UNIDAD 2. MODELO ENTIDAD/RELACIÓN

Objetivo

En esta unidad el lector aprenderá a:

- Definir el modelo Entidad/Relación y sus elementos.
- Modelar un caso práctico de automatización mediante el modelo Entidad/Relación.
- Modelar un caso práctico mediante el Lenguaje de Modelado Unificado (UML).

Introducción

El modelo de datos Entidad/Relación (E/R), está basado en una percepción del mundo real que consta de una colección de objetos básicos llamados entidades, y de relaciones entre esos objetos. Se desarrolló para facilitar el diseño de bases de datos permitiendo la especificación de un esquema de la empresa que representa la estructura lógica completa de una base de datos.

2.1. Conceptos básicos

Los elementos del modelo E/R son: entidades, atributos, relaciones y dominio, además del conjunto de valores análogo al concepto de domino. Es importante hacer buen uso de esta terminología y ser capaces de usarlos en el momento apropiado. Por ejemplo, en el modelo E/R, no se hacen referencias a tablas, aquí se llamarán *entidades*.

2.1.1. Entidad

Se puede definir como aquel objeto (real o abstracto) acerca del cual se quiere almacenar información en la base de datos. Existen dos clases de entidades: *regulares* que son aquellos que tienen existencia por sí mismas (como libro y autor), y *débiles*, cuya existencia depende de otro tipo de entidad (por ejemplo, familiar depende de empleado), y la desaparición de un empleado de base de datos hace que desaparezcan también todos los familiares que estaban a su cargo. Ejemplos de entidades son: empleado, cliente, organización, parte, ingrediente, orden de compra, orden de cliente, producto.

Un conjunto de entidades es la totalidad de las entidades del mismo tipo que comparten las mismas propiedades o atributos.

Entidades débiles y fuertes

Una entidad puede no tener suficientes atributos para formar una clave primaria. Tal entidad se denomina **entidad débil.** Una entidad que tiene una clave primaria se denomina **entidad fuerte.**

Una entidad fuerte es por definición una entidad dominante, mientras una entidad débil es una entidad subordinada.

Aunque un conjunto de entidades débil no tiene clave primaria, solamente se necesita conocer la distinción entre todas aquellas entidades del conjunto de entidades que dependen de una entidad particular fuerte. El **discriminante** de una entidad débil es un atributo o conjunto de atributos que permite que esta distinción se haga.

La clave primaria de una entidad débil se forma mediante la clave primaria de la entidad fuerte de cuya existencia depende la entidad débil, más el discriminante de la entidad débil.

2.1.2. Relación

Se entiende por relación aquella asociación o correspondencia existente entre entidades. Las relaciones típicamente son dadas por nombres. Un curso es *tomado* por un miembro de la facultad.

En un tipo de relación existen los siguientes elementos:

a) **Nombre:** Como todo objeto del modelo E/R, cada tipo de relación tiene un nombre que lo distingue unívocamente del resto y mediante el cual ha de ser referenciado.
b) **Grado o cardinalidad:** Es el número de tipos de entidad que participan en un tipo de relación.

Los siguientes son ejemplos de relaciones: Un empleado *toma* una orden de cliente, un estudiante se *inscribe* en un curso.

2.1.3. Atributo

Se refiere a cada una de las propiedades o características que tiene un tipo de entidad o un tipo de relación. Los atributos toman valores de uno o muchos dominios. Son las propiedades usadas para distinguir una instancia de una entidad de otra. Los atributos de la entidad empleado podrían incluir: *id_empleado*, *número de seguro social*, *nombre*, *apellido*, *dirección*, *ciudad*, *teléfono*.

- **Identificador:** Es un atributo especial usado para identificar una instancia específica de una entidad

- Normalmente se buscan identificadores *únicos*.
- El *número de seguro social* únicamente identifica a un empelado.
- *id_cliente* únicamente identifica a un cliente.
- También se pueden usar dos atributos para indicar un identificador: *numero_orden* y *num_elemento* únicamente identifica un elemento en una orden.

2.1.4. Dominio y valor

Las distintas propiedades o características de un tipo de entidad o de relación toman *valores* para cada ocurrencia de éstas. El conjunto de posibles valores que puede tomar una cierta característica se denomina **dominio**.

2.2. Diagramas Entidad/Relación (E/R)

La totalidad de estructuras lógicas de una base de datos se pueden expresar gráficamente mediante un **diagrama E/R**, que consta de los siguientes componentes:

- **Rectángulos:** Representan conjuntos de entidades.
- **Rectángulos dobles:** representan conjuntos de entidades débiles.
- **Elipses:** Representan atributos.
- **Rombos:** Representan relaciones entre conjuntos de entidades.
- **Líneas:** Unen los atributos con los conjuntos de entidades y los conjuntos de entidades con las relaciones.
- **Líneas dobles:** Indican participación total de una entidad en un conjunto de relaciones.

Considérense los diagramas Entidad/Relación de la Figura 2.1, que constan de dos conjuntos de entidades, *A* y *B*, relacionadas a través de un conjunto de relaciones binarias *X*.

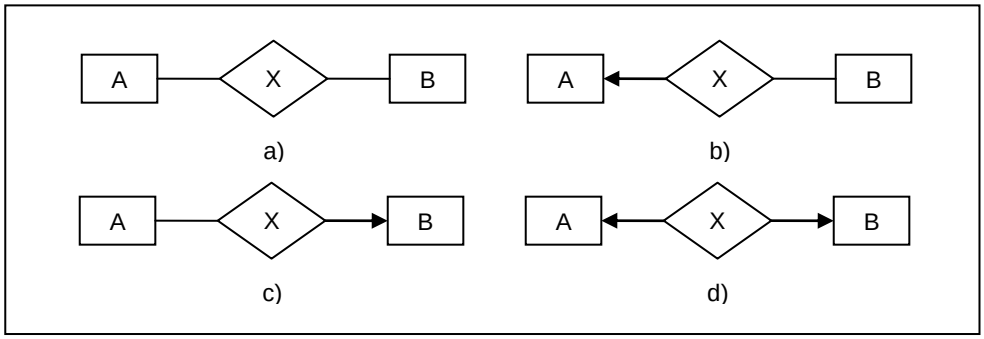

Figura 2.1. Diagrama E/R correspondiente a los tipos de relaciones.

Un conjunto de relaciones puede ser muchos a muchos, uno a muchos, muchos a uno, o uno a uno. Para distinguir entre estos tipos, se dibuja una línea dirigida (→) o una

línea no dirigida (—) entre el conjunto de relaciones y el conjunto de entidades en cuestión.

En la Figura 2.1 se muestra lo siguiente.

a) Relación uno a uno, debido a que las dos líneas que unen al conjunto de entidades *A* y *B* son no dirigidas.
b) Relación muchos a uno, debido a que la línea que une *A* y *X* es dirigida y la que une *X* y *B* es no dirigida.
c) Relación uno a muchos, debido a que la línea que une *A* y *X* es no dirigida y la que une *X* y *B* es dirigida.
d) Relación muchos a muchos, debido a que la línea que une *A* y *X* es dirigida al igual que la que une *X* y *B*.

