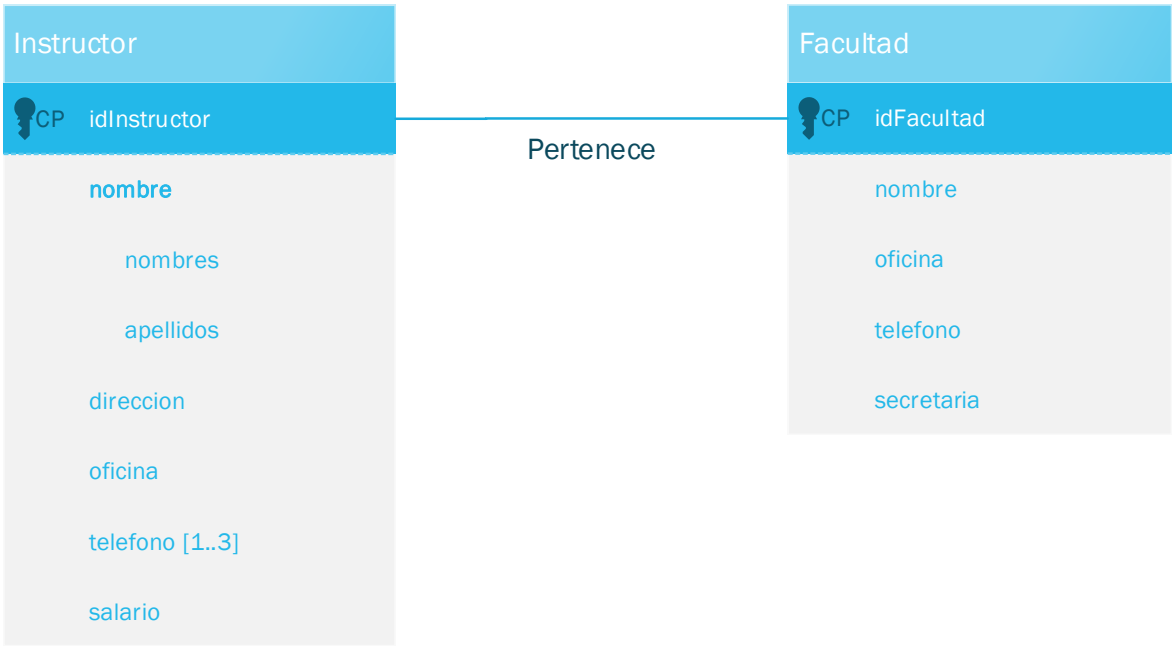


Figura 9 *Representación de Atributos de una Entidad*

Como ves, idInstructor e idFacultad usan la abreviación **CP** (clave primaria) para identificarse como llaves primarias de ambas entidades. La abreviación **PK** significa *Primary Key*, cuya traducción es llave primaria.

También se observa que el atributo nombre de Instructor es compuesto, por esto se añade una tabulación a los componentes nombres y apellidos.

En el caso de telefono que es multivariado, se incluyó su rango de elementos, expresando que mínimo debe existir un teléfono y máximo tres.

Las anteriores entidades tienen pocos atributos por lo que es muy cómodo ubicarlos visualmente. Pero si tus entidades tienen muchos atributos, no es recomendable tomarse un tiempo de transcripción en el diagrama para ponerlos todos.

Puedes solo ubicar las llaves primarias para realizar análisis rápidos como haremos frecuentemente a lo largo del ebook. Todo depende de ti y los requerimientos de modelaje que tu caso necesite.

3.7 Atributos en Relaciones

Conceptualmente los atributos también pueden usarse para describir las instancias de una relación. Esta preferencia nos permite aislar las características relacionadas a las diferentes relaciones existentes.

Estudiemos el ejemplo entre las entidades Estudiante y OfertaDeCurso. Es necesario aclarar que Curso es la declaración estructural de un curso genérico y OfertaDeCurso es el curso físico como tal.

Entrando en materia, sabemos que un Estudiante se encuentra en una oferta de curso, pero necesitamos saber que nota final obtendrá y si aprobó el curso o no.

Si Estudiante no existiera no habría a quién asignarle notas del curso, y si Curso no existiera no habría nada que calificar. Por deducción las notas individuales (notas) y la notaFinal es el resultado de la interacción entre ambas entidades por lo que debe pertenecer a relación *Asiste*.

3.7.1 Notación UML

Para representar estos atributos de la relación añadiremos una línea punteada, cuyo origen es el centro de la relación y su orientación es perpendicular a esta.

El contenido estará dentro de un rectángulo subdivido al igual que las entidades, pero en este caso la subdivisión superior no tendrá nombre.

Veamos:

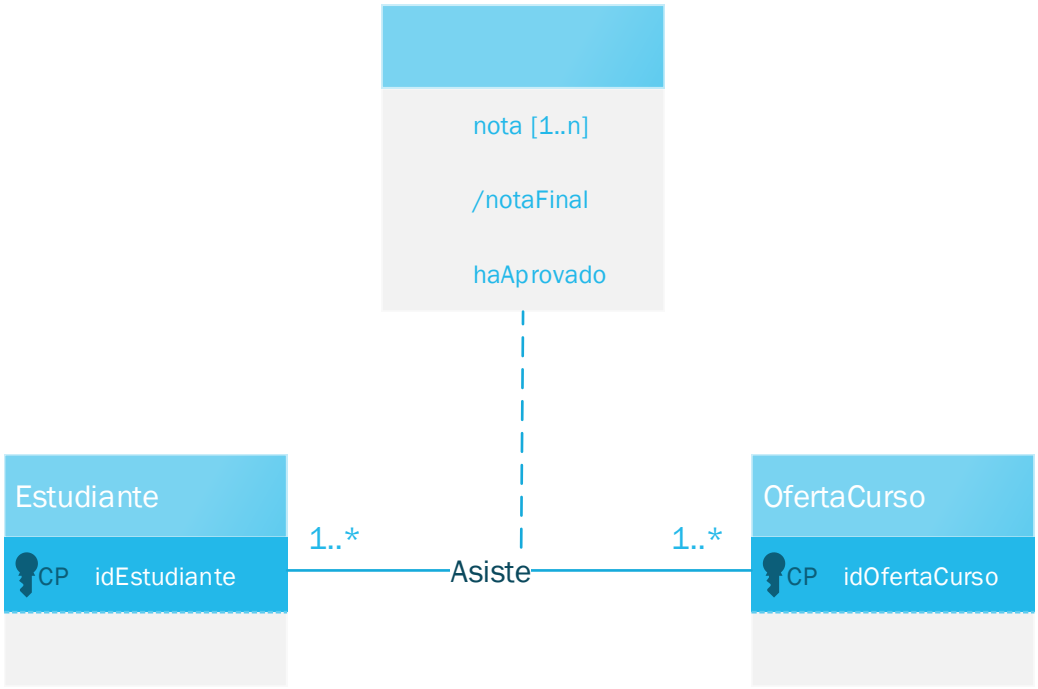


Figura 10 *Ejemplo de Atributos en una Relación*

4. ENTIDADES FUERTES Y DÉBILES

Las entidades pueden ser clasificadas tomando como factor la solidez con que se comporta en el modelo de datos. Una entidad que no depende de la existencia de otra se le llama **entidad fuerte** o **entidad padre**.

Por el otro lado, aquella que se ve afectada por la existencia de otra se le llama **entidad débil** o **entidad hija**.

Las entidades débiles tienen una características particular y es que no pueden diferenciar sus instancias con los atributos que poseen en esencia. Por lo que están atadas a la relación que tienen con la entidad fuerte para crear su llave primaria.

Por ejemplo, en la relación entre OfertaCurso y DiaOfertaCurso se identifica una relación fuerte-débil:

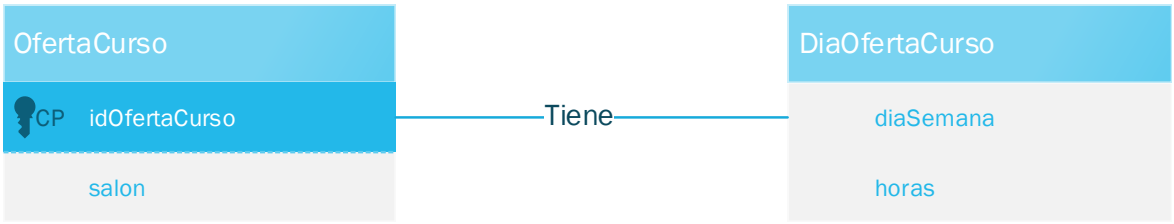


Figura 11 *Relación entre una Entidad Fuerte y una Débil*

OfertaCurso es la entidad fuerte debido a que tiene a idOfertaCurso establecido para diferenciar cada instancia, pero DiaOfertaCurso solo posee dos atributos que describen un horario.

Muchos pensarán: *"Bueno…pues le pongo una clave primaria idDiaOfertaCurso y todo arreglado"*.

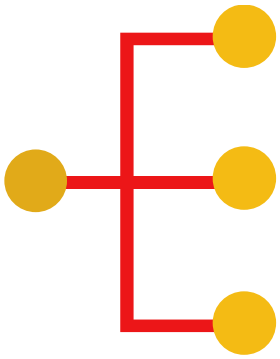
Es una solución ingeniosa, ya que se equilibra el balance de una forma automática, pero…si observas bien, la naturaleza de DiaOfertaCurso no viene con un identificador en su forma original, ya que actúa como una derivación de OfertaCurso.

Es decir, si la oferta del curso no existiera, entonces no se generarían horarios. DiaOfertaCurso es solo una añadidura generada por la necesidad de acondicionar

a otra entidad, la cual no puede ser considerada como atributo debido a su complejidad.

Diferenciar entre entidades fuertes y débiles nos permite modelar de una forma más elegante un problema y evita añadir llaves primarias innecesarias a las entidades, ya que si solo fuese añadirle un “id” a todo, entonces no estarías leyendo este ebook.

5. RESTRICCIONES ESTRUCTURALES



Hasta el momento hemos visto las relaciones entre entidades de una forma global y sencilla. No obstante, podemos detallar y limitar aún más la forma en que ocurren las relaciones para establecer más claridad en el modelo de datos.

Por eso el tema de esta sección son las **restricciones estructurales**, las cuales definen la cantidad de elementos que se relacionaran con los de otra entidad.

5.1 Multiplicidad

La **multiplicidad** es la cantidad posible de instancias de una entidad que se pueden relacionar con un solo elemento de otra entidad.

Con ella se establecen los límites que el modelo de negocios del cliente establece para relacionar la información.

Por ejemplo, en la relación Instructor *Pertenece* a Facultad, podemos afirmar que *"A una facultad pueden pertenecer varios instructores"*. En sentido contrario también podemos apreciar que: *"Un instructor solo puede pertenecer a una Facultad"*.

Este simple análisis de multiplicidad pronostica cómo evolucionará la relación entre las instancias de cada entidad.

Ahora, dependiendo del número de instancias involucradas en una relación binaria, las multiplicidades se dividen en varios posibles casos:

- Relación de *uno a uno*, representada por la notación 1:1 en UML.
- Relación de *uno a muchos*, representada por la notación 1:* en UML.
- Relación de *muchos a muchos*, representada por la notación *:* en UML.

5.2 Relaciones uno a uno (1:1)

Las relaciones uno a uno se dan cuando cada instancia de una clase se relaciona con una y nada más una instancia de otra entidad.

5.2.1 ¿Cómo determinar que la multiplicidad es 1:1?

La respuesta depende del contexto de los requerimientos planteados por el usuario. Por ejemplo, para el *Instituto San Judas* es indispensable que las facultades solo sean coordinadas únicamente por un instructor designado, por deducción los instructores no pueden ser coordinadores de más de dos facultades a la vez.

Aunque para muchos diseñadores son comprensibles mentalmente estas condiciones, existen otras profesionales que son más visuales y necesitan observar la acción.

En este caso puedes usar **redes semánticas** para asociar ejemplos de instancias entre las entidades, como se muestra en el siguiente diagrama:

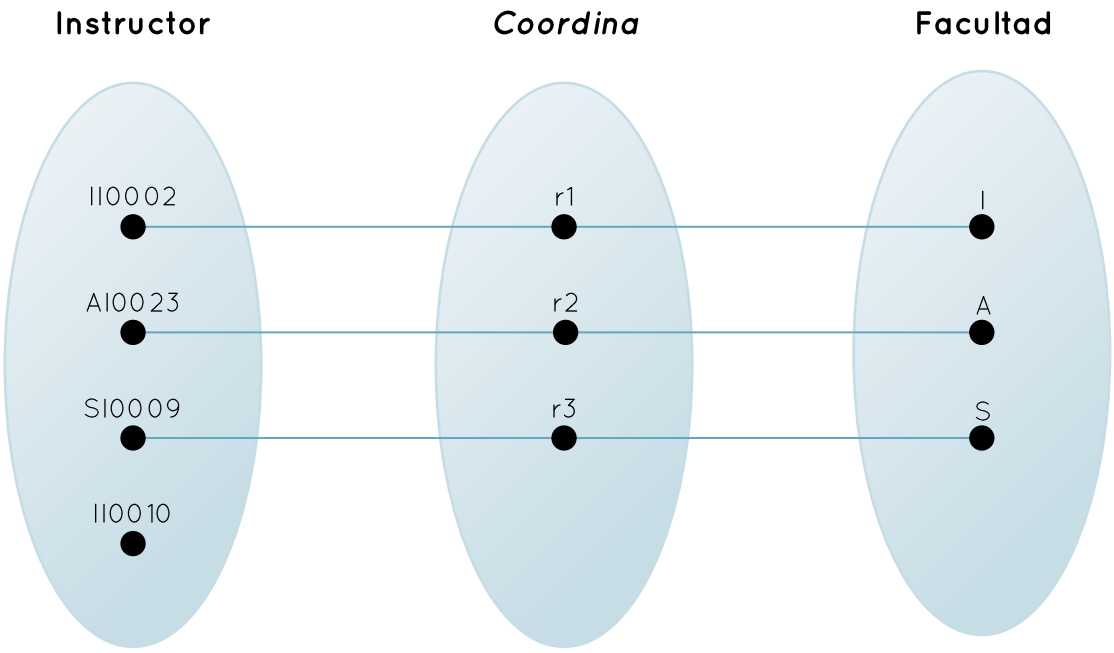


Figura 12 *Red Semántica ilustrando Multiplicidad 1:1*

Como ves, las rutas de relación ilustran que las instancias de prueba solo pueden tener una asociación a un solo elemento del lado contrario. Por ejemplo el instructor con idInstructor II0002 coordina la facultad I (idFacultad para Ingeniería).

Créeme que aunque se ve sencillo, existen relaciones que conceptualmente necesitan un nivel más avanzado de conocimiento y pensamiento para

determinar su multiplicidad. En estos casos, las redes semánticas te serán de gran ayuda.

5.2.2 Notación UML de la multiplicidad 1:1

Pondremos en el origen de la relación para cada entidad una expresión del tipo "a..b", donde *a* representa si todos los elementos participan en la relación y *b* el número máximo de elementos que pueden participar.

Veamos como incluir la multiplicidad en la relación previamente estudiada:

Figura 13 *Multiplicidad 1:1 en UML*

La Figura 12 muestra que cada facultad debe ser coordinada por un instructor (1..1), pero no es obligatorio que cada instructor coordine una facultad (0..1).

Aunque en esencia la relación es **1:1**, usamos el número *cero* (Veremos este concepto en el siguiente apartado) para representar que no todos los coordinadores pueden coordinar una facultad.

5.3 Relación uno a muchos (1:*)

Analicemos las relación entre la entidad Curso y OfertaCurso. Se tiene que por cada curso que exista dentro del pensum de una carrera se pueden generar varias ofertas a través del tiempo.

Es decir, que por cada curso establecido pueden generarse varias ofertas para los estudiantes.

Este es un claro caso de la relación uno a muchos, ya que una instancia de una entidad se puede relacionar con dos o más instancias de otra.

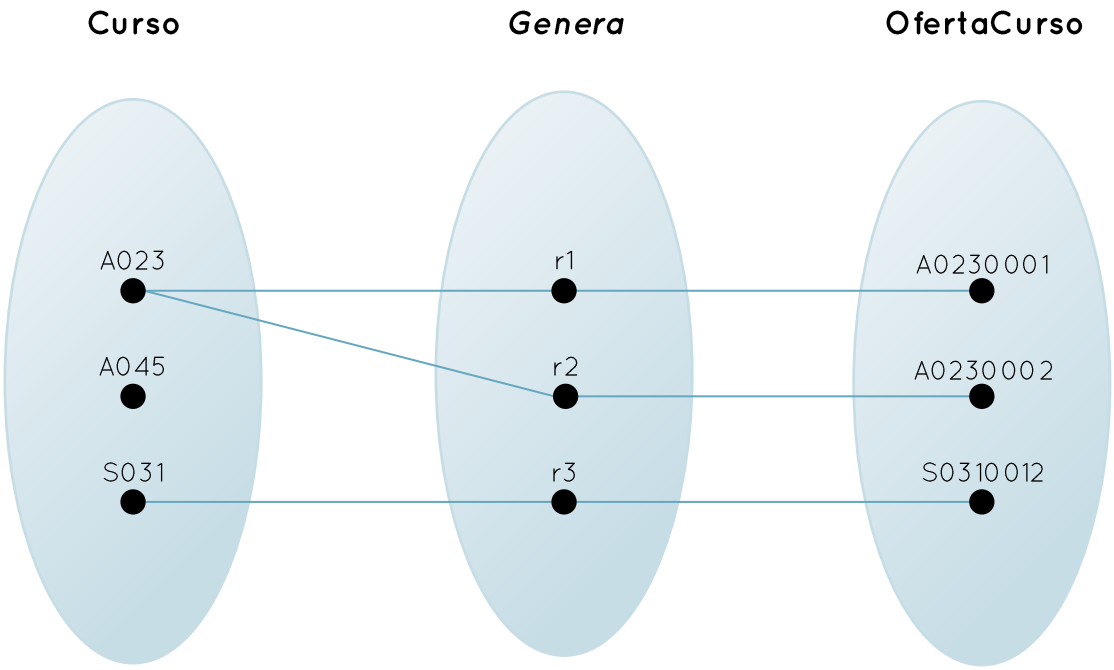


Figura 14 *Multiplicidad 1:* entre Curso y OfertaCurso*

Para confirmar esta multiplicidad debemos preguntarnos el objetivo de la relación en sentido contrario: *¿Puede una oferta de un curso generarse a partir de dos cursos?*

¡No!

Por lógica y contexto no es posible que basados en el curso *"Matemáticas I"* y *"Diseño de Bases de Datos"* se haga una combinación para generar una oferta de curso.

5.3.1 Notación UML de la Multiplicidad 1:*

La notación UML para indicar *varios elementos* es el símbolo asterisco (*). También es válido especificar el número preciso de entidades que participarán. Si el número mayor o igual a 2 se dará por sentado la intervención de múltiples instancias.

Veamos cómo quedaría el diagrama de la relación:

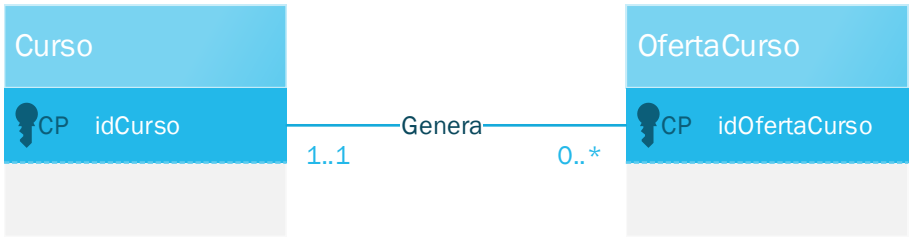


Figura 15 Relación 1: en UML*

En la Figura 14 se identifica que aunque un curso puede generar varias ofertas de curso también es posible que no genere ninguna (se indica 0).