Como se muestra en la Figura 2.2, los atributos de un conjunto de entidades que son miembros de la clave primaria están subrayados.

Analizando el diagrama E/R de la Figura 2.2, se ve que el conjunto de relaciones *renta* es muchos a muchos dado que las dos líneas que unen *cliente* y *película* son dirigidas.

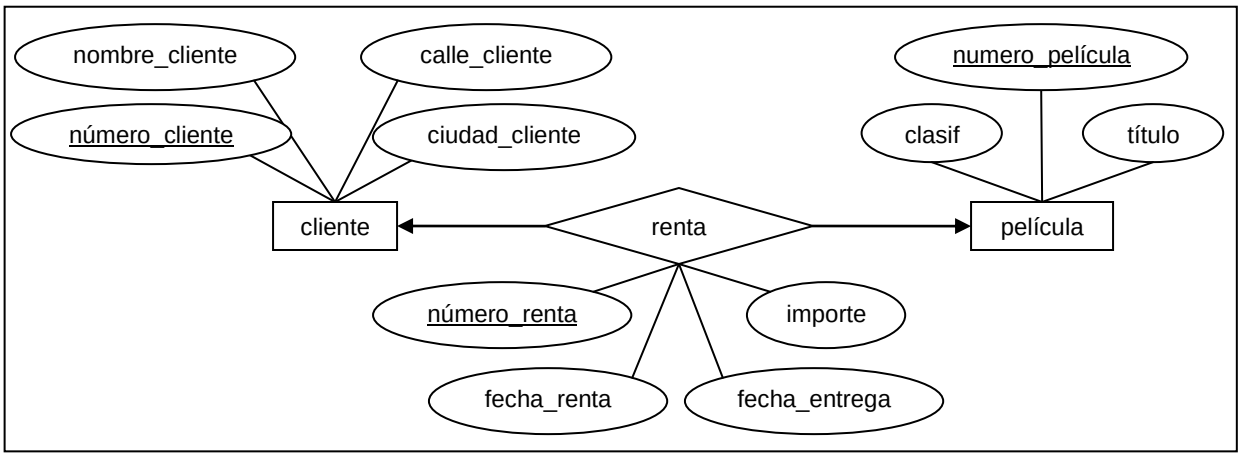

Figura 2.2. Diagrama E/R correspondiente a *cliente* y *película* en una relación muchos a muchos.

En la Figura 2.3 se muestran dos conjuntos de relaciones uno a muchos y muchos a uno. Desde *cliente* a *préstamo* es uno a muchos por lo que la línea desde *solicita* a *cliente* es no dirigida y la línea de *solicita* a *préstamo* es dirigida (Figura 2.3a). Análogamente, el conjunto de relaciones *banquero_consejero* es muchos a uno desde *cliente* a *empleado*, entonces la línea desde *banquero_consejero* a *cliente* tiene una flecha apuntando al conjunto de entidades *cliente* (Figura 2.3b).

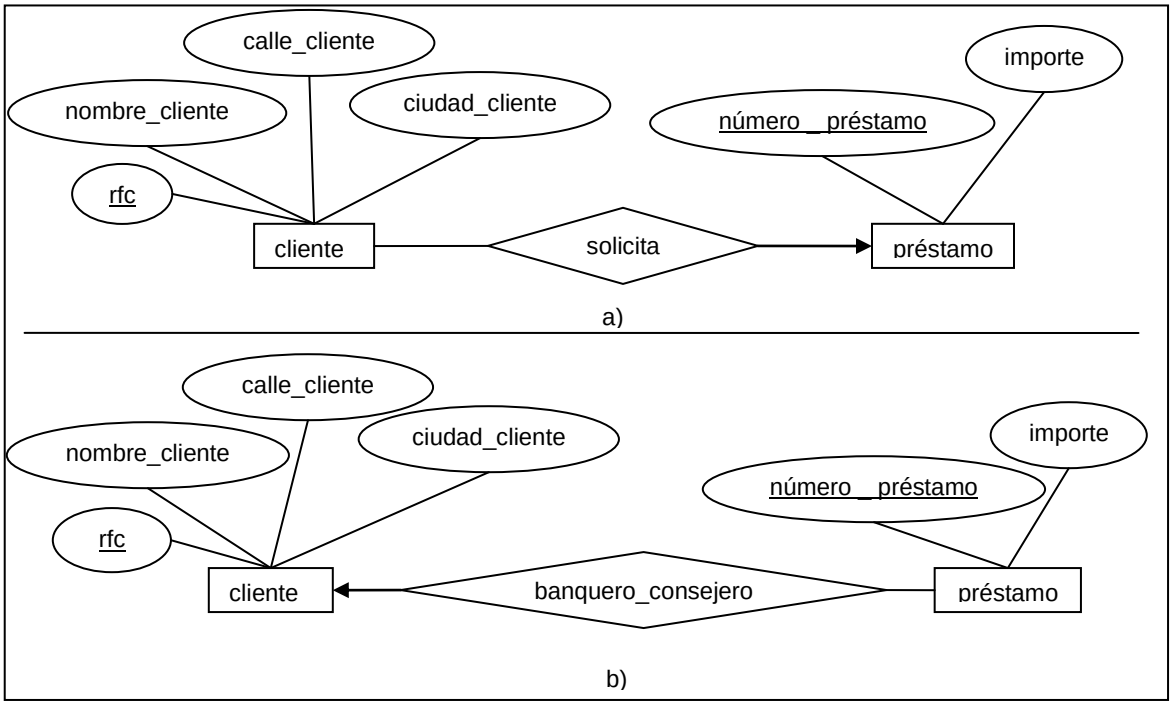

Figura 2.3. Relaciones. a) Uno a muchos; b) Muchos a uno.

Si un conjunto de relaciones tiene también algunos atributos asociados a él, entonces se unen esos atributos a ese conjunto de relaciones. Por ejemplo, en la Figura 2.4 se tiene el atributo descriptivo *fecha_acceso* unido al conjunto de relaciones *tiene* para especificar la fecha más reciente en que un cliente accedió a esa cuenta.

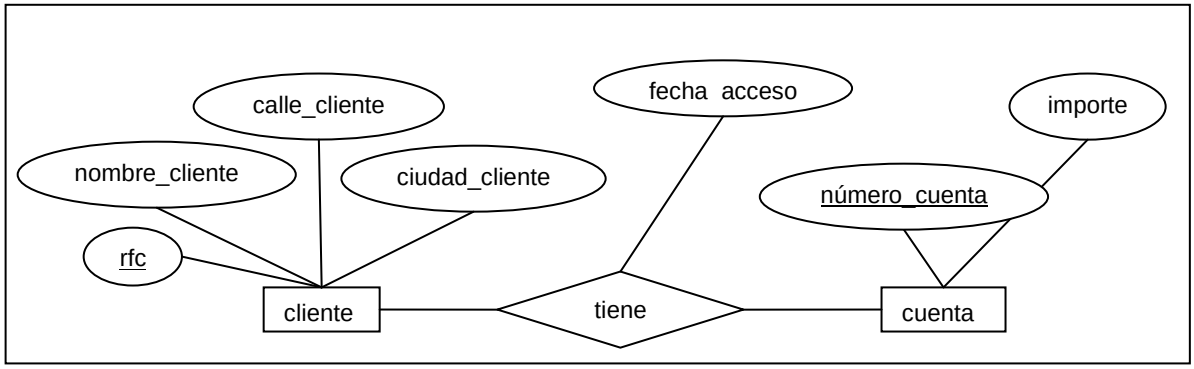

Figura 2.4. Diagrama E/R con un atributo unido a un conjunto de relaciones.

En los diagramas E/R se indican papeles mediante etiquetas en las líneas que unen rombos con rectángulos. En la Figura 2.5 se muestra el papel indicando *jefe* y *trabajador* entre el conjunto de entidades *empleado* y el conjunto de relaciones *trabaja_para*.

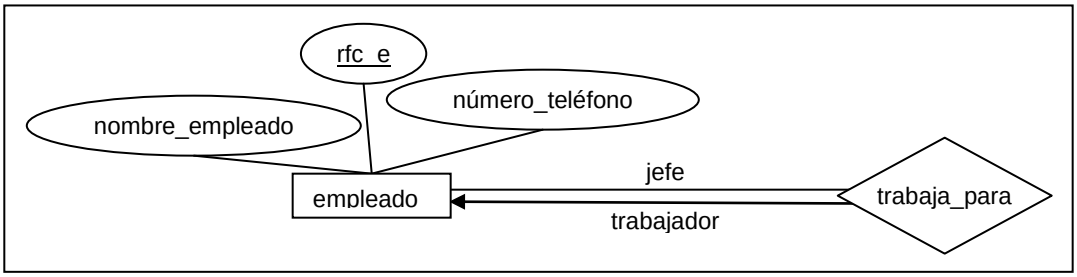

Figura 2.5. Diagrama E/R con indicadores de papeles.

En los diagramas E/R existen dos tipos de entidades, las fuertes y las débiles (como se mencionó en el apartado 2.1). En la Figura 2.6 se muestra la relación *cliente_factura*, en la cual la entidad débil *factura* es dependiente de la entidad fuerte *cliente*.

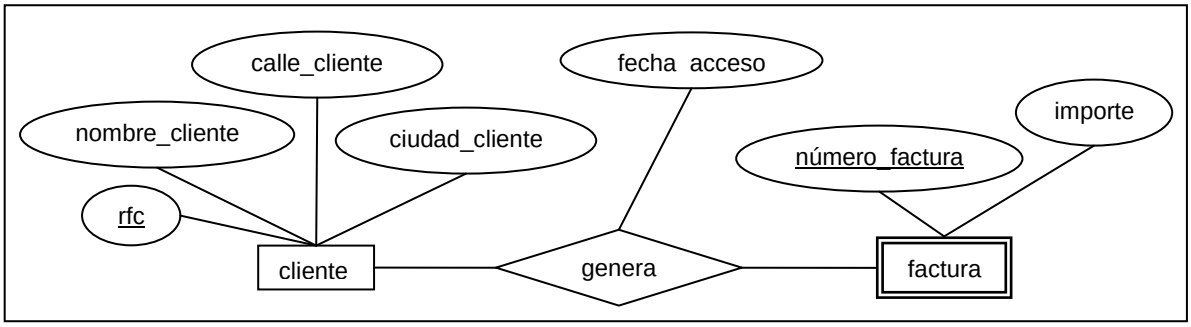

Figura 2.6. Diagrama E/R con una entidad débil.

2.3. Diseño de un esquema de base de datos

El modelo de datos E/R da una flexibilidad sustancial en el diseño de un esquema de bases de datos para modelar un desarrollo dado. En este apartado se considera cómo un diseñador de bases de datos puede seleccionar entre un amplio rango de alternativas. Entre las decisiones que se toman están las siguientes:

- Si se usa un atributo o un conjunto de entidades para representar un objeto.
- Si un concepto del mundo real se define más claramente mediante un conjunto de entidades o mediante un conjunto de relaciones.
- Si se usa un conjunto de entidades fuerte o débil; un conjunto de entidades fuerte y sus conjuntos de entidades débiles dependientes se pueden considerar como un objeto único en la base de datos, debido a que la existencia de las entidades débiles depende de la entidad fuerte.

Se verá que el diseñador de bases de datos necesita un buen entendimiento del desarrollo que se modela para tomar estas decisiones.

2.3.1. Fases de diseño

Un modelo de datos de alto nivel sirve al diseñador de la base de datos para proporcionar un marco conceptual en el que se especifican de forma sistemática los requisitos de datos de los usuarios de la base de datos que existen, y cómo se estructurará la base de datos para completar estos requisitos. La fase inicial del diseño de bases de datos, entonces, es caracterizar completamente las necesidades de datos esperadas por los usuarios de la base de datos. El resultado de esta fase es una *especificación de requisitos del usuario.*

A continuación, el diseñador elige un *modelo de datos*, y aplicando los conceptos de elección del modelo de datos, traduce estos requisitos a un esquema conceptual de la base de datos. El esquema desarrollado en esta fase de diseño conceptual proporciona una visión detallada del desarrollo. Debido a que se ha estudiado el modelo E/R hasta ahora, es usará éste para desarrollar el esquema conceptual. En términos del modelo E/R, el esquema especifica todos los conjuntos de entidades, conjuntos de relaciones, atributos y ligaduras de correspondencia. Las revisiones del diseñador del esquema confirman que todos los requisitos de datos se han satisfecho y no hay conflictos con otros. También se examina el diseño para eliminar características redundantes. Lo importante en este punto es describir los datos y las relaciones, más que especificar detalles del almacenamiento físico.

Un desarrollo del esquema conceptual completo indicará también los requisitos funcionales del desarrollo. En una especificación de requisitos funcionales los usuarios describen los tipos de operaciones (o transacciones) que tendrán lugar en los datos. Algunos ejemplos de operaciones son la modificación o actualización de datos, la búsqueda y recuperación de datos específicos y el borrado de datos. En esta fase de diseño conceptual se puede hacer una revisión del esquema para encontrar los requisitos funcionales.

El proceso de trasladar un modelo abstracto de datos a la implementación de la base de datos consta de dos fases de diseño finales: la **fase del diseño lógico** y la **fase del diseño físico**. En la fase de diseño lógico, el esquema conceptual de alto nivel está asociado al modelo de datos de implementación de los SGBD que se usarán. El esquema de bases de datos del SGBD específico resultante se usa en la siguiente fase de diseño físico, en la que se especifican las características físicas de la base de datos. Estas características incluyen la forma de organización de los archivos y las estructuras de almacenamiento interno.

En este tema se tratarán solo los conceptos de modelo E/R usados en la fase de diseño del esquema conceptual. Se ha presentado una breve visión del proceso de diseño de base de datos para proporcionar un contexto para la discusión del modelo de datos E/R.

2.3.2. Requisitos de datos para el desarrollo del banco

La especificación inicial de los requisitos de usuario se puede basar en entrevistas con los usuarios de la base de datos y en el análisis del propio diseño del desarrollo. La

descripción que surge de esta fase de diseño sirve como base para especificar la estructura conceptual de la base de datos. La siguiente lista describe los principales requisitos del desarrollo del banco.