Esto se debe a que en algunos casos puede que los estudiantes se atrasen y nadie requiera la apertura de una oferta. O simplemente existan estudiantes de primeros semestres, pero no de semestres superiores.

5.4 Relación muchos a muchos (*:*)

Como ya puedes deducir, esta relación se caracteriza porque varias instancias de una entidad tienen asociación con múltiples instancias de otra.

Justamente la relación entre Estudiante y OfertaCurso aplica para nuestro estudio. Ya que muchos estudiantes pueden estar en uno o varios cursos que se han ofrecido.

Por lo que *María Castro* puede pertenecer a *Química II* junto a *Juan Pedraza*, pero a su vez *María Castro* está viendo otro curso ofrecido llamado *Física Mecánica*.

Al igual que como lo hemos realizado con los otros dos tipos de multiplicidades, las redes semánticas nos permitirán determinar con instancias de prueba, la veracidad de la multiplicidad:

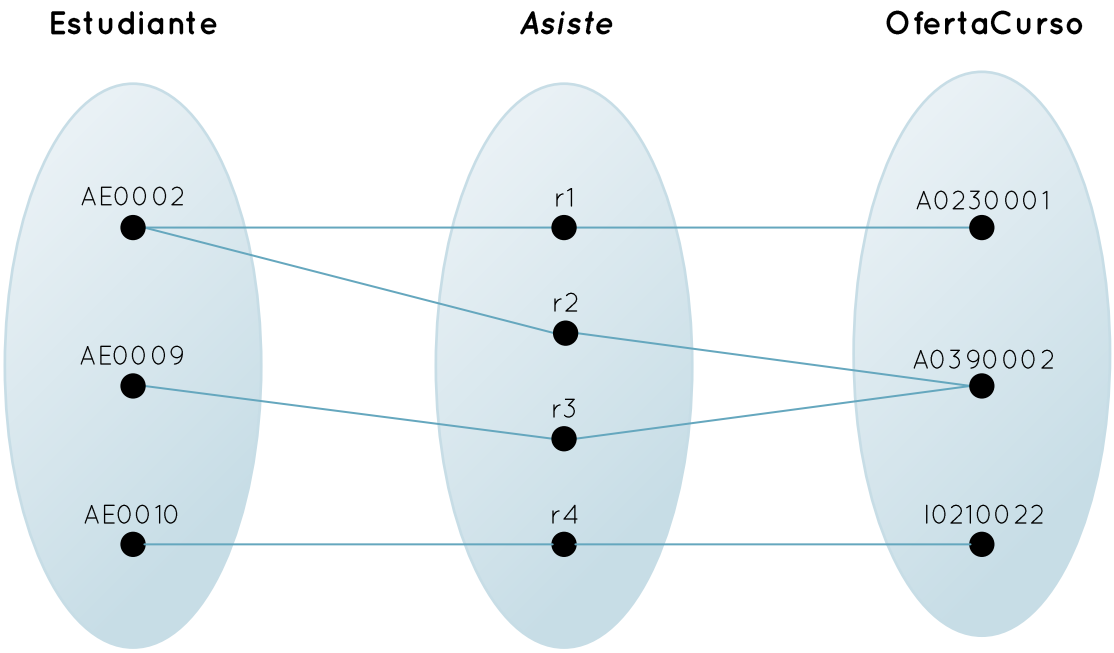


Figura 16 *Red Semántica de una Relación *:**

Veamos cómo se vería el diagrama ER de esta relación muchos a muchos:

Figura 17 *Diagrama UML de una Relación *:**

En la anterior situación deben realizarse preguntas como *¿cuál es la máxima cantidad de estudiantes en un curso?*, o *¿es posible abrir una oferta con tan solo un cupo separado?*

Dependiendo de las reglas de negocio del colegio, los extremos variarán para modelar la esencia de la operación.

5.5 Multiplicidad en Relaciones Complejas

La multiplicidad de relaciones complejas es la cantidad posible de ocurrencias de una entidad en una relación **n-naria** cuando otros valores permanecen constantes.

Este tema es mucho más elaborado, pero sin embargo entendible. A razón de la complejidad que implica una relación n-aria, los análisis de relación deben efectuarse dejando fijos las instancias de ciertas entidades para determinar la multiplicidad.

Si alguna vez has visto calculo, comprenderás un poco mejor el hecho de tener múltiples variables en una función.

Al igual que en las relaciones binarias, en las relaciones complejas se debe expresar la cantidad de elementos que intervienen de cada entidad. Se usa la misma notación para mostrar los números.

El uso de redes semánticas se lleva a cabo dejando fija las instancias de una entidad para analizar el resto de variaciones. Este proceso es muy similar a la **derivación parcial de funciones**.

Por ejemplo, analicemos la relación ternaria entre Instructor, Estudiante y OfertaCurso.

Entre estas tres entidades surge un evento puntual donde el instructor dicta uno o varios cursos a varios estudiantes.

5.5.1 Determinar la multiplicidad de una relación n-aria

Para determinar la multiplicidad, se toman grupos simplificados de *n-1* elementos vs. las instancias de una entidad que variará sus elementos para revisar el comportamiento.

En cada grupo simplificado de instancias, solamente debe variar un elemento, los demás elementos deben permanecer constantes.

Con estas instrucciones claras, veamos cómo se relacionan las instancias de Instructor con el par OfertaCurso/Estudiante:

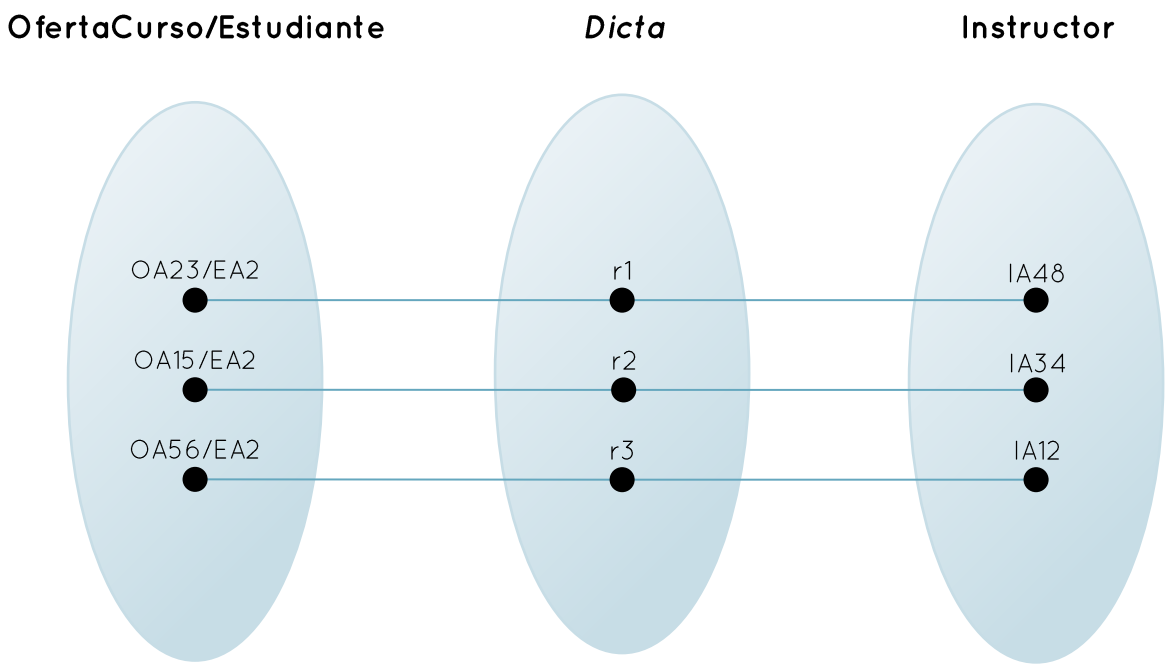


Figura 18 *Identificando la Multiplicidad de Instructor en la Relación Ternaria*

Como ves, si dejamos fijo el valor de las instancias de Estudiante y se varían las ofertas de cursos, existe una multiplicidad *1:1* con la entidad Instructor.

El objetivo era ver como el estudiante EA2 se relacionaba con los instructores que dictan los cursos que él está viendo.

Ahora determinemos la multiplicidad de Estudiante:

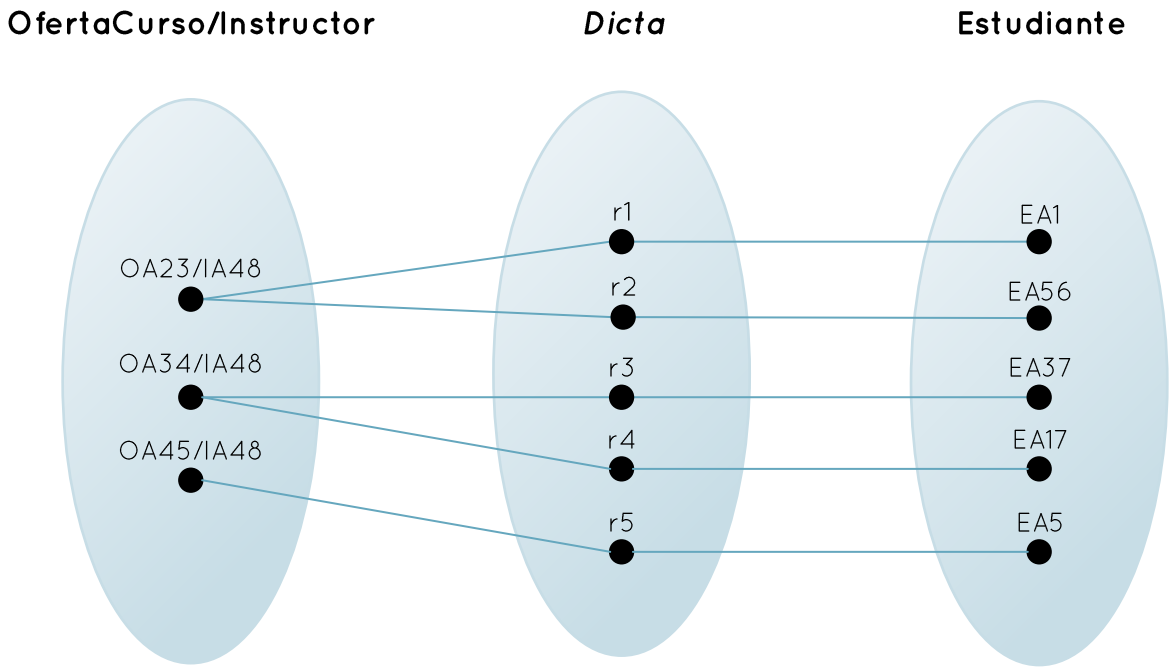


Figura 19 *Identificando la Multiplicidad de Estudiante en la Relación Ternaria*

En este caso cada instancia del par fijo puede relacionarse con uno o más estudiantes, por lo que existe una multiplicidad *1:**.

Por ultimo armamos el par fijo Estudiante/Instructor para determinar la multiplicidad de OfertaCurso:

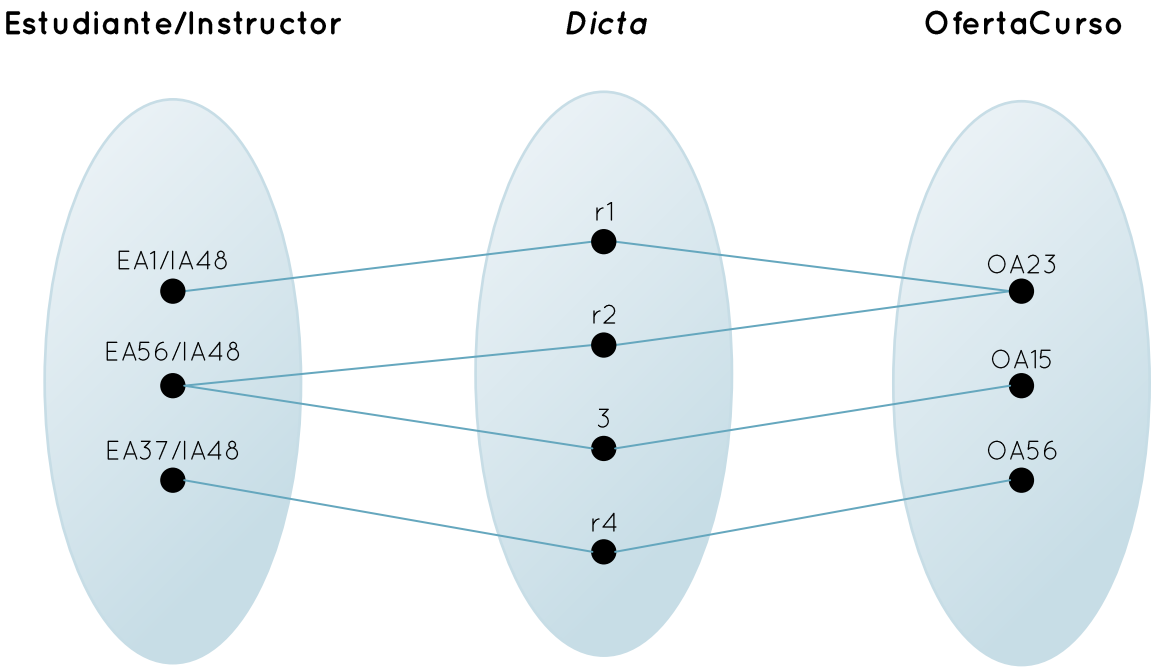


Figura 20 *Identificando la Multiplicidad de OfertaCurso en la Relación Ternaria*

De la ilustración 19 se deduce que la multiplicidad es *1:**.

Ahora solo queda usar la notación UML para representar el modelo de datos:

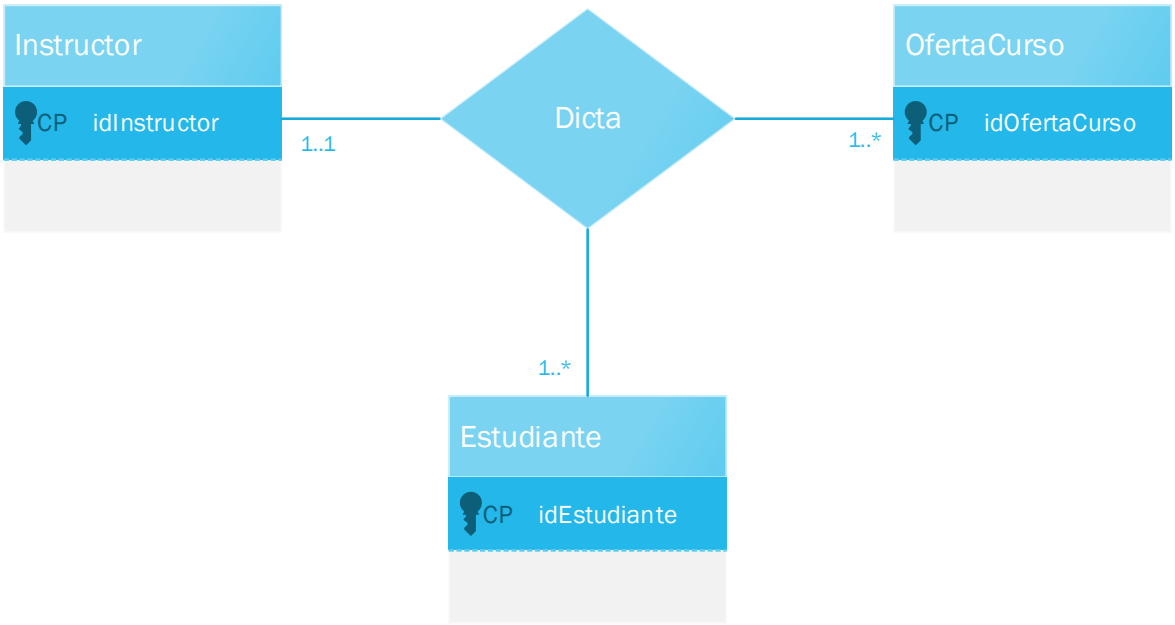


Figura 21 *Representación UML de la Multiplicidad de una Relación Ternaria*

Del anterior diagrama se deduce que cada instancia de la relación *Dicta* se puede representar como el conjunto de ternas

$$r = \{ \text{instructor, oferta\_curso, estudiante} \}$$

y su multiplicidad puede ser escrita como la abreviación *(1 : * : *)*.

> Para una relación **cuaternaria** se sigue el mismo procedimiento. Armas grupos simplificados de 3 entidades, dejando constante dos entidades y luego los relaciones con la entidad sobrante.
>
> Hazlo por cada una de las 4 entidades.

5.6 Restricciones de Participación y Cardinalidad

La multiplicidad consiste de dos restricciones individuales llamadas **participación** y **cardinalidad**.

La participación define si todas o algunas instancias de una entidad intervendrán en la relación. Y la cardinalidad establece la cantidad máxima de instancias que intervendrán.

Si analizamos la notación de las relaciones binarias *1:1*, *1:** y *:* se verán representadas por la cardinalidad de las entidades que tienen la relación.

La participación podemos definirla como **opcional** a través del número *cero* para indicar que no todas las instancias participan en la relación. Y también podemos indicar que es **obligatoria**, con tal solo indicar un el número uno.

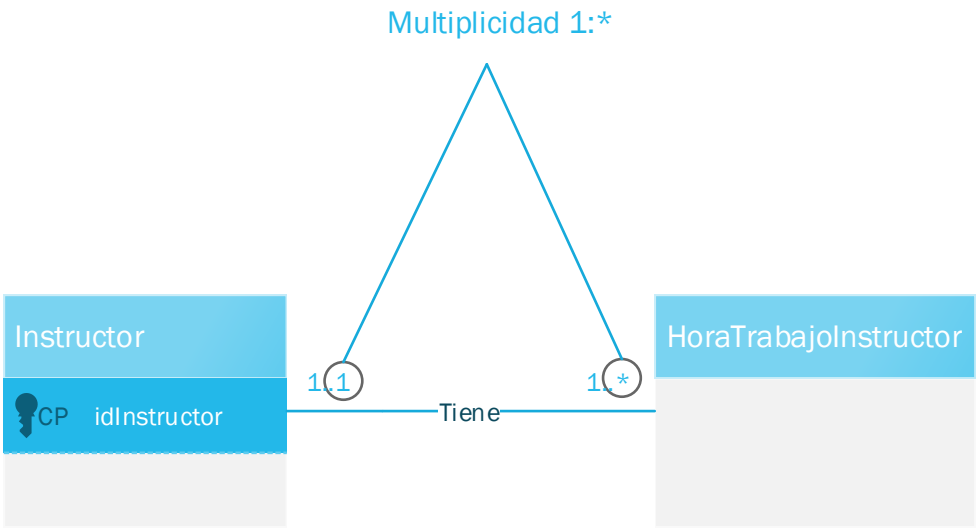


Figura 22 *Las cardinalidades representan la Multiplicidad de una Relación*

5.6.1 ¿Cómo Identificar la Participación y Cardinalidad?

Cuando hayas identificado una relación entre entidades inmediatamente debes estudiar cuales son la participación y cardinalidad para establecer la multiplicidad.