- El banco está organizado en sucursales. Cada sucursal está ubicada en una ciudad particular y se identifica por un nombre único. El banco controla los activos de cada sucursal.
- Los clientes del banco se identifican mediante su número de empleado. El banco almacena cada nombre de cliente, la calle y ciudad donde viven los clientes. Los clientes pueden tener cuentas y pueden pedir préstamos. Un cliente puede estar asociado con un banquero particular, que puede actuar como responsable de préstamos o banquero personal para un cliente.
- Los empleados del banco se identifican mediante su número de empleado. La administración del banco almacena el nombre y número de teléfono de cada empleado, los nombres de los subordinados del empleado y el número de empleado del jefe del empleado. El banco también mantiene el registro de la fecha de comienzo del empleado así como del tiempo del empleado.
- Un préstamo tiene lugar en una sucursal particular y puede estar asociado a uno o más clientes. Un préstamo se identifica mediante un único número de préstamo. Para cada préstamo el banco mantiene el registro del importe del préstamo y de los pagos del préstamo. Aunque un número de pago del préstamo no identifica de manera única un pago entre todos los préstamos del banco, un número de pago identifica un pago particular para un préstamo específico. Para cada pago se almacena la fecha y el importe.

En un desarrollo de un banco real el banco mantendría información de los abonos y cargos en las cuentas, igual que se mantiene el registro de los pagos para los préstamos. Debido a que los registros del modelo para este seguimiento son similares en este modelo no se mantiene un seguimiento de tales abonos y cargos.

2.3.3. Designación de los conjuntos de entidades en el desarrollo bancario

La especificación de los requisitos de datos sirve como un punto de partida para la construcción de un esquema conceptual para la misma, a partir del apartado 2.3.2 se ha comenzado a identificar los conjuntos de entidades y sus atributos.

- El conjunto de entidades *sucursal*, con los atributos *nombre_sucursal*, *ciudad_sucursal* y *activos.*
- El conjunto de entidades *clientes*, con los atributos *nombre_cliente*, *numero_empleado*, *calle_cliente* y *ciudad_cliente.* Un posible atributo adicional es *nombre_banquero.*
- El conjunto de entidades *empleado*, con los atributos *numero_empleado_e*, *nombre_empleado*, *numero_telefono*, *sueldo* y *jefe*. Algunas características

descriptivas adicionales son el atributo multivalorado *nombre_subordinado*, el atributo base *fecha_comienzo*, y el atributo derivado *antigüedad*.

- El conjunto de entidades *prestamo,* con los atributos *numero_prestamo*, *importe* y *sucursal_origen*. Un posible atributo adicional es el atributo compuesto multivalorado *prestamo_pago*, con los atributos componentes *numero_pago*, *fecha_pago* e *importe_pago*.

2.3.4. Designación de los conjuntos de relaciones en el desarrollo del banco

Volviendo ahora al esquema simple de diseño definido en el apartado 2.3.3, se especifican los siguientes conjuntos de relaciones y correspondencia de cardinalidades.

- *solicita*, un conjunto de relaciones muchos a muchos entre *cliente* y *prestamo.*
- *prestamo_sucursal,* un conjunto de relaciones uno a muchos desde *prestamo* a *pago*, que documenta que un pago se hace en un préstamo.
- *tiene,* con el atributo de relación *fecha_acceso*, un conjunto de relaciones muchos a muchos entre *cliente* y *cuenta*, indicando que un cliente posee una cuenta.
- *banquero_consejero,* un conjunto de relaciones muchos a uno que expresa que un cliente puede ser aconsejado por un empleado del banco, y que un empleado del banco puede aconsejar a uno o más clientes.
- *trabaja_para*, un conjunto de relaciones entre entidades de *empleado* con papeles que indican *jefe* y *trabajador*; la correspondencia de cardinalidades expresa que un empleado trabaja para un único jefe, y que un jefe supervisa a uno o más empleados.

Nótese que se ha sustituido el atributo *nombre_banquero* del conjunto de entidades *cliente* por el conjunto de relaciones *banquero_consejero*, y el atributo *jefe* del conjunto de entidades *empleado* por el conjunto de relaciones *trabaja_para*. Se ha elegido mantener *prestamo* como un conjunto de entidades. Los conjuntos de relaciones *prestamo_sucursal* y *prestamo_pago* han sustituido, respectivamente, los atributos *sucursal_origen* y *prestamo_pago* del conjunto de entidades *prestamo*.

2.3.5. Diagrama E/R para el desarrollo bancario

El dibujo en cuestión discutido en el apartado 2.3.4 se presenta ahora en el diagrama E/R completo para el ejemplo del desarrollo bancario.

En la Figura 2.7 se muestra la representación completa de un modelo conceptual de un banco, expresada en términos de los conceptos E/R. El diagrama incluye los conjuntos de entidades, atributos, conjuntos de relaciones y correspondencia de cardinalidades alcanzados a través del proceso de diseño de los apartados 2.3.2 y 2.3.3, y refinados en el apartado 2.3.4. Para identificar las claves primarias éstas se muestran subrayadas.

En la Figura 2.8 se muestra el mismo ejemplo para el desarrollo bancario utilizando otro esquema de bases de datos, el modelo relacional extendido.

En este esquema cabe resaltar algunos puntos:

- Las relaciones desaparecen a menos que sean muchos a muchos y/o tengan atributos, en este caso se creará como una entidad unida a las originales y teniendo las llaves primarias de las entidades como llaves primarias y foráneas, además de sus atributos si tuviese.
- En una relación muchos a uno o uno a muchos la llave primaria de la entidad que tiene la línea no dirigida se coloca en la entidad con línea dirigida como llave foránea. En caso de ser una relación uno a uno el desarrollador selecciona en que entidad colocar la llave foránea de acuerdo a sus necesidades. En caso de que se haya creado una entidad (no se realiza este punto, ver punto anterior).
- Para fines de documentación se acostumbra escribir después de la llave foránea la leyenda *(FK entidad)*. *FK* indica que es una llave foránea y *entidad* la entidad de la que proviene.

2.4. Lenguaje de Modelado Unificado UML (Modelo Conceptual)

2.4.1. Modelado visual

Tal como indica su nombre, UML es un lenguaje de modelado. Un modelo es una simplificación de la realidad. El objetivo del modelado de un sistema es capturar las partes esenciales del sistema. Para facilitar este modelado, se realiza una abstracción y se plasma en una notación gráfica. Esto se conoce como **modelado visual**.

El modelado visual permite manejar la complejidad de los sistemas a analizar o diseñar. UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten. Otro objetivo de este modelado visual es que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos). UML es además un método formal de modelado, esto aporta las siguientes ventajas:

- Mayor rigor en la especificación.
- Permite realizar una verificación y validación del modelo realizado.
- Se puede automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). Esto permite que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de más alto nivel, de la estructura de un proyecto.