Las siguientes indicaciones pueden ser de utilidad para indicar ambas restricciones:

Cardinalidad

Pregunta: ¿Es obligatorio que todas las instancias de esta entidad participen?

Acción: Si la respuesta es no, ubica un **0**. Si es afirmativa, entonces un **1**.

Multiplicidad

Pregunta: ¿Máximo cuantas entidades pueden llegar a participar con una instancia de esta entidad?

Acción: Indica el número exacto si está a tu disposición. De lo contario ubica el símbolo *****. En caso de solo ser un elemento ubica el número **1**.

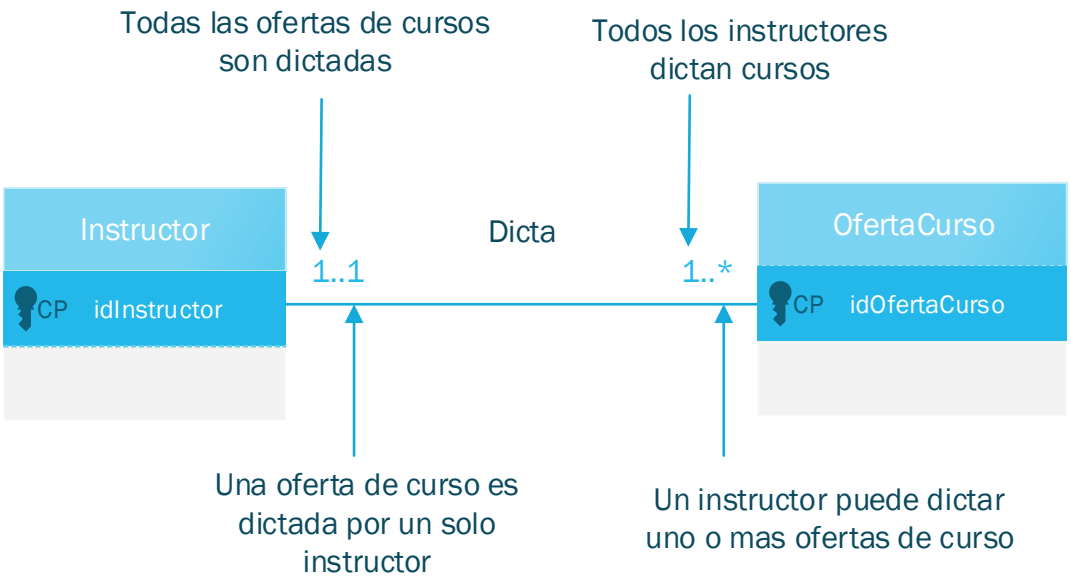


Figura 23 *Cardinalidad y Participación expresadas*

6. PROBLEMAS COMUNES DE MODELAMIENTO

A continuación veremos algunos de los errores más comunes en el diseño de base de datos, tanto estructurales como metodológicos.

6.1 La Trampa del Ventilador

Este error surge cuando existe una relación entre entidades donde las ocurrencias se asociación de forma ambigua. Esto quiere decir que se dificulta obtener una ruta concreta para llegar a una instancia.

Las trampas ventilador ocurren cuando se establecen relaciones 1:* desde una entidad central. Un ejemplo concreto podemos verlo entre la posible relación de Facultad, Instituto e Instructor. Donde Instituto es cada una de las sedes del en la región (los requerimientos no lo expresan),

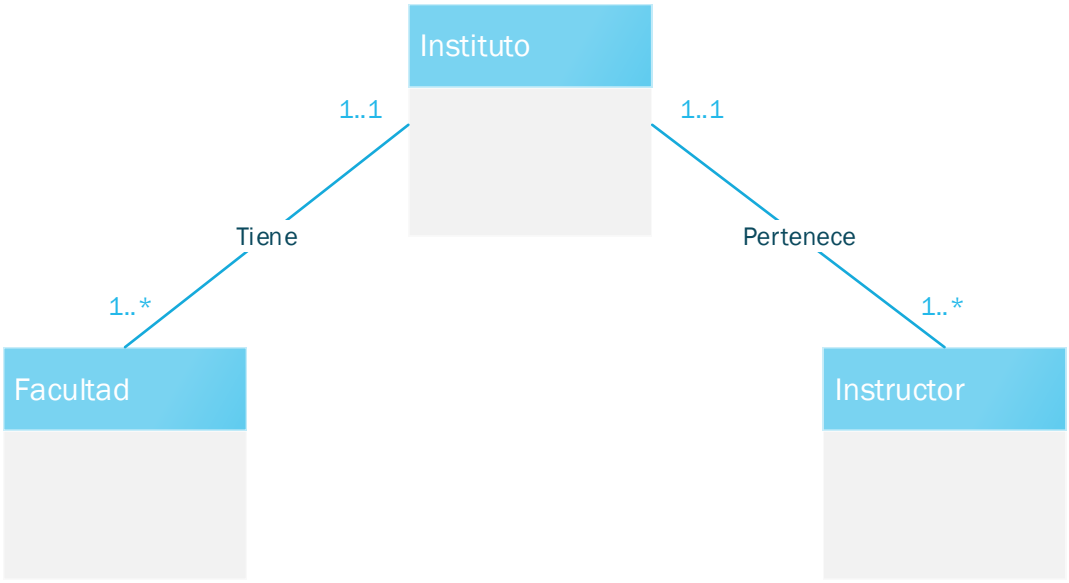


Figura 24 *Ejemplo de Trampa del Ventilador*

Como se ve, un instituto tiene varias facultades y además en él pueden pertenecer varios instructores. Veamos una representación con redes semánticas:

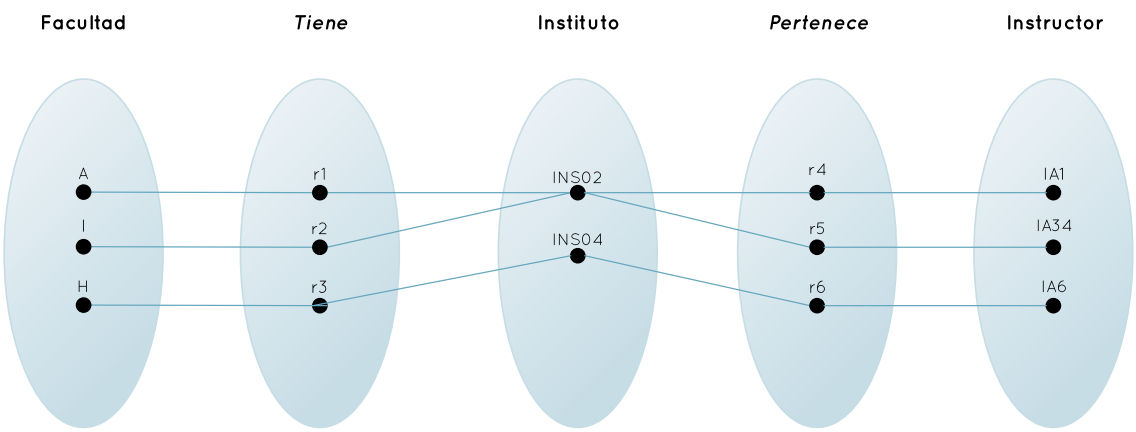


Figura 25 *Red Semántica Evidenciando las Anomalías en una Trampa Ventilador*

A primera vista no se nota la anomalía con facilidad, pero que ocurre si nos preguntamos:

¿A qué facultad pertenece el instructor con el idInstructor IA1?

Al seguir el recorrido desde esta instancia hacia la entidad Facultad verás que la respuesta es ambigua, ya que la conexión termina en 2 instancias de la facultad (A e I).

Este es el problema de diseño generado por una **trampa ventilador**.

SOLUCION

Se debe eliminar la trampa invirtiendo la multiplicidad de la relación entre Instituto y Facultad para que las asociaciones fluyan ante una sola ruta específica. Veamos en una imagen la reestructuración del modelo ER:

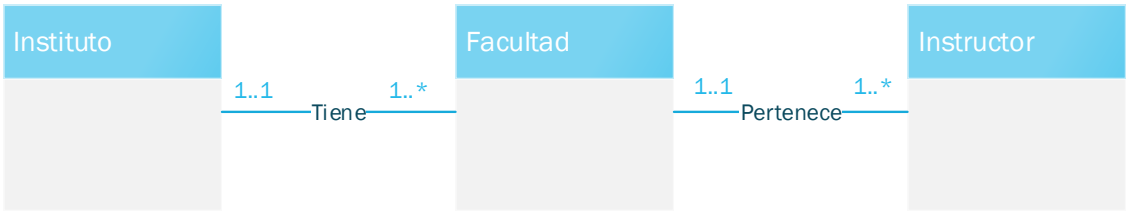


Figura 26 *Reestructuración del Diagrama para Solucionar la Trampa del Ventilador*

Este problema en particular fue originado por una mala interpretación de la realidad, ya que un instituto contiene a una facultad y una facultad contiene a los instructores.

Ahora veamos la red semántica:

Figura 27 *Congruencia de las Relaciones luego de Solucionar la Trampa del Ventilador*

Al ver de nuevo las relaciones ya podemos obtener respuestas limpias. Con esta solución se esclarece que el instructor IA1 pertenece a la facultad de Ciencias Administrativas y Contables (A).

6.2 La trampa del Abismo

La trampa del abismo ocurre cuando se da por hecho que una relación entre entidades cubre la ruta desde las instancias de una hacia la otra, pero en la práctica se descubre que hay elementos que al final quedan aislados.

Claramente esto destruye la integridad de la base de datos, ya que se solicita información que existe, pero nunca se encontrará.

Esta anomalía se produce cuando se usa participación opcional (cero o más) en las multiplicidades de alguna relación. Un ejemplo de este caso sería la cadena de relaciones entre Facultad, Instructor y Seminario:

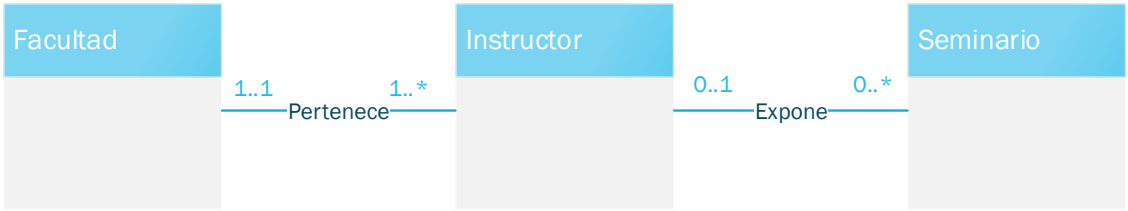


Figura 28 *Ejemplo de Trampa del Abismo*

Esta relación implica que un instructor de una facultad puede exponer cero o varios seminarios.

Donde la red semántica se ve así:

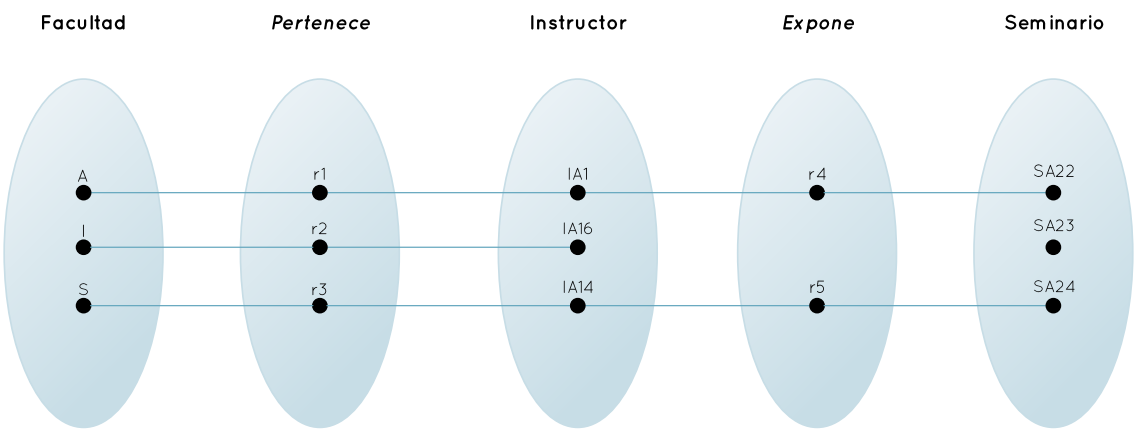


Figura 29 *Evidencia de los Efectos de una Trampa de Abismo*

Haz el ejercicio e intenta obtener la facultad a la que pertenece el seminario SA23.

¿Imposible?

¡Exacto!, este elemento queda totalmente perdido y ya nadie lo puede salvar. Literalmente cae al "abismo".

SOLUCIÓN

Para tapar este hueco debemos simplemente agregar una nueva relación desde la entidad inicial hacia la entidad herida. Con ello subsanamos la falta de integridad.

El modelo ER y la red semántica muestran como la solución se hace efectiva al establecer este nuevo canal:

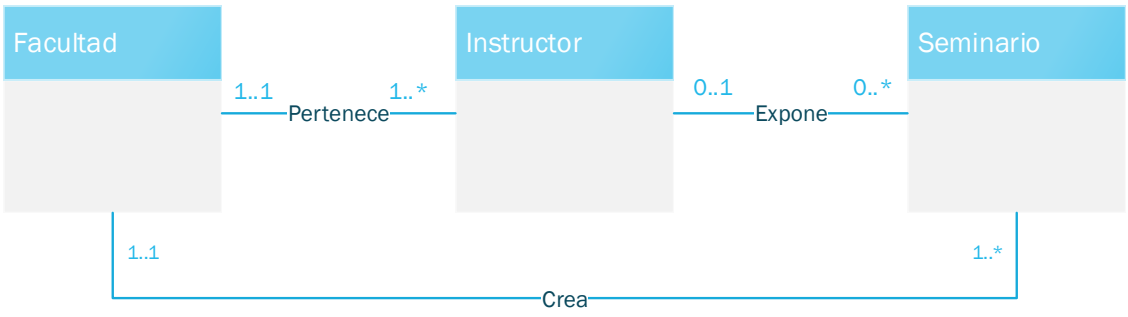


Figura 30 *Inclusión de la Relación Crea para Solucionar la Trampa Abismo*

Al ver de nuevo la red semántica veremos que el camino hacia el seminario AS23 ya ha sido cubierto con la relación *Crea*:

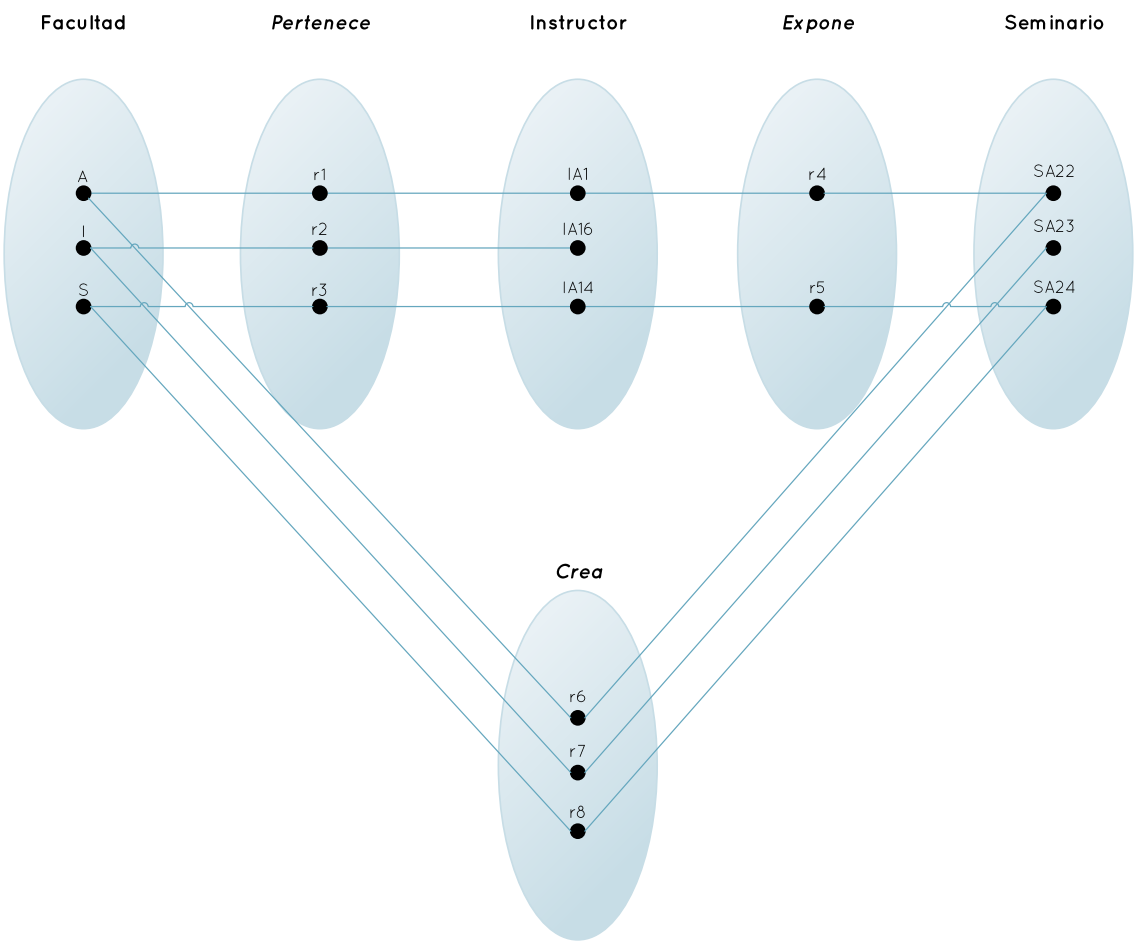


Figura 31 *Solución de la Trampa Abismo a través de una Red Semántica*

6.3 Mala Elección de Nombres

Al crear un esquema de la base de datos es frecuente que se elijan nombres que a lo largo del diseño no sean acordes a la realidad.

Normalmente esto sucede porque en el contexto que se intenta modelar no existen pruebas suficientes para obtener buenos nombres o simplemente porque el diseñador no se ha detenido a pensar el alcance de los elementos.

SOLUCIÓN

Veamos algunas reglas para reducir una mala convención de nombres:

- Usa un sustantivo en singular para las entidades. Esto se debe a que estamos analizando la interacción de elementos particulares para llegar a

generalidades. Normalmente se usan los nombres más populares por si no existe un estándar de calidad, de lo contario se sigue dicho estándar.

- Asegúrate que el verbo que describe una relación esté en presente simple y además pueda ser leído fácilmente en ambas direcciones de una relación binaria. Por ejemplo, *un Estudiante pertenece a una Facultad*; en sentido contrario se manifiesta que *a una Facultad pertenece un Estudiante*.
- Conserva el nombre verdadero de los atributos para mantener la claridad. Si el nombre tiene más de 12 caracteres, puedes abreviarlo para compactarlo, dejando un nombre intuitivo. Por ejemplo `noTelefono` puede ser leído rápidamente como número telefónico aunque haya sido abreviado.

6.4 Pensar en Tablas

El diseño conceptual tiene como objetivo comprender la situación actual de tu cliente. Se trata de explorar continuamente como representar la realidad del negocio con tu capacidad de abstracción.

Pero muchas personas saltan este paso y piensan directamente en que tablas deben crearse, en cómo deben acomodarse las llaves foráneas, en que DBMS elegir y demás características relacionales.