2.4.2. UML

UML es ante todo un lenguaje. Un lenguaje proporciona un vocabulario y reglas para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema. Este lenguaje nos indica cómo crear y leer los modelos, pero no dice cómo crearlos. Esto último es el objetivo de las metodologías de desarrollo. Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- **Visualizar:** UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- **Especificar:** UML permite especificar cuáles son las características de un sistema antes de su construcción.
- **Construir:** a partir de los modelos especificados se puede construir los sistemas diseñados.
- **Documentar:** Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Aunque UML está pensado para modelar sistemas complejos con gran cantidad de software, el lenguaje es lo suficientemente expresivo como para modelar sistemas que no son informáticos, como flujos de trabajo (workflow) en una empresa, diseño de la estructura de una organización y por supuesto, en el diseño de hardware.

Un modelo UML está compuesto por tres clases de bloques de construcción:

- **Elementos:** Los elementos son abstracciones de cosas reales o ficticias (objetos, acciones, etc.)
  Elementos estructurales.
  - Actores: Un actor es "algo" o "alguien" que puede interaccionar con el sistema que se está desarrollando (véase Figura 2.9a).
  - Caso de uso: Es una descripción de un conjunto de secuencias de acciones que un sistema ejecuta y que produce un resultado observable de interés para un actor particular (véase Figura 2.9b).
  - Clases: Es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica (véase Figura 2.9c).
  - Objeto: Es una instancia de alguna clase (véase Figura 2.9d).

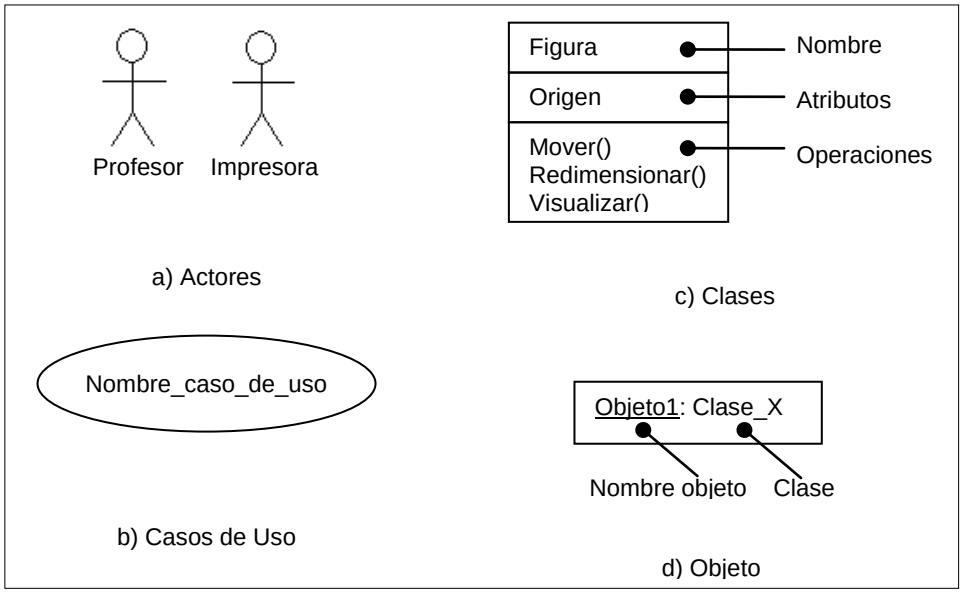

Figura 2.9. Elementos estructurales.

Elementos de comportamiento.

- Mensajes: Se usan para especificar una comunicación entre objetos (véase Figura 2.10.). Se utilizan en los diagramas de secuencia que se analizarán más adelante.

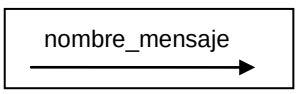

Figura 2.10. Mensajes.

Elementos de agrupación.

– Paquete: Sirve para organizar elementos en grupos. Un paquete es puramente conceptual y sólo existe en tiempo de desarrollo (véase Figura 2.11).

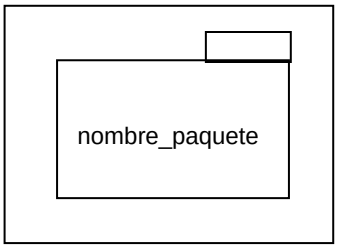

Figura 2.11. Paquete.

- **Relaciones:** Relacionan los elementos entre sí.
  – Dependencia: Es una relación semántica entre dos elementos (o dos conjuntos de elementos), en la cual un cambio en un elemento puede afectar a la semántica de otro elemento. Existen varios tipos de dependencia predefinidas que se indican mediante estereotipos, por ejemplo: «extend», e «include» para casos de uso (véase Figura 2.12).

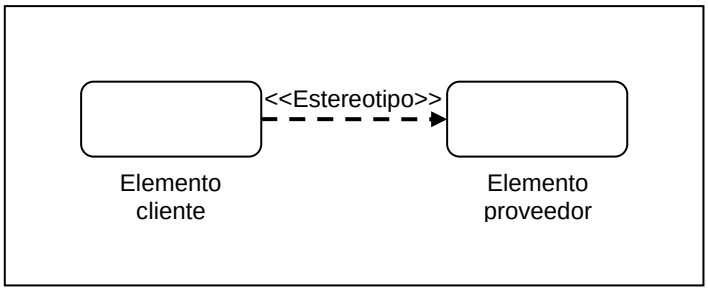

Figura 2.12. Relaciones de dependencia.

  – Asociación: Es una relación estructural entre dos elementos, que describe las conexiones entre ellos (suele ser bidireccional). Es la única relación permitida entre los actores y los casos de uso ya que refleja la comunicación existente entre un actor y un caso de uso (véase Figura 2.13)

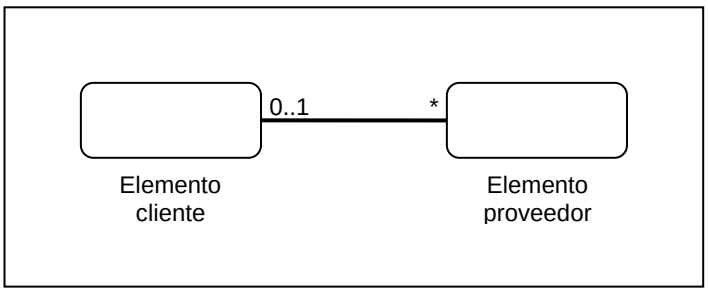

Figura 2.13. Relaciones de Asociación.

  – Agregación: Es una relación estructural entre un todo y sus partes. Se denota por una línea terminada en un "diamante" en el extremo de la clase que representa el todo (véase Figura 2.14).

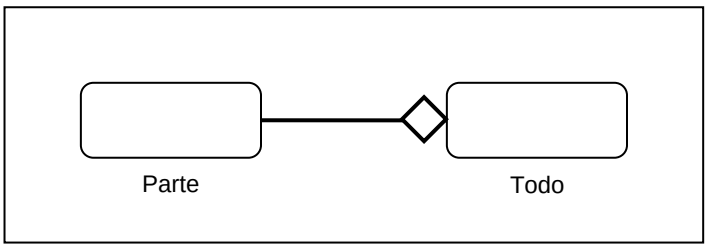

Figura 2.14. Relaciones de Agregación.