Ese continuo deseo de querer implementar rápidamente es natural, pero intentar encontrar una solución sin siquiera haber comprendido de forma general el dominio del problema, puede traducirse en pérdida de recursos.

SOLUCIÓN

Si crees que este este es uno de tus problema, quizás puedas adoptar un **modelado de datos ágil**. Este paradigma es excelente para quienes se basan en metodologías ágiles. Donde el diseño, la codificación, las pruebas y el despliegue son continuos.

Esto no significa saltarse el diseño. Significa realizar el diseño conceptual, diseño lógico y el diseño físico en bloques de tiempo más cortos, lo que acerca con mayor agilidad la codificación.

Particularmente este es el enfoque que uso actualmente, junto a la metodología de diseño de base de datos que explico en este ebook.

> Si deseas saber más sobre *Modelado Ágil* puedes visitar este sitio dedicado especialmente a este tema:
>
> http://www.agiledata.org

PARTE 2
EL MODELO ENTIDAD-RELACIÓN EXTENDIDO

INTRODUCCIÓN

¿Cómo modelar situaciones donde las instancias de una entidad pueden clasificarse en subgrupos?

¿Has notado que hay entidades que tienen atributos similares aunque sean consideradas por separadas?

La evolución en la administración de la información hace que cada vez más los diseñadores de bases de datos se ingenien nuevas formas de dar soluciones a problemas complejos presentados en distintos campos, como la medicina, producción musical, herramientas CRM, ERP, etc.

Debido a esto, los conceptos que hemos visto hasta ahora pueden ser limitados para modelar situaciones complejas.

Es por eso que esta sección aborda el modelo **Entidad-Relación Extendido** (EER), el cual comprende nuevas implementaciones abstractas para comprender aquellas situaciones donde hay conceptos avanzados de bases de datos como lo son la *Generalización*, *Especialización*, *Agregación* y *Composición*.

Puedes ver el modelo EER final para el caso de estudio en el siguiente diagrama o buscar el archivo *PNG* llamado *“Diagrama EER San Judas”:*

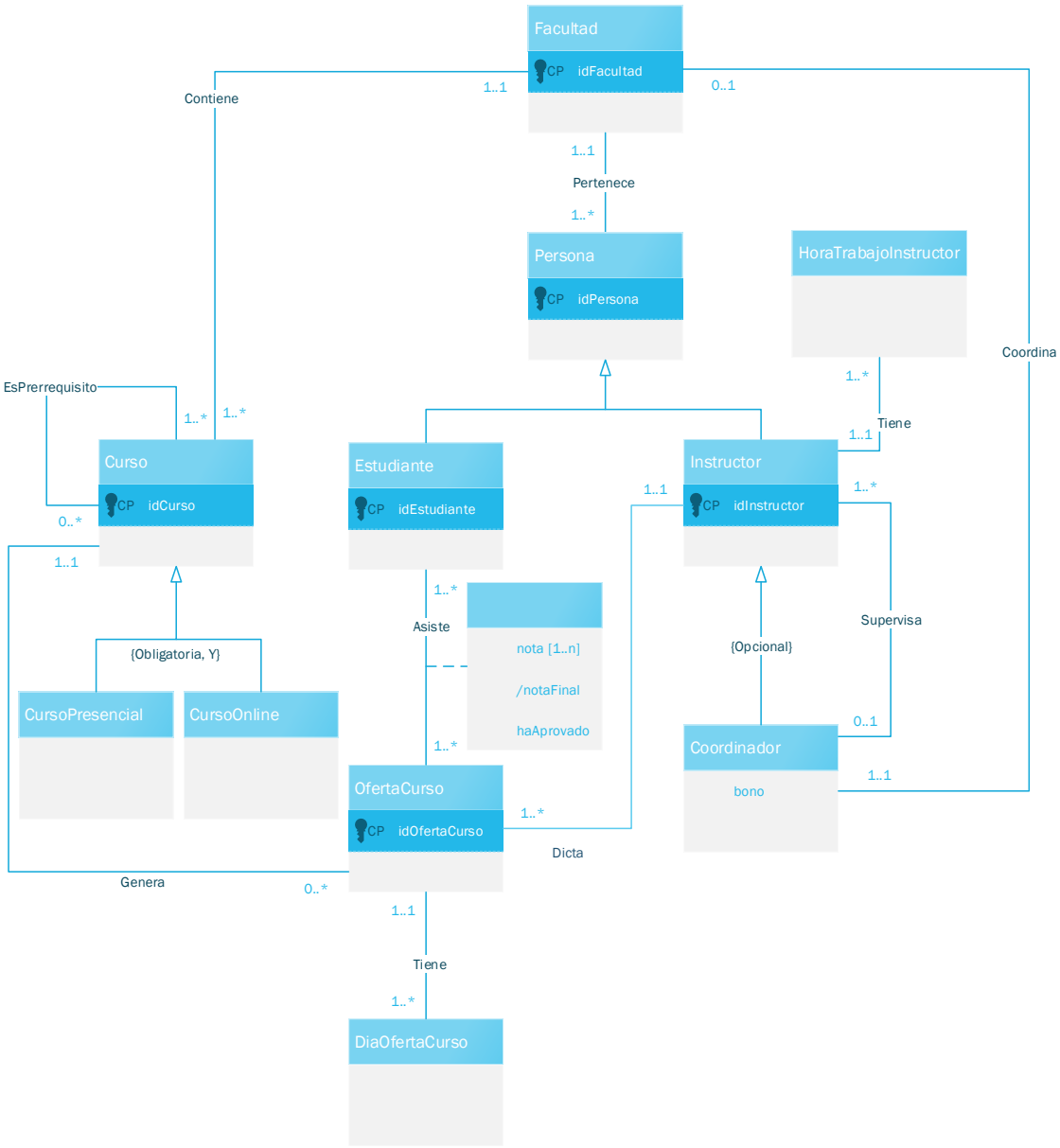
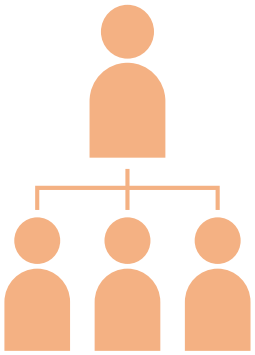


Figura 32 *Diagrama EER final del Instituto San Judas*

7. GENERALIZACIÓN Y ESPECIALIZACIÓN



Los procesos de generalización y especialización surgen al entender que existen entidades que pueden derivarse de otras debido a su estructura jerárquica o calificativa. Lo que clasifica a las entidades en superclases y subclases.

Una **superclase** es una entidad cuyas instancias muestran claramente que se pueden clasificar en distintos grupos aun perteneciendo a la misma entidad. Del otro lado, una **subclase** es cada uno de esas divisiones de las instancias de una entidad.

Un claro ejemplo de esta situación es la división de los instructores en aquellos que son coordinadores y los que no lo son. En ese caso la entidad Instructor es una superclase y la entidad Coordinador es una subclase.

También podríamos ver a las entidades Instructor y Estudiante como subclases de una entidad oculta denominada Persona, ya que ambas tienen en común varios aspectos.

Otro ejemplo lo podemos encontrar en los instructores que trabajan tiempo completo y aquellos que son contratistas. Donde Instructor es la superclase.

Los estudiantes que se han graduado y aquellos que aún no son otra opción para clasificar las instancias de estudiante. Siendo estudiante la superclase y EstudianteGraduado y EstudianteNoGraduado las subclases.

A este tipo de relaciones les llamaremos relación **superclase/subclase**, por lo que si hablásemos de la relación entre Persona y Estudiante nos referiríamos a la relación Persona/Estudiante.

7.1 Relaciones Superclase/Subclase

Por simple deducción sabemos que una instancia que pertenece a una subclase también pertenece a la superclase.

Este tipo de relaciones siempre serán *1:1* entre las instancias de la superclase y de las subclases. En la siguiente red semántica se representa la relación entre Persona, Instructor y Estudiante:

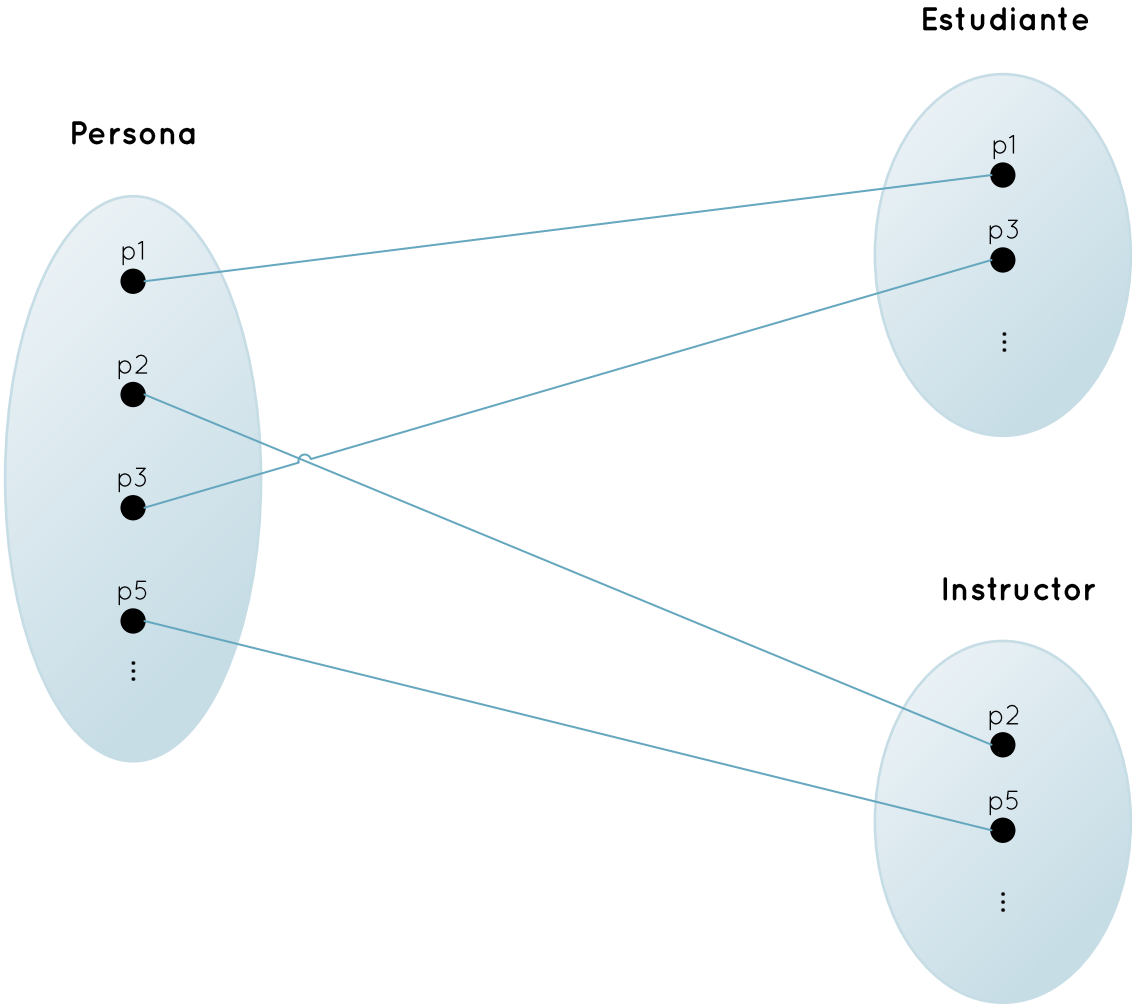


Figura 33 *Red Semántica de una Relación superclase/subclase*

Una relación superclase/subclase brinda las posibilidades de que cada subclase pueda tener atributos particulares que otras subclases o incluso la misma subclase no contiene.

Similar sucede con las relaciones, una subclase puede tener una relación que solo afecte a estas instancias sin tocar a otras subclases.

Estas características evitan que el diseñador rescriba conceptos similares más de una vez, lo que representa ahorro en tiempos de diseño y mejor legibilidad. Además añade más detalle y comprensión para el equipo encargado del desarrollo, acercando el enfoque a la programación orientada a objetos.

7.2 Herencia de atributos

El proceso de herencia de atributos se da naturalmente al considerar la pertenencia de una subclase a una superclase.

Si tomamos el ejemplo de que un Estudiante es una Persona, será claro que estudiante tendrá los datos comunes de Persona como lo son el nombre, teléfono, dirección, etc. Al igual que Instructor, quién también hereda estos mismos atributos por pertenecer a Persona.

También puede existir el caso cuando dos superclases le heredan sus atributos a una subclase. A esta situación se le llama **herencia múltiple** y la subclase que se ve afectada se le llama **clase compartida**.

7.3 Proceso de Especialización

Cuando el diseñador nota que las instancias de una entidad se pueden clasificar en subclases, se está realizando un proceso de **especialización**. Esto nos permite distinguir sus diferencias para darles un trato particular en el modelado.

Por ejemplo, la entidad Instructor en inicio pasa por inadvertida hasta que se descubre que un instructor puede ser el coordinador de una facultad. Justo en ese momento se revela la distinta naturaleza entre ciertas instancias de esta entidad.

Por lo que podemos pensar en los beneficios de realizar una especialización y clasificar a los instructores en quienes son coordinadores y entre quienes no.

7.4 Proceso de Generalización

La **generalización** es el proceso contrario a la especialización. Se trata de identificar que atributos tienen en común ciertas entidades para abstraer una superclase que las referencie.

En origen de la superclase Persona se da por generalización, ya que se identificaron que Instructor y Estudiante poseen varios atributos en común, por lo que era lógico pensar en obtener una superclase que estaba implícita.

7.5 Notación UML de Especialización y Generelización

Las relaciones superclase/subclase se representan con una línea desde la superclase hacia las subclases, donde el extremo que apunta a la superclase es un triángulo vacío.

Veamos la representación de la generalización entre Persona, Estudiante e Instructor:

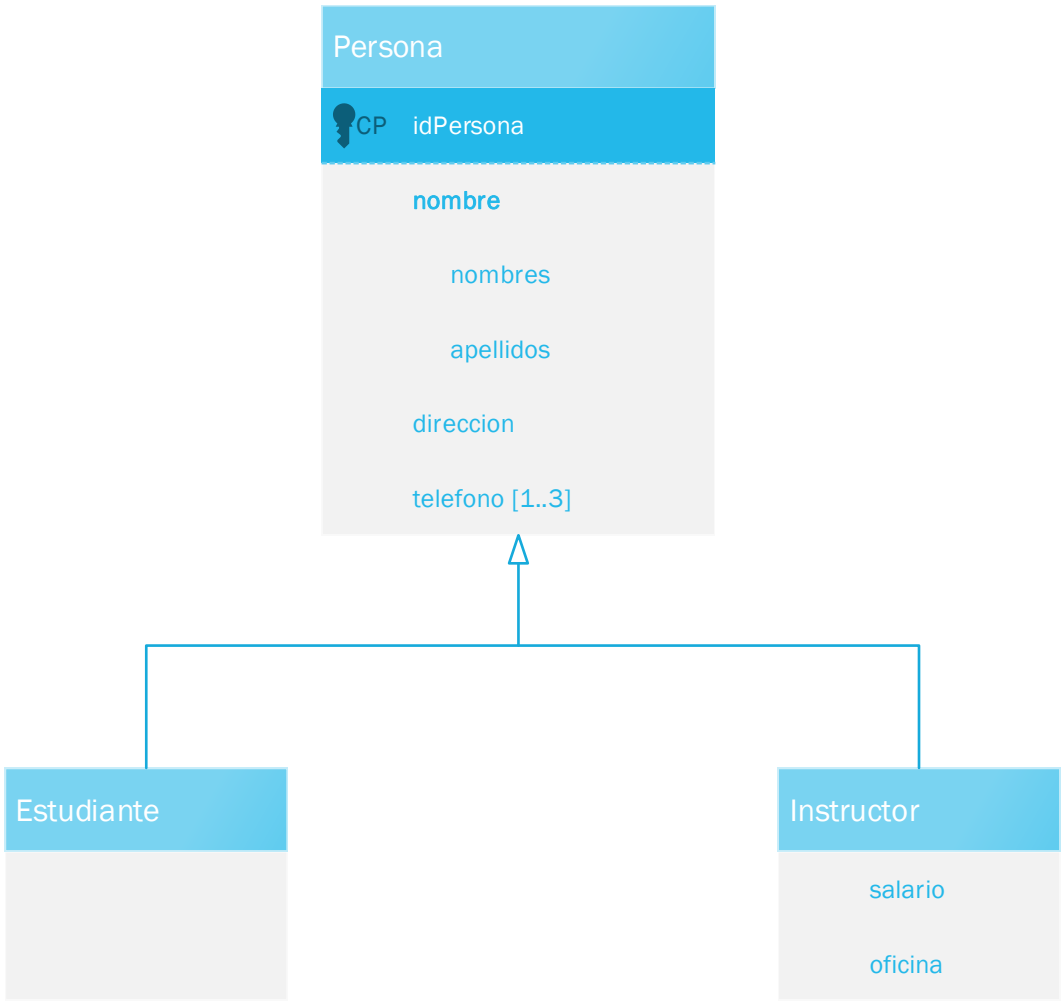


Figura 34 *Generalización de la Clase Persona*

7.6 El problema del diamante

Supón que un instructor tiene la posibilidad de estudiar una carrera en el mismo instituto. Esto lo convertiría en estudiante inmediatamente y el diagrama de la especialización se podría proyectar de la siguiente forma:

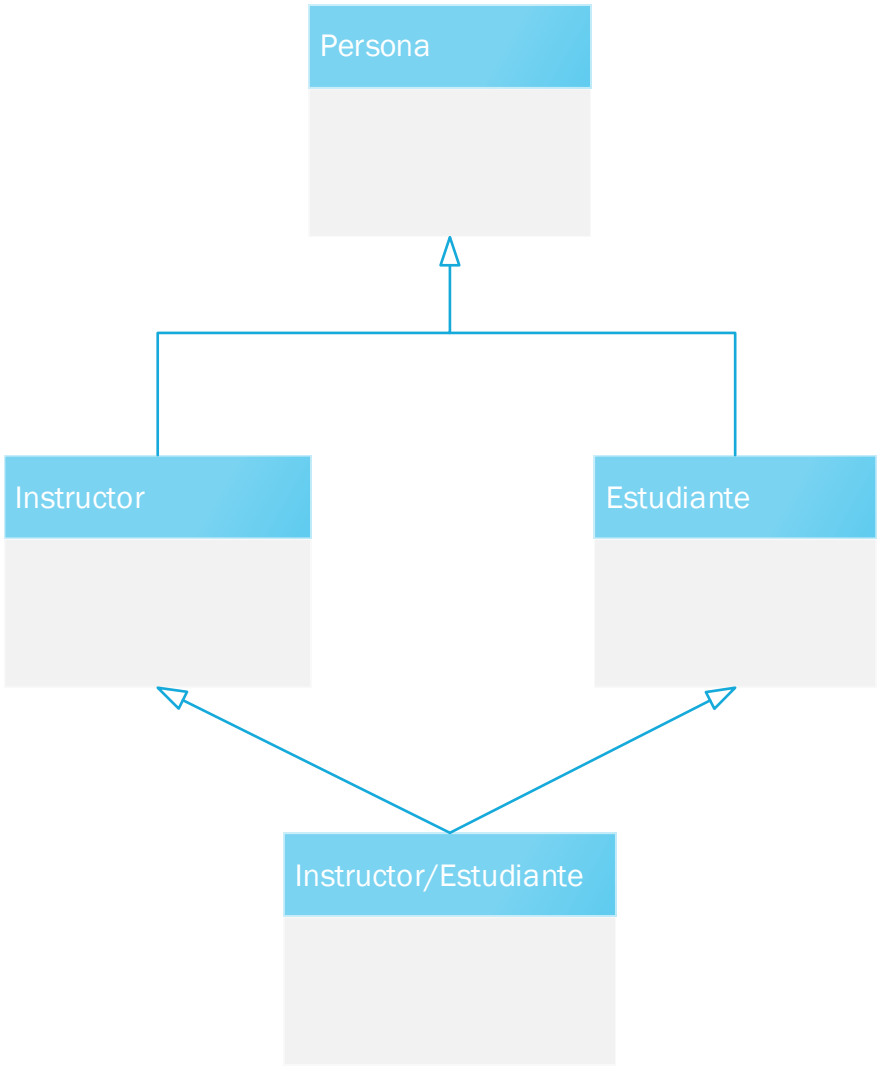


Figura 35 Herencia de Atributos en Forma de Diamante

En este caso Instructor/Estudiante es el resultado de una múltiple herencia por parte ambas entidades, lo que resolvería momentáneamente el dilema.