- Generalización: Es una relación taxonómica entre un elemento más general (el padre) y un elemento más específico (el hijo). Se usa tanto en diagramas de clases como en diagramas de casos de uso (véase Figura 2.15).

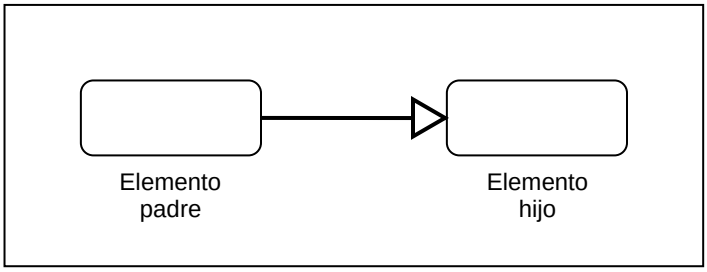

Figura 2.15. Relaciones de Generalización.

- **Diagramas:** Son colecciones de elementos con sus relaciones. Este tema se verá con más detalle en el tema 2.4.3.

2.4.3. Diagramas UML

Un diagrama es la representación gráfica de un conjunto de elementos con sus relaciones. En concreto, un diagrama ofrece una vista del sistema a modelar. Para poder representar correctamente un sistema, UML ofrece una amplia variedad de diagramas para visualizar el sistema desde varias perspectivas. UML incluye los siguientes diagramas:

- Diagrama de casos de uso.
- Diagrama de clases.
- Diagrama de objetos.
- Diagrama de secuencia.
- Diagrama de colaboración.
- Diagrama de estados.
- Diagrama de actividades.
- Diagrama de componentes.
- Diagrama de despliegue.

Los diagramas más interesantes (y los más usados) son los de casos de uso, clases y secuencia, por lo que los temas siguientes se centrarán en éstos. Pare ello, se utilizarán ejemplos de un sistema de venta de entradas de cine por Internet.

- **Diagrama de casos de usos:** Representa gráficamente los casos de uso que tiene un sistema. Se define un caso de uso como cada interacción supuesta con el sistema a desarrollar, donde se representan los requisitos funcionales. Es decir, se está diciendo lo que tiene que hacer un sistema y cómo. En la Figura 2.16 se muestra un ejemplo de casos de uso, donde se muestran tres actores (los clientes, los taquilleros y los jefes de taquilla) y las operaciones que pueden realizar (sus roles).

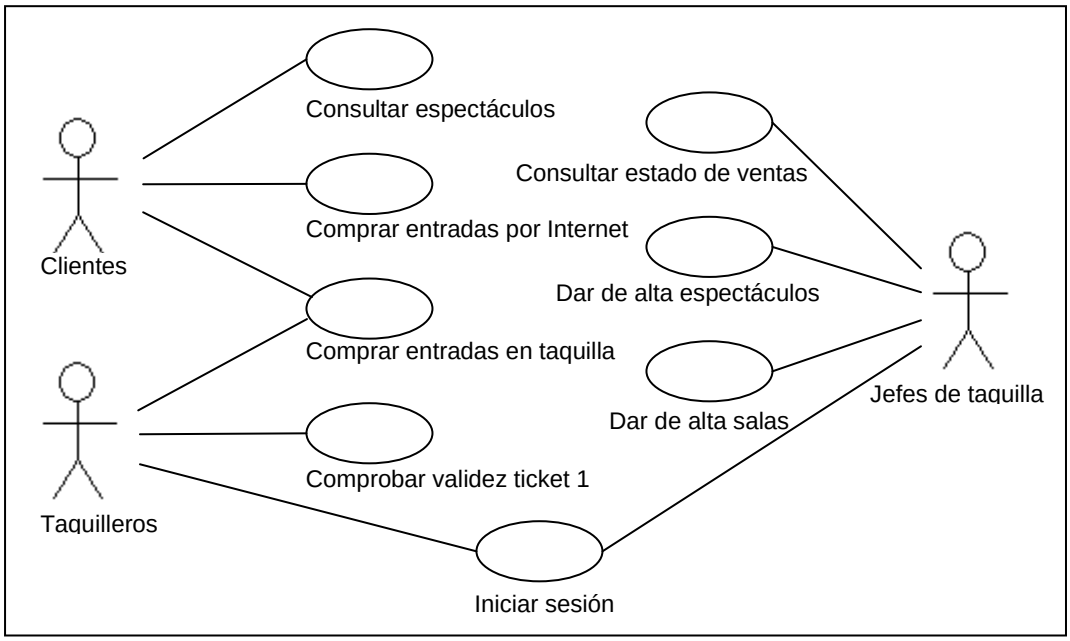

Figura 2.16. Diagrama de Casos de Uso.

- **Diagrama de clases:** Muestra un conjunto de clases, interfaces y sus relaciones. Éste es el diagrama más común a la hora de describir el diseño de los sistemas orientados a objetos. En la Figura 2.17 se muestran las clases globales, sus atributos y las relaciones de una posible solución al problema de la venta de entradas.

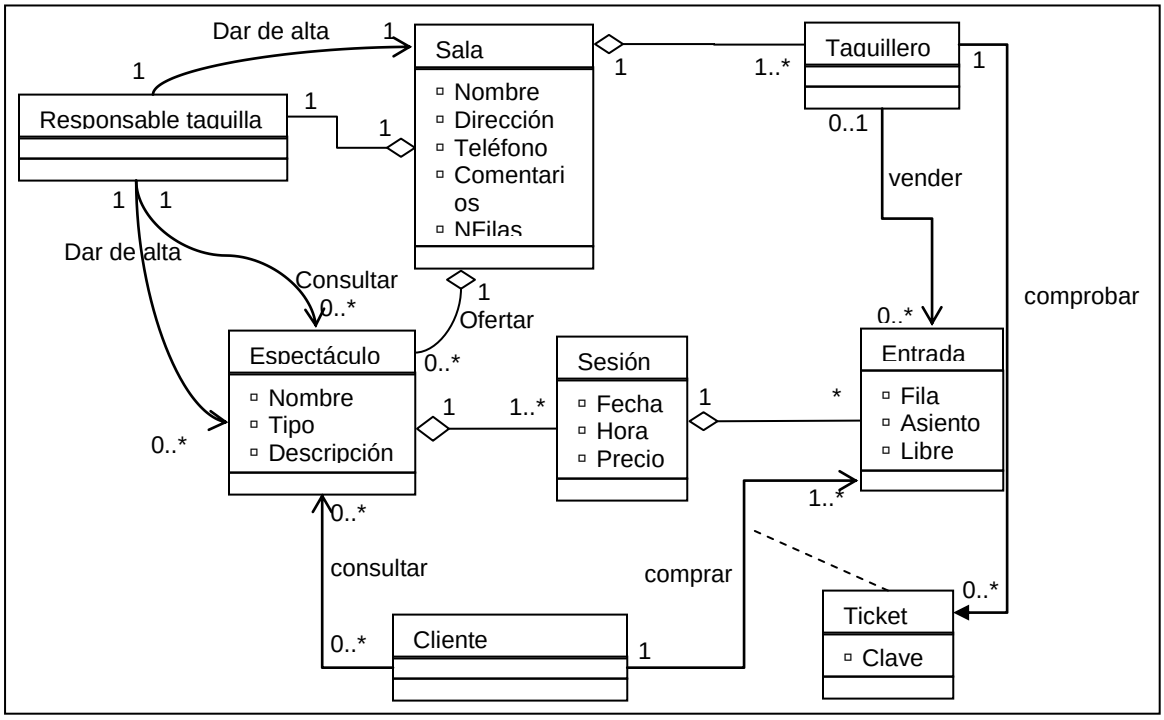

Figura 2.17. Diagrama de clases.

- **Diagrama de secuencia:** Se muestra la interacción de los objetos que componen un sistema de forma temporal. Siguiendo el ejemplo de venta de entradas, la Figura 2.18 muestra la interacción de crear una nueva sala para un espectáculo.

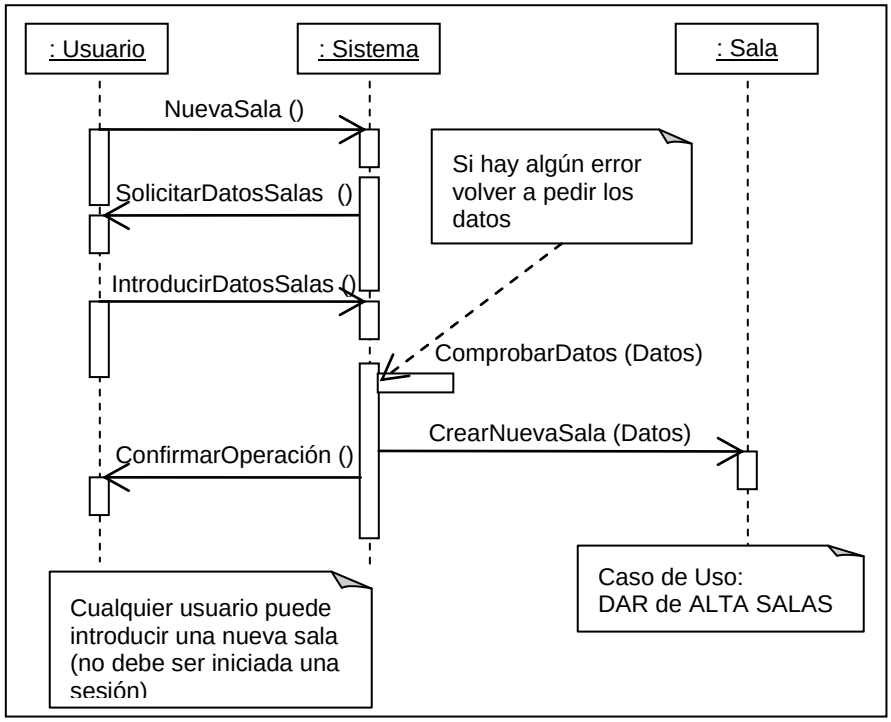

Figura 2.18. Diagrama de Secuencia.

El resto de los diagramas muestran distintos aspectos del sistema a modelar. Para modelar el comportamiento dinámico del sistema están los de interacción, colaboración, estados y actividades. Los diagramas de componentes y despliegue están enfocados a la implementación del sistema.

2.4.4. Reglas.

Los bloques de construcción se deben combinar siguiendo las normas que establece UML. UML establece una serie de normas sobre cómo nombrar a los elementos, relaciones y diagramas; la visibilidad y alcance de dichos nombres y sobre su integridad (cómo diseñar relaciones consistentes). Podremos decir que un modelo está bien formado cuando cumpla estas reglas.

2.4.5. Mecanismos comunes.

Bajo una serie de mecanismos que se aplican durante todo el proceso de desarrollo de modelos en UML, se consiguen diseños simples y eficientes. Estos mecanismos son los siguientes:

- **Especificaciones:** UML no es simplemente un lenguaje que proporciona elementos gráficos para modelado, tras cada elemento existe una especificación que nos va a permitir detallar textualmente el comportamiento de los mismos; de esta forma con los elementos gráficos conseguimos una visualización global del sistema y con la especificación conseguimos una visualización detallada del mismo.
- **Adornos:** Todos los elementos gráficos en UML poseen una notación gráfica que representa los aspectos más importantes de éstos; aún así existe una notación que permite representar los aspectos más específicos a través de adornos, consiguiendo así una especificación aún más detallada.
- **Divisiones comunes:** UML permite representar las distintas divisiones que presenta el diseño orientado a objetos. Por ejemplo la división clase/objeto donde un objeto es la manifestación concreta de la clase. UML permite el modelado conjunto de los mismos a través de una notación específica.
- **Mecanismos de extensibilidad:** UML proporciona un lenguaje abierto en el que se pueden expresar los matices de los distintos sistemas, para ello hace uso de tres mecanismos que permiten extender el lenguaje:
  - Estereotipo: Permite crear nuevos bloques de construcción partiendo de los ya existentes.
  - Valor etiquetado: Permite añadir nuevas propiedades a un bloque de construcción.
  - Restricción: Permite añadir o modificar las reglas referidas a los bloques de construcción.

2.4.6. Proceso de desarrollo.

Aunque UML es bastante independiente del proceso de desarrollo que se siga, los mismos creadores de UML han propuesto su propia metodología de desarrollo, denominada el *Proceso Unificado de Desarrollo*. El Proceso Unificado de Desarrollo está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidos. Además, el proceso unificado utiliza el UML para expresar gráficamente todos los esquemas de un sistema software. Pero, realmente, los aspectos que definen este proceso unificado son tres: es iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura:

- **Dirigido por casos de uso:** Basándose en los casos de uso, los desarrolladores crean una serie de modelos de diseño e implementación que los llevan a cabo. Además, estos modelos se validan para que sean conformes a los casos de uso. Finalmente, los casos de uso también sirven para realizar las pruebas sobre los componentes desarrollados.
- **Centrado en la arquitectura:** En la arquitectura de la construcción, antes de construir un edificio éste se contempla desde varios puntos de vista: estructura, conducciones eléctricas, fontanería, etc. Cada uno de estos aspectos está representado por un gráfico con su notación correspondiente. Siguiendo este ejemplo, el concepto de arquitectura software incluye los aspectos estáticos y dinámicos más significativos del sistema.
- **Iterativo e incremental:** Todo sistema informático complejo supone un gran esfuerzo que puede durar desde varios meses hasta años. Por lo tanto, lo más práctico es dividir un proyecto en varias fases. Actualmente se suele hablar de ciclos de vida en los que se realizan varios recorridos por todas las fases. Cada recorrido por las fases se denomina iteración en el proyecto en la que se realizan varios tipos de trabajo (denominados flujos). Además, cada iteración parte de la anterior incrementado o revisando la funcionalidad implementada. Se suele denominar proceso (véase Figura 2.19).