Ahora consideremos un pequeño test. Supongamos que existen dos estudiantes, *María Carvajal* y *Pedro Restrepo*. También tenemos dos profesores: *Astrid Villa* y *Pablo Milanés*.

Suponga que los dos profesores han decidido estudiar y ahora tendremos dos nuevas personas en el modelo, Astrid Villa y Pablo Milanés como instructores que estudian.

Pregunta: *¿Cuántas personas hay en el modelo?*

Cuatro no es la respuesta correcta. Desde que ambos instructores se han vuelto estudiantes se interpreta que existen seis personas. De las cuales 2 cumplen roles distintos.

Ahora piensa en los atributos de cada instructor/estudiante. *¿Si fuésemos a actualizar la dirección de Astrid Villa que pasaría?* Tendríamos que actualizar dos atributos, ya que se encuentra dos veces en la entidad Persona, por ende debe heredar los atributos de la entidad Instructor y la de Estudiante.

SOLUCIÓN
Este inconveniente se arregla cambiando el enfoque conceptual de la situación.

Hasta ahora se había pensado que un estudiante y un instructor son diferentes personas, pero como están las cosas, estos pueden ser la misma persona si no que realizando acciones distintas.

Debido a que es una misma persona asumiendo varios roles dentro del instituto, es mejor cambiar la relación entre la entidad Persona y la condición de su acción.

Para ello ubicamos una entidad llamada Acuerdo o Contrato, la cual contiene todos los atributos de cada condición especifica. De ella podemos derivar una subclase especial para un acuerdo de estudiante y otra para los acuerdos de los instructores:

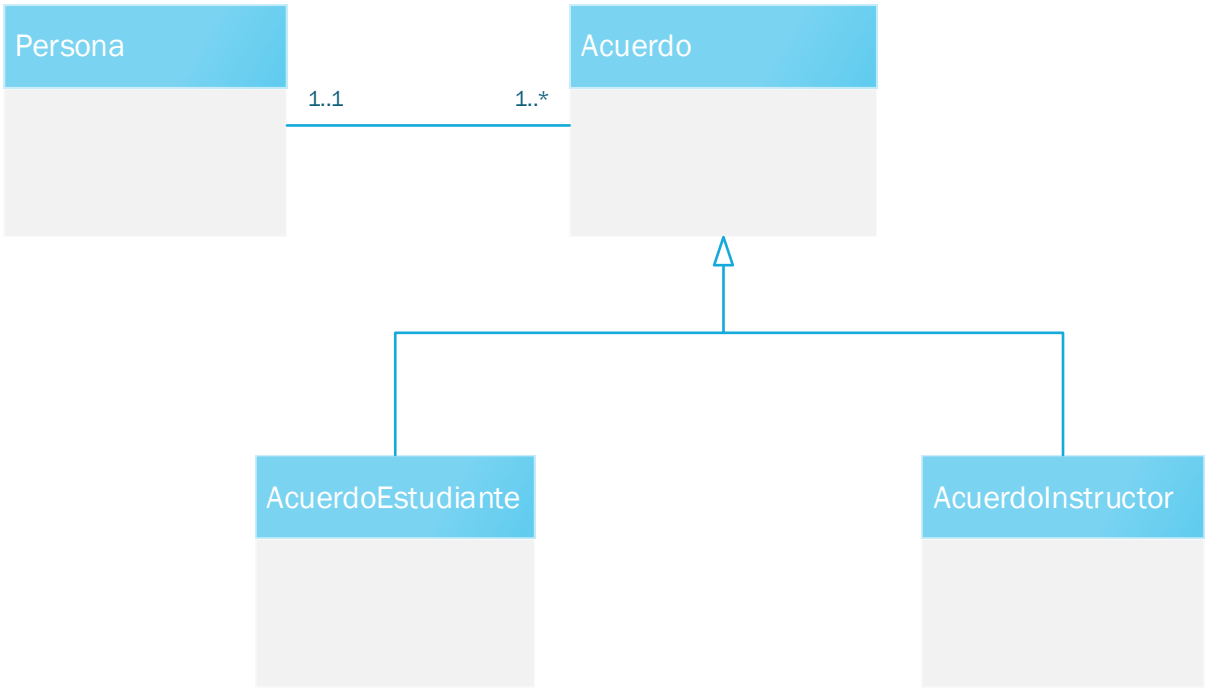


Figura 36 *Implementación para Solucionar el Problema del Diamante*

En cada acuerdo incluiríamos los atributos que se adjudicarán a cada persona dependiendo del rol que vaya a cumplir en el instituto. Adicionalmente como pueden ser varios roles usamos la multiplicidad *uno a muchos*.

7.6 Restricciones de la Generalización y Especialización

7.6.1 Restricción de Participación

La **restricción de participación** define si todas las instancias de una superclase deben pertenecer o no a las subclases. Esta se da o de forma **obligatoria** u **opcional**.

Cuando la participación es obligatoria, todas las instancias de la superclase deben pertenecer a las subclases. Pero cuando se trata de participación opcional, no todas pertenecerán a las subclases definidas.

La restricción de participación opcional se aplica en el caso de los instructores que pueden ser coordinadores de facultad. Aunque exista la subclase Coordinador, no es obligatorio que todos los instructores tengan este rol.

7.6.2 Restricción de Disyunción

La **restricción de disyunción** indica si una instancia de una superclase puede pertenecer solamente a una subclase o a varias a la vez.

Si cada instancia pertenecer a una subclase se le llama relación **disjunta**. Por otro lado, si puede pertenecer a más de una subclase se le dice relación **solapada**.

Un ejemplo de restricción disjunta se ve en la especialización de Instructor en cuanto a la temporalidad de sus turnos, donde una instancia debe pertenecer a InstructorTemporal o a InstructorTiempoCompleto.

Podemos encontrar una relación solapada en la especialización de Curso, la cual genera las subclases CursoOnline y CursoPresencial. Recuerda que un curso esta creado para pertenecer a las dos modalidades.

Notación UML

Puedes representar la restricción de participación ubicando entre corchetes las palabras *Obligatoria* u *Opcional*, debajo del triángulo de la relación.

Dentro de las mismas llaves se ubica la restricción de disyunción. Usa los conectores *O* e *Y* para representar la restricción disyunta y solapada respectivamente.

Veamos el ejemplo de los cursos:

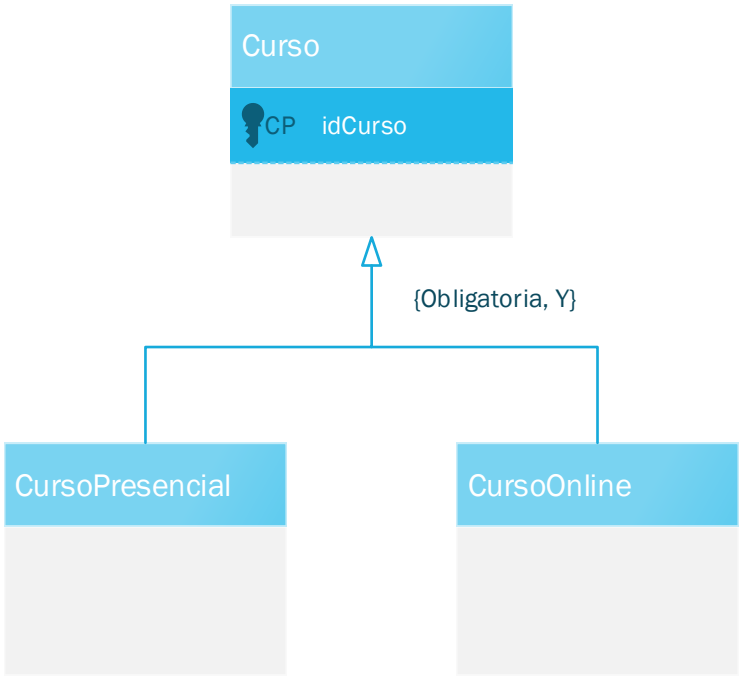


Figura 37 *Ubicación de las Restricciones de Participación y Disyunción en UML*

Representemos también la especialización de coordinadores:

Figura 38 *Restricción de la Relación Instructor/Coordinador*

8. AGREGACIÓN

Una relación de agregación se da cuando una entidad hace parte de otra. Frecuentemente existe cuando se usan relaciones del tipo “tiene”, “es parte de”, “contiene”, etc.

El uso de la agregación permite reemplazar nuestras asociaciones comunes por un nuevo concepto que indica los componentes de una entidad compleja.

Para representarla en el modelo ER se dibuja un rombo en la línea de asociación en el extremo de la entidad contenedora.

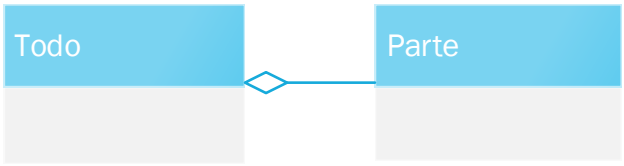


Figura 39 *Agregación en UML*

Por ejemplo, la relación entre Facultad e Instructor es de agregación, ya que conceptualmente una facultad necesita de un instructor para funcionar. Veamos su representación UML:

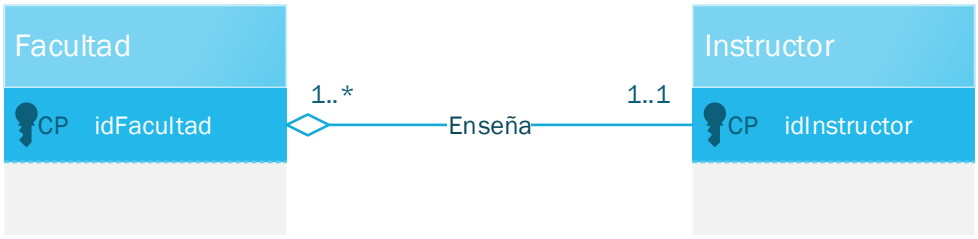


Figura 40 *Agregación entre Facultad e Instructor*

9. COMPOSICIÓN

La composición es una variación de la agregación, solo que esta vez la entidad que representa la parte de la entidad todo está ligada al tiempo de vida de esta. Lo que quiere decir que las partes únicamente existen cuando existe la entidad contenedora.

Su representación es a través de un diamante relleno ubicado en el extremo de la entidad contenedora. Veamos:

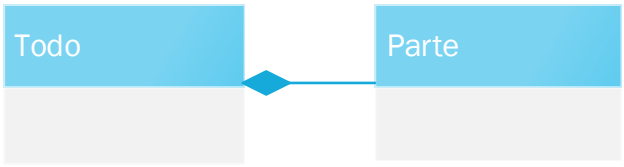


Figura 41 *Composición en UML*

Las composiciones son de gran utilidad para expresar que se desea eliminar o actualizar datos en forma dependiente si es que los requerimientos lo indican.

Un buen ejemplo de ello es la dependencia entre Instructor y HoraTrabajoInstructor. Si el instructor no se encuentra laborando en el instituto no tendría sentido que haya jornadas de trabajo para él. Por eso es posible modelarlo como composición:

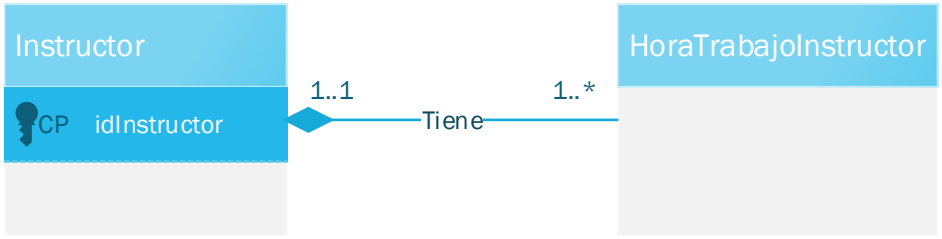


Figura 42 *Composición entre Instructor y HoraTrabajoInstructor*

10. HERRAMIENTAS PARA CREAR DIAGRAMAS ER

Existen aplicaciones que pueden facilitarnos el modelado visual de un diagrama Entidad-Relación con cualquiera de las notaciones existentes. Dependiendo de tus necesidades, bolsillo y filosofía, puedes elegir algunas de las siguientes:

10.1 Lucidchart

Lucidchart es una es una herramienta de diagramado muy simple y potente. Ofrece una gran cantidad de plantillas para distintos tipos de diagramas empresariales, educativos, de desarrollo y modelado de Software, etc.

Su experiencia de usuario es realmente simple, facilitando la conexión entre objetos automáticamente. Aunque podemos usarlo de forma gratuita, nos limita a usar una cantidad de formas en todos los diagramas. Por lo que para maximizar su potencial debemos realizar pagos mensuales.

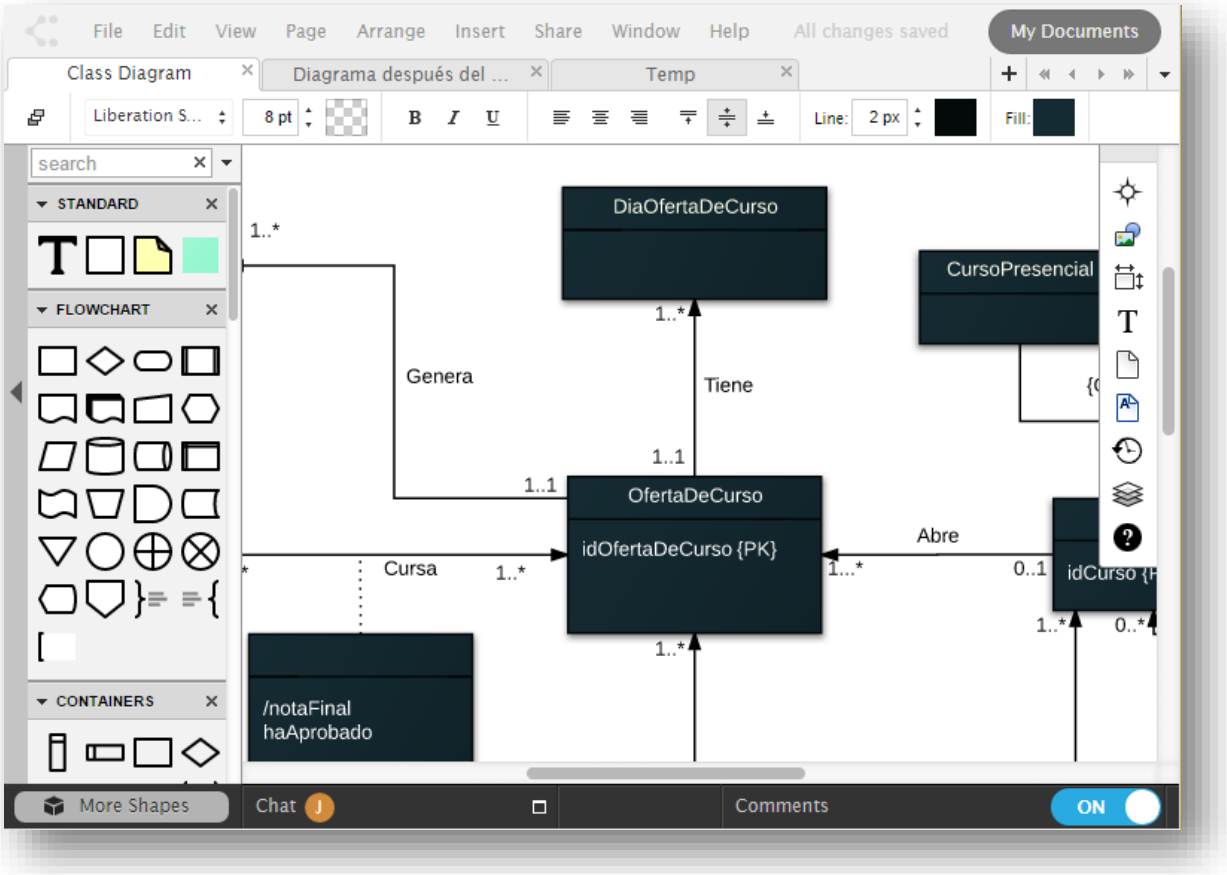


Figura 43 *Screenshot del Lienzo de LucidChart*

En su interfaz veremos un panel de figuras al lado izquierdo, donde podemos arrastrar elementos hacia el lienzo central de la aplicación. Además contamos con una barra de diseño para formato, alineación, y otras características.

Permite exportar nuestros diagramas en varios formatos como PDF, PNG, JPEG, etc. También permite compartir nuestros proyectos con otras personas que usen la aplicación, hasta incluso embeber los diagramas en páginas HTML a través de un código personalizado.

Accede a la aplicación aquí:

https://www.lucidchart.com

10.2 creately

creately es una herramienta de diagramas basada en colaboración entre equipos de trabajo. Es por eso que su modelo de negocio se basa en el pago por cantidad de personas cubiertas y la privacidad de tus diagramas.

Esta aplicación se trabaja sobre la nube y se caracteriza por tener una base de datos con miles de plantillas para el uso de diagramas. Adicional a eso, permite que la comunidad de usuarios comparta sus ejemplos al público y crear retroalimentación entre proyectos.

En su versión gratuita no tenemos restricción sobre la cantidad de figuras que vamos a implementar, si no en la cantidad de diagramas que podemos crear (en el momento que escribo este ebook, son solo 5 diagramas).

Además te permite guardar una historia de las versiones de tus diagramas, característica que es muy útil para documentar los avances como si se tratase de un sistema de control de cambios.

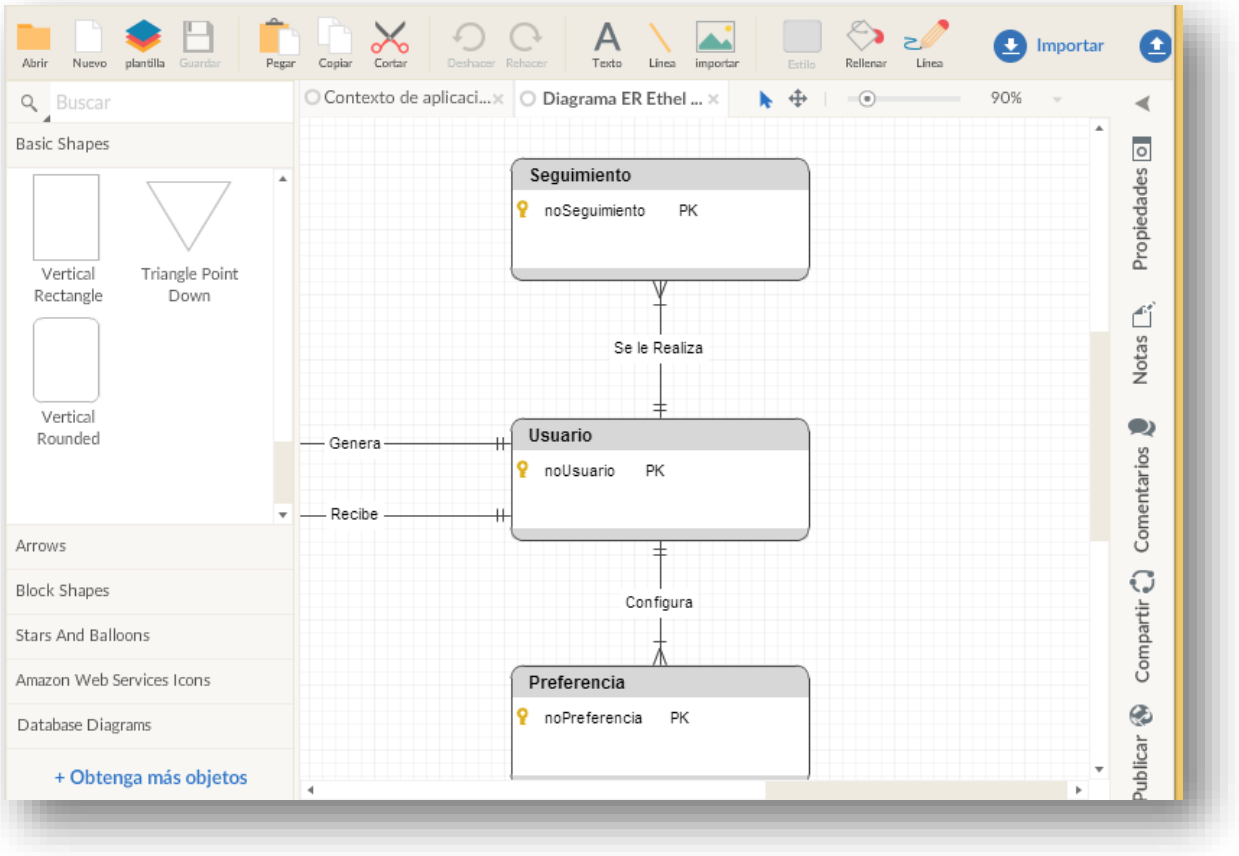


Figura 44 *ScreenShot del Lienzo de Creately*

Su interfaz es muy sencilla e intuitiva, donde predominan los colores sobrios ante todo. Al igual que Lucidchart (y las otras herramientas que observaremos), encontraremos un panel izquierdo donde se alojan las formas que podemos arrastrar al lienzo.

Debido a que Creately tiene infinidad de recursos y plantillas, las categorías de diagramas son muchas, por lo que tendremos acceso a una gran diversidad de figuras.

Ve al siguiente link para su uso:

http://www.creately.com/

10.3 draw.io

Otra opción web muy común es **draw.io**. Este servicio permite crear diagramas de varias áreas profesionales. Nos permite guardar los avances en varios sistemas de almacenamiento en la nube como *Google Drive*, *One Drive* y *Dropbox*. Todo ello sin proporcionar nuestra información personal.

Esta aplicación posee pocas plantillas en comparación a las otras opciones vistas, ya que se construyó como muestra de la implementación de la tecnología **mxGraph** de la compañía *JGraph*. El objetivo era mostrar el potencial de su producto a base de **Javascript** en alguna aplicación. De allí es donde nace draw.io.

Con draw.io tendremos muchas de las características vistas en las herramientas anteriores como exportación en diferentes formatos, historiales de versiones (solo para Google Drive), importación, etc.

En cuanto a pagos, draw.io solo exige el pago para el uso de trabajo en red a través con las herramientas de *Google for Work*, en el caso de que sean más de 10 personas al tiempo.

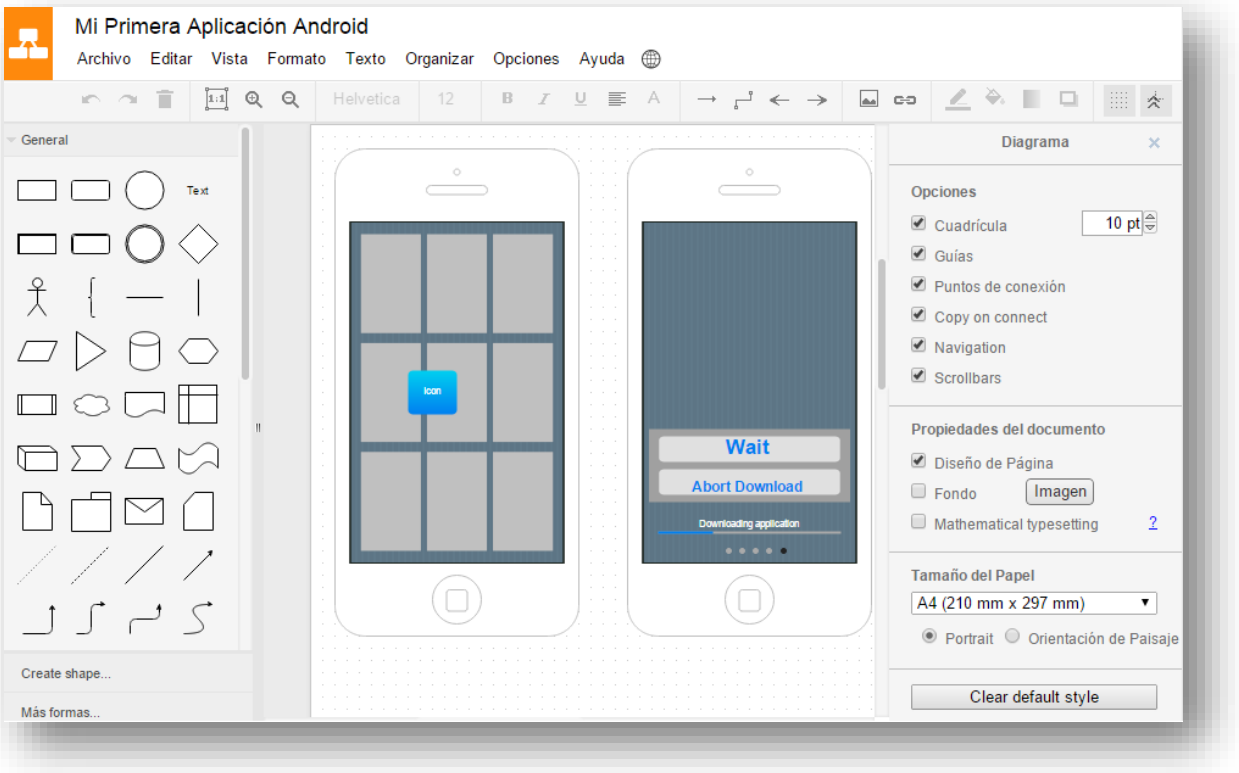


Figura 45 *Screenshot del Lienzo de draw.io*

Puedes utilizar la herramienta en el siguiente sitio:

https://www.draw.io/

10.4 Microsoft Visio Profesional

Como en la mayoría de campos, *Microsoft* aparece con uno de sus ases bajo la manga, **Microsoft Visio Profesional** (usada para este ebook).

Visio es una poderosa herramienta de diagramado con una altísima personalización integrada con las herramientas del paquete *Microsoft Office*.

También posee una fuerte relación con el gestor **SQL Server** para vincular datos dinámicos en los diagramas.

Permite crear diagramas colaborativos donde varias personas pueden realizar cambios, los cuales se sincronizan en tiempo real. Además de poseer un poderoso sistema de comentarios, notas y flujos de trabajo.

Tantas funcionalidades aumenta un poco la curva de aprendizaje de la herramienta si en realidad se le desea obtener el mayor provecho.

Como bien ya sabrás, este producto cuesta por su licenciamiento si alguna vez deseas usarlo con todas sus características. No obstante puedes usar su versión de prueba por un tiempo.

Visio impacta con su gran capacidad de diseño en las formas que se presentan en las plantillas. Usa un estilo muy acorde a las nuevas tendencias visuales para dar un toque profesional a nuestros diagramas.

Para el modelo de datos relacional trae una plantilla muy interesante llamada **Notación de Bases de Datos UML**, donde ajusta la notación para que se acomode al diagrama entidad-relación.

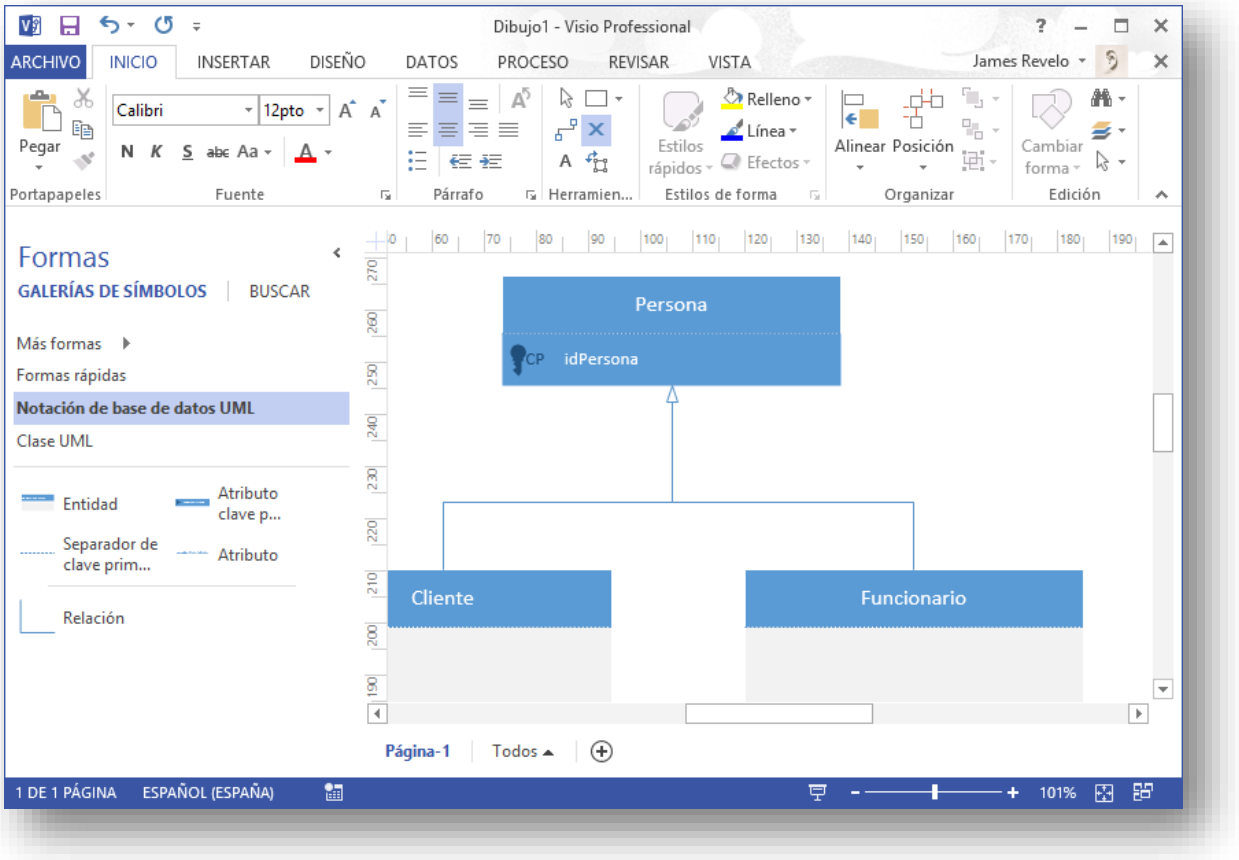


Figura 46 *Screenshot del Lienzo de Visio*

Como ves en la Figura 42, el estilo de las entidades es vistoso y estilizado, algo que puede atraer rápidamente a la mayoría de diseñadores.

Obtén Microsoft Visio 2013 desde el siguiente enlace:

https://products.office.com/es-es/visio/flowchart-software

10.5 Otras Herramientas

A continuación te dejo una lista de herramientas adicionales por si deseas explorar más opciones:

- Gliffy
- Onmigraffle
- Cacoo
- Visual Paradigm

- EdrawSoft
- SmartDraw

10.6 Evaluando La Mejor Solución

La elección que mejor se acomoda a tu situación es desconocida para mí, por lo que no sería coherente recomendarte alguna de las aplicaciones mencionadas. Para solucionar este inconveniente, se creó la siguiente tabla comparativa con algunos factores a tener en cuenta a la hora de adquirir la herramienta apropiada.

El objetivo de la tabla es medir y comparar cada opción, lo que permitirá confrontar tus necesidades con cada métrica. Obviamente elegirás la que más valor aporte a tu negocio y dependiendo de la cantidad de requerimientos que satisfaga.

Para ciertos campos existe una calificación de 1 a 5, donde 5 es excelente, 4 es bueno, 3 regular, 2 malo y 1 pésimo.

Característica	Herramienta			
	Lucidchart	Creately	Draw.io	Visio
Plataforma	Web	Web	Web	Windows
Pago Premium	$21USD/mes 5-10 Usuarios	$50USD/mes 50 Usuarios	$0	$590USD por una licencia en versión *Profesional*
Almacenamiento	25MB	Ilimitado	Ilimitado	Capacidad en Disco
Características de la Versión Gratuita				
Cantidad Usuarios	1	3	10	Depende de la máquina local
Formato Importación	Imágenes, Visio, Gliffy, Onmigraffle	Imágenes, SVG, formato creately, Visio	Imágenes	Imágenes, SVG y DWG
Usabilidad	4	4	5	3
Cantidad de Plantillas	Media	Alta	Baja	Media
Formato Exportación	PDF, PNG, JPEG y Visio	Formato creately, SVG, PDF, PNG, JPEG,	PNG, GIF, JPEG, PDF, SVG, HTML, XML draw.io	PNG, JPEG, EMF, SVG, DWG, HTML, PDF, XPS
Limitación especial	Se permiten solo 60 formas por diagrama	Sin Privacidad	NA	60 días de prueba.
Calidad Visual	3	4	2	5

Tabla 2 *Evaluación de Herramientas para Diagramado*

PARTE 3
METODOLOGÍA PARA EL DISEÑO CONCEPTUAL DE BASES DE DATOS

INTRODUCCIÓN

Como vimos al inicio del libro, el diseño conceptual es la primera etapa del diseño de una base de datos, donde se representa la realidad de la información con un enfoque global.

En él se abordan los requerimientos de un sistema, empresa, cliente, usuario, etc., para crear una representación de alto nivel que permita comprender la visión general y el alcance del manejo de la información.

Pero… ¿Por qué un diseño conceptual?

- Permite abordar formalmente el problema de la administración de datos.
- Mide el entendimiento de los diseñadores y desarrolladores del problema.
- Permite estandarizar el vocabulario a utilizar hacia el problema.
- Concientiza al equipo desarrollador que primero se piensa en las condiciones actuales y luego si se piensa en soluciones.

Esta sección contiene la explicación de una serie de pasos secuenciales que me han sido de gran utilidad en los proyectos que he desarrollado a lo largo de mi carrera profesional.

Afortunadamente siempre habrá varias opciones o variantes para diseñar bases de datos. Todo depende de tu filosofía personal y laboral, ya que eres tu quien elige si seguir al pie de la letra los pasos, saltarte algunos, modificar el orden y demás.

Básicamente nos apoyaremos con el **Diagrama Entidad-Relación** con notación UML visto en la sección anterior y otro documento llamado **Diccionario de Datos**. Ambos se complementan supremamente bien, facilitándonos el camino para emprender un excelente modelo lógico.

A continuación se listan los pasos de la metodología a seguir:

Paso 1 *Identificar Entidades*

Paso 2 *Identificar Relaciones*

Paso 3 *Identificar Atributos de Cada Entidad y Relación*

Paso 4 *Definir Llaves Candidatas, la Llave Primaria y las Llaves Alternativas*

Paso 5 *Decidir si es Conveniente Usar el Modelo EER*

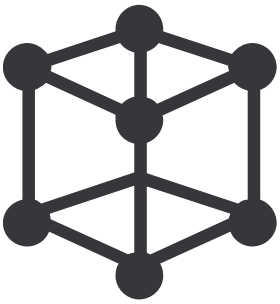
Paso 6 *Inspeccionar el Cumplimientos de las Transacciones del Usuario*

Paso 7 *Añadir Validaciones Temporales al Esquema*

Paso 8 *Revisar el Modelo Conceptual con el Usuario*

> Puedes encontrar un checklist de los pasos y sus respectivas tareas en el archivo *"Checklist para el Diseño Conceptual de Bases de Datos.docx"*.

PASO 1 IDENTIFICAR ENTIDADES



En primera instancia debemos identificar todas las entidades en las cuales están interesados nuestros usuarios. Lo ideal es conseguir aquellas entidades que sean prioritarias para el negocio y tengan mayor evidencia, ya que el objetivo del diseño conceptual inicial es conocer y no resolver.

Es posible partir de un modelo de datos que comprenda la lógica necesaria que sirva para la comprensión general del problema. A medida que se avance en el diseño y haya un conocimiento mayor, entonces podrán identificarse más entidades.

Técnicas de Búsqueda de Entidades

A continuación te dejo una lista de las técnicas más efectivas para encontrar entidades en los requerimientos de un sistema de información:

- **Extracción de sustantivos**: Esta es la forma más fácil e intuitiva para encontrar entidades. Se basa en investigar la *Lista de Requerimientos* si usas metodologías tradicionales o en el *Product Backlog* (si usas metodologías ágiles), en búsqueda de frases que involucren distinciones de objetos vitales para el comportamiento del negocio.
- **Cantidad de instancias**: Cuando desees saber si un sustantivo es una entidad pregúntate: *¿Existirán muchos <sustantivo> en el problema?*, por ejemplo para identificar si el sustantivo cliente es una entidad nos preguntamos: ¿Existirán muchos clientes en el problema? La respuesta es sí, por lo que Cliente es una entidad.
- **Comienza por lo tangible:** Una excelente opción es identificar las entidades tangibles que puedan existir en la lógica del negocio como el inventario, las personas, las oficinas, cargos, dispositivos, herramientas, etc. Cognoscitivamente este tipo de entidades son más fáciles de percibir, además de que son la base del soporte de un negocio o actividad como tal.
- **Cantidad de atributos:** Si en algún momento llega a ser confuso si un concepto es una entidad o no, puedes decidir tomando en cuenta si la

cantidad de atributos que intervienen es considerablemente importante, ya que puede ser que simplemente se esté hablando de una relación o inclusive de atributos comunes.

- **Crea un prototipo estático:** Crear una serie de formularios estáticos también es una excelente opción para identificar que entidades deben ser almacenadas desde el punto de vista del usuario. Con la interfaz puedes estimular la imaginación del usuario para que despliegue todas sus ideas e incluso requerimientos ocultos (Visual Studio y NetBeans son herramientas que pueden ayudarte).

Añadir Entidades al Diccionario de Datos

El diccionario de datos es un artefacto que se crea en el diseño conceptual para documentar las entidades, relaciones y atributos que existen en el modelo de datos.

Esto con el fin de ayudar a estandarizar el vocabulario y la interpretación de los datos entre el equipo.

Por lo general utilizo una aplicación propia para el almacenamiento del diccionario de datos, pero no existe ningún inconveniente en emplear *Microsoft Office Excel* para esta tarea. Por lo que he construido una plantilla para que la uses en tus proyectos.

> Puedes encontrarla en los archivos que venían con este ebook (Ver archivo "*Plantilla de Diccionario de Datos.xlsx*").

Nombre de la Entidad	Descripción	Alias
Estudiante	Cada uno de las personas que pagan para estudiar en el instituto	Alumno, Aprendiz
Instructor	Personas que enseñan las materias de cada carrera en el instituo	Profesor, Maestro, Docente
Coordinador	Es un instructor que se le otorgaron las responsabilidades de coordinador de una facultad	Director, Decano
Facultad	Subdivisión de los saberes que se imparten en	-

Figura 47 *Ejemplo de Entidades en el Diccionario de Datos*

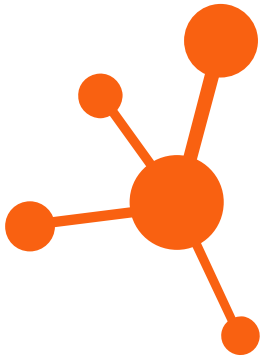
Esta plantilla consta de tres hojas, una es para las entidades, otra para las relaciones y la tercera es para la descripción de los atributos.

Para documentar las entidades se usan datos como el *Nombre de la Entidad*, donde ubicamos el sustantivo que elegimos para representar a la entidad.

Otro campo importante es la *Descripción* que tiene para el negocio, la cual debe ser definida con ayuda del cliente y el equipo desarrollador.

También se documentan los *Alias* que pueda llegar a tener por si existen otros nombres posibles para la entidad. Con esto sabremos que sinónimos existen y que opción es la más apropiada.

PASO 2 IDENTIFICAR RELACIONES



Las relaciones entre entidades son intuitivas si vemos la conexión gramatical dentro de los requerimientos o historias de usuario.

Pero una excelente forma de encontrar relaciones es analizar las siguientes fuentes de información:

- Sistema de Gestión de Calidad
- Manual de procesos
- Formatos de entrega
- Formatos de salida
- Listas de administración
- Reportes de área
- Archivos de Excel
- Formularios del sistema de información que se esté intentando reemplazar.
- Bases de datos del sistema actual.

Todas estas herramientas contienen datos en su estructura, donde puedes ver como se mezclan las entidades en ellos. Una vez analizado esto, es casi que seguro que las entidades que intervengan están relacionadas de alguna manera.

Un ejemplo muy común es la factura generada por el área financiera de las empresas. Veamos el siguiente formato de la ferretería *La Más*.

Ferretería La Más

Cra 2C #23 -56
Cali

19-Marzo-2015
Factura #2334889
PO 456001200

(123) 456 789
soporte@lamasferre.com

Cliente: Sra. Carla Gomez
ID C0034

Querida Sra. Carla Gomez,

Asegúrese que la información de los detalles de su factura sea veraz y congruente con su pedido actual. Cualquier inquietud puede comunicarla al numero 01 8000 92838

Muchas Gracias,

Att:
James Revelo
Director Comercial

#	Descripción	Unidades	Precio Unitario (€)	Total (€)
1	Tornillos de media	20	125.00	2500.00
2	Tablas	12	34.00	408.00
Subtotal				2908.00
IVA (16%)				465.28
Total				3373.28

Figura 48 *Ejemplo de Factura para la Identificación de Relaciones*

Como ves, existen datos sobre la seccional de la ferretería, datos del cliente, datos del director comercial y datos sobre los artículos vendidos.

¿Qué puedes deducir a vuelo de pájaro de este documento?

¡Exacto!, un cliente puede generar una factura, por la compra de unos artículos del inventario existente en la seccional de la ferretería.

De esa sentencia se pueden abstraer varias relaciones que alimentará el modelo de datos.

Tarea 1. Hallar la Multiplicidad de las Relaciones

Cuando halles las posibles relaciones, se determina la multiplicidad a través de la participación y cardinalidad de entidad.

Recuerda que esto lo vimos en el *Apartado 2*, donde se indicaba si todos los elementos se relacionaba y la cantidad máxima de elementos en la entidades que se comprometían en la relación.

Tarea 2. Verificar posibles fallas

En este paso verificamos si existen trampas ventilador y trampas de agujero como se estudió en el *Apartado 6*.

Tarea 3. Añadir al Diccionario de Datos

En la segunda hoja de nuestra plantilla del diccionario de datos podemos documentar las relaciones que existen por cada entidad identificada.

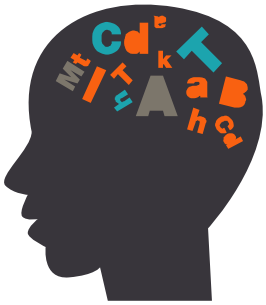
Ubicaremos en frente de cada una de las entidades la *Multiplicidad* que posee con respecto a la entidad que se relaciona. La entidad del otro extremo también debe incluir su multiplicidad.

Veamos un ejemplo de algunas relaciones:

Nombre de la entidad	Multiplicidad	Relación	Multiplicidad	Nombre de la entidad
Estudiante	1..1	Cursa	1..*	OfertaCurso
Instructor	1..1	Cumple	1..*	DiaTrabajoInstructor
	1..1	Dicta	1..*	OfertaCurso
Coordinador	1..1	EsUn	0..1	Instructor
	1..1	Supervisa	0..*	Instructor
	1..1	Coordina	1..1	Facultad
Facultad	1..1	Acoge	1..*	Estudiante
	1..1	SeEncuentra	1..*	Curso
	1..1	Enseña	1..*	Instructor
	0..*	EsPrerrequisito	0..*	Curso

Figura 49 *Ejemplo de Relaciones en el Diccionario de Datos*

PASO 3 IDENTIFICAR ATRIBUTOS DE CADA ENTIDAD Y RELACIÓN



Para determinar los atributos de una entidad o relación es importante repasar de nuevo los requerimientos del usuario.

Puedes interrogar a los usuarios por los datos que desean almacenar para el funcionamiento del negocio.

Si la empresa tiene un sistema de gestión de calidad o un manual de procesos establecido, tendrás el trabajo mucho más fácil, ya que existen estándares, categorías y un lenguaje formal para cada operación.

Esto es una ventaja, ya que podrás extraer los datos precisos sin caer en ambigüedades o exigencias subjetivas.

¿Cómo Obtener los Atributos de una Relación?

Para identificar los atributos de una relación hazte la siguiente pregunta:

> *¿Qué características aparecen solo si las entidades actúan en conjunto?*

Por ejemplo, las notas que tiene un estudiante surgen solo porque existe una interacción entre Estudiante y OfertaCurso. Este atributo es el resultado de la interacción conjunta.

Tarea 3.1. Diferenciar entre Atributos Simples y Compuestos

Esta clasificación es supremamente vital para la identificación de los atributos. Debes investigar si es necesario que un atributo sea simple o se mantenga compuesto.

Ejemplos populares de esta situación es el nombre de una persona y su dirección. Como sabes, el nombre está compuesto por nombres y apellidos, pero dependiendo de las acciones que se quieran realizar con esos componentes así mismo se decidirá si es un atributo compuesto o un atributo simple.

Puede que una aplicación comercial desee filtrar a sus clientes por el primer nombre o el segundo apellido, por lo que es necesario considerar al nombre como atributo compuesto.

O quizás una aplicación móvil simplemente necesite mostrar el nombre completo de los contactos en una lista sin tener en cuenta esta separación. En consecuencias nombre sería un atributo simple.

Tarea 3.2. Diferenciar entre Atributos Monovalorados y Multivalorados

En el apartado 3 vimos que la mayoría de atributos contienen un solo valor para una instancia (monovalorados), sin embargo pueden ser multivariados al contener múltiples valores por instancia.

Un ejemplo de esta situación es el número de teléfono de los estudiantes como se había visto. Donde se discutía que un estudiante puede tener al menos un teléfono de contacto y máximo 3.

Esta situación también sucede con los emails de una persona, la cual puede tener un email principal, uno alternativo o tal vez uno corporativo.

Tarea 3.3. Identificar atributos derivados

Los atributos derivados son aquellos que se calculan con base en otros como se vio en la sección 3.3. Aunque en el modelo conceptual aún no es posible decidir si un atributo de este tipo va a ser almacenado o no en la base de datos, es importante que conservemos la congruencia con la interpretación del problema y escribirlo en la entidad.

Uno de los ejemplos más frecuentes de atributos derivados es la edad de un estudiante, el cual se calcula a través de la fecha de nacimiento.

La nota final de un estudiante para un curso también es un atributo derivado, ya que se computa a través de todas las notas obtenidas hasta el momento y el peso que contenga cada una.

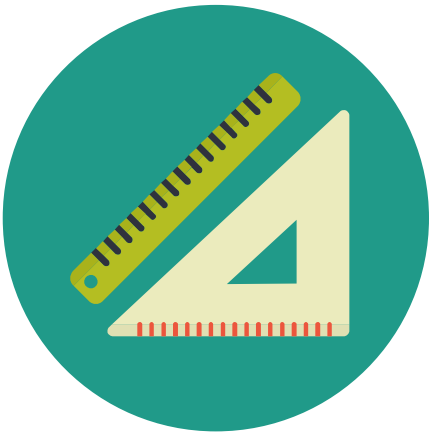
Tarea 3.4. Determinar el dominio de los atributos

Esta tarea consiste en definir cuáles serán los valores que contenga el dominio de cada atributo. Esto permite limitar los valores, el formato y el tamaño, lo que será de gran ayuda en la implementación de la base de datos.

En este apartado puede ayudarnos el entendimiento de las reglas de negocio.

Reglas de Negocio

Las **reglas de negocio** son restricciones que tiene cada operación de un negocio para funcionar.



Dentro de ellas encontramos lineamientos como el cargo responsable de ejecutar cierta acción, los horarios para ejecutar actividades, el valor del descuento que se realiza a un cliente, los tiempos necesarios para atender al cliente, las normativas necesarias para facturar, etc.

El dominio de un atributo es un tipo de regla de negocio llamada **restricción de cordura**, ya que limitamos los posibles valores según la percepción normal del entendimiento como el manejo de números mayores que 0, el uso exclusivo de caracteres numéricos dentro de un teléfono e infinidad de casos.

Por ejemplo, en Colombia las empresas de telecomunicaciones deben usar un identificador para el servicio al cliente de sus usuarios llamado **CUN** (Código Único Numérico).

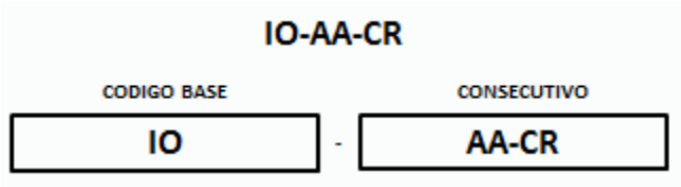


Figura 50 *Formato del **CUN** establecido por la Superintendencia de Industria y Comercio Colombiana*

Este número tiene un formato de tres segmentos separados por guion como se muestra en la Figura 46.

Donde el primer componente son los últimos cuatro dígitos del identificador del operador, el segundo segmento es los últimos dos dígitos del año donde se le realizó el servicio al cliente. Y el tercero es un consecutivo de 10 números, el cual inicia en 0000000001 desde el primer día del año.

Un ejemplo sería:

1266-15-0000000004

Como ves, este formato debe ser documentado para que la implementación tenga en cuenta su estructura y sepan que el dominio es el conjunto de todos los números que cumplan con el formato *IO-AA-CR*.

Las reglas de negocio incluyen muchas más tipos de restricciones que ayudarán a optimizar el diseño de una base de datos. Pero en este ebook no profundizaremos en este tema, ya que la aplicación completa de este concepto se ve en el diseño lógico.

Tarea 3.5. Añadir al Diccionario de Datos

Ve a la hoja número 3 del diccionario de datos y añade para cada entidad de la lista los atributos que se han identificado hasta el momento.

Teniendo en cuenta lo visto en este paso, deberá diligenciar los siguientes campos:

- *Nombre*: Nominación del atributo. Los atributos compuestos se subdividen en una jerarquía diferenciada con una tabulación.
- *Descripción*: Se refiere al significado del atributo para el negocio.
- *Alias*: Son los sinónimos que pueda tener el atributo.
- *Tipo* de dato: Es el tipo de dato del atributo
- *Tamaño*: Cantidad de caracteres que usarán los valores del atributo.
- *Nulo*: Identifica si el atributo permite valores nulos o no.
- *Multivalorado*: Determina si es multivalorado o no.
- *Derivado*: Indica la fórmula de obtención en caso de ser derivado.
- *Valor* por defecto: Es un valor predeterminado que se usará en caso de que no se especifique directamente un valor.
- *Dominio*: Especifica el dominio del atributo. Puedes hacer uso de las notaciones de conjuntos por extensión y comprensión.

A continuación se muestra un fragmento de los atributos de la entidad Facultad:

Entidad/Relación	Atributos	Descripción	Llave Primaria	Tipo de dato	Longitud	Valor por defecto	Nulo
Facultad	idFacultad	Identificador único que asigna el instituto a la facultad	SI	carácter	2	NO	NO
	nombre	Nombre de la facultad	NO	carácter	15	NO	NO
	oficina	Oficina donde esta ubicada la facultad	NO	carácter	3	NO	NO
	telefono	Número telefónico para comunicarse con la facultad	NO	carácter	10	NO	NO
	secretaria	Secretaria que atiende los asuntos de la facultad	NO	carácter	15	NO	NO

Figura 51 *Ejemplo de los Atributos de la entidad Facultad en el Diccionario de Datos*

Es válido extender la plantilla. Recuerda que una metodología no es un plan riguroso. Si observas que existen elementos innecesarios para tu forma de trabajo o por lo contrario, identificas que faltan elementos, entonces puedes actuar libremente para modificarla.

Lo importante es que encuentres tu espacio de acción acorde a tu visión y la de tu equipo.

PASO 4 DEFINIR LLAVES CANDIDATAS, LA LLAVE PRIMARIA Y LAS LLAVES ALTERNATIVAS

Al tener todos los atributos de las entidades definidos, es posible seleccionar aquel que pueda diferenciar a todas las instancias, es decir, la llave primaria.

Todos aquellos atributos que puedan ser la llave primaria se le llaman **llaves candidatas**. Cuando ya seleccionaste la llave primaria, las llaves candidatas restantes pasan a ser **llaves alternativas**.

Pero... ¿Qué criterios seguir para elegir la mejor llave candidata?

A continuación te comparto una serie de factores para identificar la mejor opción:

Integridad

¿La llave candidata permite valores nulos?

Esta es una pregunta que no puede tener respuesta positiva a la hora de elegir una llave primaria. Esto haría perder la integridad referencial de cualquier base de datos.

Afortunadamente los gestores de bases de datos nunca dejarán que esto pase, ya que trae restricciones que aseguren que los valores de una llave primaria siempre existan.

Manejo

¿La llave candidata es solo un atributo?

Es preferible tener un solo atributo que represente la llave primaria a tener una llave compuesta por varios atributos.

Aunque conceptualmente esto no afecta de ninguna forma, al momento de la creación de consultas SQL las operaciones pueden ser demoradas, debido a que existen muchas más comparaciones.

Simplicidad

¿Los valores de la llave candidata contienen formatos especiales como guiones bajos, espacios, caracteres especiales o letras capitales?

Los valores de una llave primaria deben ser lo más simple posible. Evita elegir atributos que tengan caracteres especiales, espacios, letras capitales, etc.

La mayoría de búsquedas se realizan a partir de la llave primaria por lo que sería una pérdida de tiempo escribir valores como “ASD-123G 22” o “Registro dE la Persona001”.

Velocidad

¿El tipo del valor de la llave candidata es distinto de entero o carácter de tamaño fijo?

En la computación los enteros son un tipo de dato que pueden ser procesados muy rápido. Por lo que vendría bien que tu llave primaria sea de este tipo de dato.

Como segunda opción se podría intentar conseguir una llave primaria de tipo carácter fijo. Aunque se procesan rápido, tienen la desventaja de que deben convertirse a su equivalente **ASCII** primero, lo que representa tiempo adicional.

Inmutabilidad

Lo ideal es que los valores de una llave primaria nunca cambien porque representan la identificación de una instancia en toda la base de datos.

El solo hecho de cambiar un valor puede repercutir en todo el esquema como si se tratase de una cadena. Esto se traduce en un trabajo exhaustivo de modificaciones de todos los registros relacionados con esa llave primaria.

Tamaño

¿Los valores de la llave candidata son muy grandes?

Elige la llave candidata con el mínimo de caracteres en sus valores si son caracteres o el mínimo consumo de computo sin son enteros.

Quiere decir que es mejor una llave de 4 caracteres que una de 6. O una llave de 2bytes enteros a 4 bytes.

Familiaridad

¿Es fácil de usar para los usuarios?

El usuario es la persona que al final administrará la información en un alto nivel. Son ellos quienes terminan realizando búsquedas a través de los criterios que mejor les parezca.

La llave primaria debe ser intuitiva y fácil de usar para estos, ya que son los únicos que conocen el día a día de la operación del negocio. En consecuencia es necesario incluirlos en el diseño para debatir el alcance de la elección.

Aquella llave candidata que cumpla todas esas condiciones será una opción potencial para llave primaria.

> Puedes usar la plantilla *"Checklist para Identificar Llave Primaria.xlsx"* donde encontrarás estos criterios resumidos.

Llave suplente

En el peor de los casos, donde no exista la llave candidata que cumpla los criterios anteriores, entonces la solución será crear una **llave suplente**.

Una llave suplente es una llave primaria artificial, generada con el propósito de diferenciar a nuestras entidades debido a que no existe una llave candidata adecuada.

De otro lado a las llaves primarias que cumplieron con los criterios se les llaman **llaves naturales** o **llaves del negocio**.

Por ejemplo, si una entidad tiene 1000 instancias este año y de acuerdo a los últimos años se muestra un crecimiento anual promedio del 20%, se pronostica que habrá 38340 instancias dentro de 20 años.

Una de las primeras ideas para crear una llave suplente de esa entidad es elegir un entero autoincrementable desde 00001 hasta 99999.

Otra idea sería elegir una cadena con dos caracteres al inicio y una serie numérica el final de dos dígitos, como por ejemplo “AF23”. Lo que representa 72172 posibles valores.

Crear una llave suplente debe ser un trabajo conjunto entre el diseñador y el usuario.

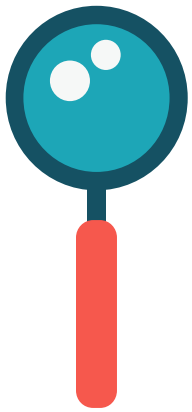
Añadir Llave Primaria al Diccionario de Datos

Para diferenciar las llaves primarias en el diccionario de datos puedes crear una nueva columna para marcar el atributo o atributos que sean la llave primaria. Incluso la celda puede ser marcada con algún color distintivo.

Entidad/Relación	Atributos	Descripción	Llave Primaria	Tipo de dato	Longitud	
OfertaDeCurso	idOfertaCurso	identificador numérico asignado a la oferta de un curso	SI	carácter	5	NC
	salon	Lugar donde se dicta el curso	NO	carácter	3	NC

Figura 52 *Identificación de una Llave Primaria en el Diccionario de Datos*

PASO 5 DECIDIR SI ES CONVENIENTE USAR EL MODELO EER



Los conceptos del modelo entidad-relación extendido son opcionales en la definición conceptual de un problema.

Recuerda que es posible usar los enfoques de generalización, especialización, agregación y composición para expresar relaciones estructurales entre las entidades.

Por ejemplo, usamos especialización en la entidad Curso, donde se abstrajeron las entidades CursoOnline y CursoPresencial. Esto con el fin de comprender extensivamente el panorama del caso de estudio.

Pero... ¿Cuándo es conveniente usar los conceptos del modelo entidad-relación extendido?

Eso puedes saberlo si en algún momento percibes que:

1. Algunas instancias tendrán un valor para ciertos atributos pero para otros no.
2. Solo algunas instancias de una entidad tendrán una relación con las instancias de otra entidad.
3. Dos entidades comparten la mayoría de sus atributos en común.
4. La precisión de los datos es necesaria en el modelo. No uses relaciones superclase/clase para todo, mídete en el uso de estos conceptos y solo impleméntala si es estrictamente necesario.

Casos donde No se debe Usar Herencia de Atributos

Evitar usar herencia cuando las diferencias existen en los valores de los atributos y no en la instancia como tal.

Si un cliente vive en México y otro en España, no por eso vas a crear una especialización (Ver Figura 51). Simplemente el atributo que representa su ubicación tiene valores distintos.

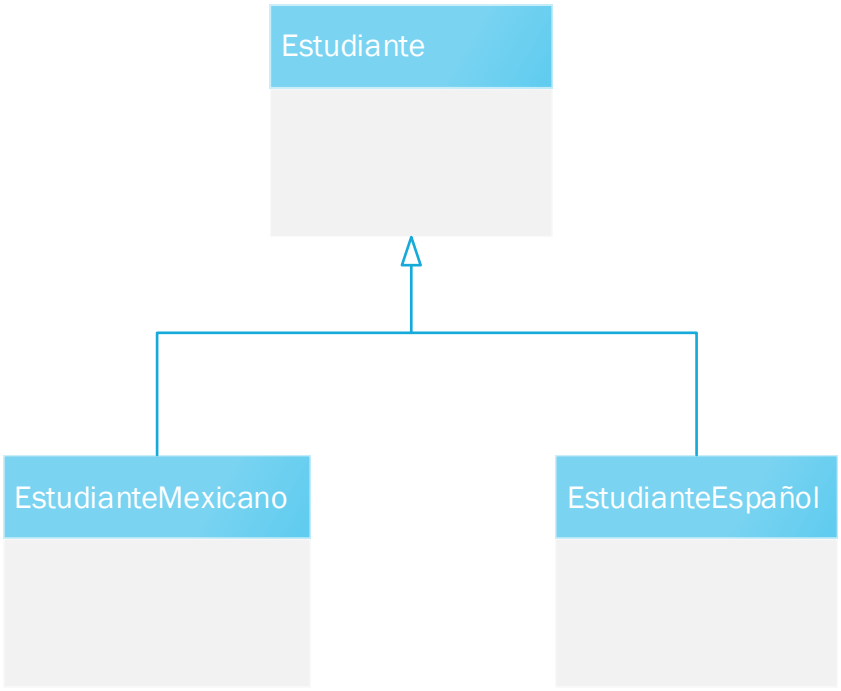


Figura 53 *Modelamiento Erróneo por Confundir Instancias con Atributos*

PASO 6 COMPROBAR EL CUMPLIMIENTO DE LAS TRANSACCIONES DEL USUARIO

Este paso consiste en verificar que las consultas que se identificaron en la fase de análisis puedan representarse con el modelo de datos que se acaba de construir.

Para realizar la comprobación puedes usar una flecha que represente la ruta entre las entidades que describen la transacción. Puedes incluir el identificador de la transacción justo al lado.

En el siguiente diagrama se muestra la comprobación de la transacción 1 vista en el caso de estudio:

1. *Obtener una lista todos los cursos vistos por un estudiante.*

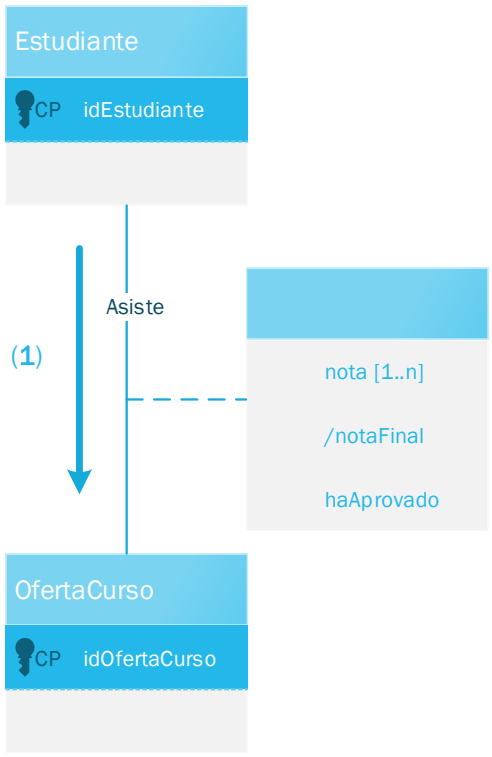


Figura 54 *Técnica de Comprobación de Consultas a través de Rutas entre las Entidades*

Esta técnica no es obligatoria. Si deseas solo puedes recorrer con un dedo la ruta que satisface cada transacción y ser consciente de la calidad de tu modelo.

PASO 7 AÑADIR ANOTACIONES TEMPORALES AL ESQUEMA

Este paso se fundamenta en el enfoque de bases de datos basadas en los cambios temporales, donde se analizan las posibles variaciones de los elementos del modelo de datos a través del tiempo.

Hasta el momento hemos creado nuestro modelo de datos viendo estáticamente las entidades, relaciones y atributos. Por lo que tenemos una foto del estado actual. Pero en la vida real estos objetos pueden variar de acuerdo a la lógica del negocio.

Es por eso que se debe crear un plan de contingencia que registre los cambios vitales, para mantener funcional el modelo de datos.

La implementación de las características de una **base de datos temporal** depende de las necesidades del cliente. El será quien establece en sus requerimientos si desea realizar seguimiento a sus datos a lo largo del tiempo y la precisión que necesita.

Instante, Intervalo y Periodo

Los bloques de tiempo que caracterizan la temporalidad de un elemento dentro de la base de datos, permiten determinar las acciones a realizar sobre estos.

El lapso básico para medir la existencia de un elemento se llama **instante**. Un instante es un espacio de tiempo que se queda anclado en la línea temporal, es decir, sucede una sola vez y luego se considera enterrada para siempre en el pasado. Esto hace que su duración sea considerada 0.

Los siguientes ejemplos de instantes son:

- Tú fecha de nacimiento.
- La fecha de muerte de Steve Jobs (5 de octubre de 2011).
- La batalla de Boyacá un 8 de agosto de 1819.

Un **intervalo** es una porción de tiempo contigua que no está atada a la línea de tiempo. Los intervalos son relativos en el tiempo, tienen un punto de

inicio, un punto de terminación y una dirección. Con dirección me refiero a que pueden ir en incremento o decremento.

Un ejemplo sería las 24 horas que se debe demorar la obtención de una solución por Servicio al cliente por parte del *Instituto San Judas*. Este va en decremento desde 24:00:00 hasta 00:00:00.

Un **periodo** comprende un espacio de tiempo anclado en la línea temporal y además tiene una duración considerable. Por ejemplo, el primer semestre de 2014 del *Instituto San Judas* comenzó el 2 de Enero y terminó el 7 de Mayo.

Como ves, este periodo tiene un punto inicio temporal (2 Enero) y un punto terminal (7 de Mayo), pero igual quedó rezagado por el pasado.

Granularidad

La **granularidad** es una unidad discreta en la línea de tiempo, ya sea un segundo, un minuto, una hora, un día, semana, año, etc.

Este término lo usaremos para indicar la progresión de tiempo con que varían los elementos del esquema de un modelo de datos.

> Puedes leer más sobre este tema en el siguiente ebook gratis de *Richard T. Snodgrass* llamado *Developing Time-Oriented Database Applications in SQL*:
>
> http://www.cs.arizona.edu/people/rts/tdbbook.pdf

Tarea 7.1. Identificar la Esperanza de Vida de Cada Entidad

Cada entidad tiene una **esperanza de vida** que expresa en que momento comenzó a existir y hasta cuando existirá. Normalmente la esperanza de vida se identifica con las medidas de instante, intervalo y periodo.

Por ejemplo, la condición de *cotizante activo* de una entidad promotora de salud comienza en el momento

que un cliente realiza su primer pago. Pero esta condición ya no estará vigente si en algún momento cambia de empresa.

Ahora haz de cuenta que tal vez en el régimen de seguridad social de tu país, los pacientes deben permanecer por lo menos un año en su entidad promotora de salud. Por lo se deduce que la esperanza de vida de Cotizante tiene una granularidad de 1 AÑO como mínimo.

Anotaciones Temporales

Para registrar la esperanza de vida de las entidades creamos una tabla que contenga el nombre de la entidad, la granularidad y los comentarios pertinentes. En la Tabla 3 se ve el tiempo valido para cada una de las entidades.

Nombre de la Entidad	Granularidad de la esperanza de vida	Comentarios
Persona	INDEFINIDO	Es necesario realizar un seguimiento de entrada y salida a los individuos que pertenecen a la institución. La entidad es temporal pero su granularidad es variable.
Estudiante	5 AÑOS	La condición de estudiante comienza el día en que se matricula. Su trayectoria ideal para terminar su profesión dura 5 AÑOS aproximadamente. Cabe aclarar que un estudiante en cualquier momento puede abandonar la institución. Además puede atrasarse en sus materias por razones personales, reintegrase y continuar.
Instructor	INDEFINIDO	Aunque la vida de un instructor está atada a una temporalidad, esta es indefinida por su contrato. Sin embargo pueden ser despedidos en cualquier momento.
Coordinador	No temporal	El Instituto no está interesado en hacer seguimiento de la esperanza de vida de los coordinadores.
Facultad	No temporal	Esta entidad no se ve afectada por los cambios del tiempo.

Curso	No temporal	Los cursos duran a lo largo del tiempo. En ocasiones particulares pueda que no se vuelvan a abrir al público, pero no se requiere su seguimiento.
CursoPresencial	No temporal	Se desprende de una entidad no temporal
CursoOnline	No temporal	Se desprende de una entidad no temporal
OfertaDeCurso	4 MESES	Una oferta de curso tiene una vida limitada, por lo que se desea realizar un seguimiento al año en que se dictó y al semestre que perteneció.
DiaOfertaCurso	4 MESES	Esta entidad hace parte de OfertaCurso, por ende su esperanza de vida es igual.
HoraTrabajoInstructor	INDEFINIDA	Esta entidad es parte de Instructor. Aunque representa un intervalo de tiempo para un día general de la semana, se repite constantemente.

Tabla 3 *Anotaciones Temporales de las Entidades*

Tarea 7.2. Identificar el Tiempo Válido para las Relaciones

En este caso buscaremos si las relaciones ocurren instantáneamente o tienen una duración determinada.

Lógicamente el tiempo válido de una entidad debe estar contenido entre la intersección de las esperanzas de vida de las relaciones que sostienen la relación.

No sería congruente que la entidad *A* tenga esperanza de vida de 5 minutos, la entidad *B* tenga esperanza de vida de 10 minutos y que la relación entre ambas dure media hora.

Recuerda que el contexto del problema es quién define si una relación es temporal, debes ser objetivo y no añadir esta característica a cualquier elemento, solo porque tu lógica personal lo ve con buenos ojos.

Al igual que se realizó con las esperanzas de vida, los tiempo válidos para las relaciones se documentan con su granularidad y comentarios como se ve en la Tabla 4.

Nombre de la Relación	Granularidad del tiempo válido	Comentarios
Asiste	No temporal	Los estudiantes se demoran cursando la oferta de curso el tiempo que dura el curso.
Dicta	No temporal	La información temporal se puede obtener de la esperanza de vida de OfertaCurso.
Supervisa	No temporal	El cliente especificó que no desea saber datos sobre la supervisión.
Coordina	No temporal	No se requiere seguimiento. Aunque si es posible establecer el periodo en que un instructor coordinó a una facultad.
Pertenece	No temporal	Una facultad siempre tendrá profesores y estudiantes. El momento en que comienzan a hacer parte de esta se obtiene a través de la esperanza de vida de la entidad Persona.
EsPrerrequisito	No temporal	No se ve afectada por variaciones temporales.
Genera	No temporal	El periodo en que se abren las ofertas de curso ya está siendo monitoreado en con la esperanza de vida de estos.
Relaciones "EsUn"	No temporal	No se identificaron variaciones temporales
Tiene	No temporal	Cuando el horario de un curso o instructor se establece, es poco probable que se modifique.
Contiene	No temporal	Las facultades siempre contendrán cursos.

Tabla 4 *Anotaciones Temporales para las Relaciones*

Como se puede observar, ninguna de las relaciones que existe dentro del modelo de datos ha sido considerada temporal.

Esto se debe a que tenemos a que los requerimientos no abordan el posible cambio de características. Tampoco encontramos relaciones que sean efímeras y deban registrarse cada vez que ocurren.

Tarea 7.3. Identificar el Tiempo Válido de los Atributos

Ahora debemos decidir a qué atributos se les debe ser capturado el valor en un instante, ya que en el futuro puede que varíen.

La capacidad de cambio de un atributo debe estar contenida en la esperanza de vida de la entidad a la que haga parte, o al tiempo valido de una relación si es un atributo de relación.

Incluso se puede registrar el valor la llave primaria a través del tiempo, indicando que en cada instante la entidad es única, pero para este caso no aplicaremos este concepto debido a su complejidad.

Revisando el caso de estudio se concluye que el único atributo sometido a variaciones de tiempo y para el cual debe realizarse un seguimiento, es a salario de la entidad instructor.

Anotaciones Temporales para Atributos

Al instituto le interesa saber cómo ha sido la evolución del salario de cada instructor para estar al tanto de su desempeño.

Nombre de la entidad	Atributo	Granularidad del tiempo válido	Comentarios
Instructor	salario	INDETERMINADO	Se deben capturar los valores del salario a lo largo del tiempo para analizar las variaciones.

Tabla 5 *Anotaciones Temporales para los Atributos*

Tarea 7.4. Identificar la Temporalidad de las Transacciones

Hasta ahora hemos visto como asumir las variaciones temporales de los elementos de nuestro modelo de datos. Lo que nos permitirá mantener la lógica del negocio incluso considerando variaciones en el futuro.

El último paso es identificar el tiempo inmiscuido en las transacciones que sucederán sobre el modelo de datos. Lo cual se basa en saber qué cambios ocurren estructuralmente en nuestro esquema de datos.

Buenos ejemplos de estos sucesos son: Registrar el momento en que se creó una nueva instancia, El momento en que se modificó, El día en que se eliminó, los estados pasados de una instancia, etc.

Este paso consiste en identificar a que entidades, relaciones o atributos se les debe llevar un historial o registro de seguimiento (más conocido como *Tracking Log* o *Audit Trails*). Lo que es distinto de la esperanza de vida y el tiempo válido.

Anotaciones Temporales de Transacciones

Para llevar a cabo este diagnóstico se crea una nueva tabla que contenga las anotaciones sobre el objeto con los datos de la granularidad del tiempo de transacción y los comentarios pertinentes.

Por ejemplo podemos ver el caso de los atributos de la entidad estudiante en la Tabla 6.

Nombre de la entidad	Atributo	Granularidad temporal de la transacción	Comentarios
Estudiante	dirección	1 DIA	Realizar seguimiento
	teléfono	1 DIA	Realizar seguimiento

Tabla 6 *Anotaciones Temporales para las Transacciones de los Atributos*

Aunque en el caso de estudio del *Instituto San Judas* no se nos pide un historial, pero se supuso que se requiere una auditoría sobre los atributos direccion y telefono de la entidad Estudiante.

Esta declaración nos permitiría saber los valores que han tomado los atributos a lo largo de las transacciones realizadas en el día.

Anotaciones Temporales en el Diseño Lógico

Las anotaciones que se han realizado hasta el momento, serán de gran ayuda en el diseño lógico para realizar las transformaciones correspondientes que ayuden a seguir la temporalidad de los datos.

Solo en ese paso se hará uso de atributos y relaciones especiales que transformen los conceptos temporales en tablas que mantengan un record histórico del esquema de datos.

PASO 8 REVISAR EL MODELO DE DATOS CON EL USUARIO

El paso final es presentar el modelo de datos a los interesados directos en el desarrollo.

Debemos explicar el diagrama ER, el diccionario de datos y aquella documentación adicional que se haya creado para modelar la esencia del negocio.

El objetivo de este encuentro es que el usuario o encargado apruebe el modelo de datos y manifieste que se tiene claro el propósito del software y la forma en que se manejará el conglomerado de información.

Si el usuario rechaza algunas características del modelo, simplemente se acogen las nuevas recomendaciones y se adapta la solución hasta que se consiga la luz verde para pasar al diseño lógico.

CONCLUSIONES

- Hasta el momento se ha abordado por completo la fase de diseño conceptual de bases de datos, donde se apreció los beneficios que tiene comprender las necesidades de un cliente y el funcionamiento de lo que desea.
- Los pasos vistos para el diseño conceptual pueden abordarse para cualquier tipo de metodología a usarse. Ya sea con las metodologías predictivas o metodología ágiles.
- La actitud de un diseñador de base de datos hacia un problema debe enfocarse primero en el entendimiento de este y no en la solución. Modelar un sistema nublado desperdicia recursos.
- Frecuentemente el diseñador de bases de datos es a su vez un desarrollador, por lo que puede suceder que piense directamente en tablas sin tan solo siquiera entender que sucede en su entorno. Este enfoque debe modificarse para reducir los riesgos de implementación sin conceptos claros.
- El diseño conceptual es un proceso iterativo en constante cambio y evolución, por lo que se inspeccionar y adaptar de acuerdo a las necesidades del usuario.
- Un buen diseño conceptual es la base del éxito del diseño total y de la implementación.

ACERCA DEL AUTOR

James Revelo

James Revelo es blogger, desarrollador de software independiente y fundador de *Hermosa Programación*. Actualmente invierte la mayoría de su tiempo capacitándose en desarrollo de aplicaciones móviles y creación de videojuegos con el fin de proporcionar contenidos y recursos de alta calidad a la comunidad web.

ACERCA DE HERMOSA PROGRAMACIÓN

Hermosa Programación es un espacio dedicado a todos los desarrolladores de software que buscan artículos, guías, tutoriales y recursos sobre programación de forma autodidacta. Se caracteriza por el uso de una metodología detallada y dinámica, que aporte soluciones a los problemas cotidianos de los desarrolladores más exigentes.

Cada uno de los contenidos liberados es escrupulosamente planeado, diseñado, redactado e implementado para que los usuarios obtengan el mayor beneficio. Donde encontrarán contenidos sobre programación en diversos lenguajes como **C++**, **Java**, **Php**; **Diseño y Administración de bases de datos**, **Programación Gráfica** y **Desarrollo Móvil en la Plataforma Android**.

Puedes seguirnos en las siguientes redes sociales:

CREDITOS

Las ilustraciones de este ebook fueron obtenidas de los recursos gratuitos que tienen Freepik e IconFinder. Ambas son webs muy profesionales que alojan miles y miles de contenidos gráficos de excelente calidad.

Todos los créditos del diseño gráfico son para ellos.