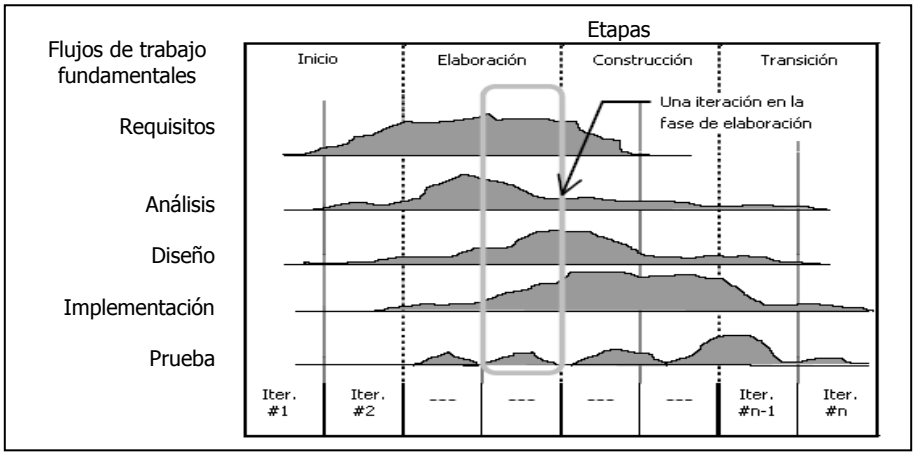

Figura 2.19. Proceso iterativo e incremental.

2.5. Resumen

El diseño de bases de datos se descompone en tres etapas: diseño conceptual, diseño lógico y diseño físico. El diseño conceptual es el proceso por el cual se construye un modelo de la información que se utiliza en una empresa u organización independientemente del SGBD que se vaya a utilizar para implementar el sistema y de los equipos informáticos o cualquier otra consideración física.

Un modelo conceptual es un conjunto de conceptos que permiten describir la realidad mediante representaciones lingüísticas y gráficas.

El modelo conceptual más utilizado es el modelo Entidad/Relación, que posee los siguientes conceptos: entidades, relaciones, atributos, dominios de atributos e identificadores.

Cada entidad comprende los datos que un usuario maneja para llevar a cabo una determinada tarea. Normalmente estas entidades corresponden a las distintas áreas funcionales de la empresa y se pueden identificar examinando los diagramas de flujo de datos o entrevistando a los usuarios, examinando los procedimientos, informes y formularios, y observando el funcionamiento de la empresa.

El modelo de datos E/R permite al diseñador de bases de datos seleccionar entre el amplio rango de alternativas para definir que tipo de entidad, relación y/o atributo utilizar. Sin embargo se debe tomar en cuenta que el diseñador debe tener un excelente entendimiento del problema que se quiere resolver para así definir las características principales del modelo.

Otro modelo conceptual que se explicó en este capítulo es el UML, el cual es un lenguaje de modelado, lo cual quiere decir que nos ayuda a obtener una simplificación de la realidad.

UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten. Otro objetivo de este modelado es que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML. Aunque el modelo E/R también tiene esta característica la diferencia es que UML está pensado principalmente para lenguajes orientados a objetos. El lenguaje UML nos permite visualizar, especificar, construir y documentar.

Otra característica de UML es que aunque está pensado para modelar sistemas complejos con gran cantidad de software, el lenguaje es lo suficientemente expresivo como para modelar sistemas que no son informáticos.

Un modelo UML esta compuesto por tres clases de bloques de construcción: elementos, relaciones y diagramas, cada uno de los cuales tienen tipos y sub-tipos.

Aunque UML es bastante independiente del proceso de desarrollo que se siga, los mismos creadores de UML han propuesto su propia metodología de desarrollo, denominada el *Proceso Unificado de Desarrollo*.

2.6. Glosario

Atributo	Cada una de las propiedades o características que tiene un tipo de entidad o un tipo de relación.
Cardinalidad o grado	Especifica el número mínimo y máximo de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad. La participación de una entidad en una relación es obligatoria si la existencia de cada una de sus ocurrencias requiere la existencia de al menos una ocurrencia de la otra entidad participante.
Diagrama Entidad/Relación (E/R)	La totalidad de estructuras lógicas de una base de datos. Que incluyen entidades, relaciones, atributos, etc.
Dominio y valor	Las distintas propiedades o características de un tipo de entidad que toman valores para cada ocurrencia de éstas.
Entidad	Objeto acerca del cual queremos almacenar información en la base de datos. Existen dos tipos las fuertes o regulares, las cuales tiene una clave primaria; y las débiles que no tienen suficientes atributos para formar dicha clave.
Lenguaje de Modelado Unificado UML	Es un lenguaje que proporciona un vocabulario y reglas para permitir la representación gráfica de un sistema. Nos indica cómo crear y leer los modelos. Consta de elementos, relaciones y diagramas.
Relación	Es una asociación o correspondencia existente entre entidades. Y tienen un nombre y un grado o cardinalidad.

2.7. Ejercicios

1. Proporcione ejemplos de:
   - Una relación muchos a muchos en el que uno de los participantes sea una entidad débil.
   - Una relación muchos a uno en el que uno de los participantes sea otra relación.
   - Una relación muchos a muchos en el que uno de los participantes sea una entidad débil.
   - Una relación muchos a uno en el que uno de los participantes sea otra relación.

2. Diseñe un diagrama E/R para la base de datos personal de una compañía con las siguientes consideraciones:

   - La compañía tiene un conjunto de departamentos.
   - Cada departamento tiene un conjunto de empleados, un conjunto de proyectos y un conjunto de oficinas.
   - Cada empleado tiene una historia laboral (conjunto de puestos que ha ocupado). Para cada uno de estos puestos el empleado tuene una historia salarial.
   - Cada oficina tiene un conjunto de teléfonos.

   La base de datos deberá almacenar la siguiente información:

   - Para cada departamento: número de departamento (único), presupuesto y número de empleado (único) del gerente del departamento.
   - Para cada empleado: número de empleado (único), número de proyecto actual, número de oficina y número telefónico; además el nombre de cada puesto que ha ocupado el empleado más la fecha y salario para cada salario distinto recibido en ese puesto.
   - Para cada proyecto: número de proyecto (único) y presupuesto.
   - Para cada oficina: número de oficina (único), área del piso y número telefónicos (únicos) de todos los teléfonos de esa oficina.

   Declare cualquier suposición que haga sobre los atributos y claves primarias usadas.

3. Una base de datos contendrá información relativa a representantes de ventas, áreas de ventas y productos. Cada representante es responsable de las ventas en una o más áreas; cada área tiene uno o más representantes responsables. En forma similar cada representante es responsable de la venta de uno o más productos, y cada producto tiene uno o más representantes responsables. Todos los productos se venden en todas las áreas; sin embargo, dos representantes no pueden vender el mismo producto en la misma área. Todos los representantes venden el mismo conjunto de productos en todas las áreas de las que son responsables. Diseñe un diagrama E/R para este ejercicio.

4. En un sistema de registro de pedidos se emplea una base de datos conteniendo información sobre clientes, artículos y pedidos. Diseñe un diagrama E/R que incluya la siguiente información:

   - Para cada cliente: número de cliente (único), direcciones “enviar a” (varias por cliente), estado de cuenta, límite de crédito y descuento.
   - Para cada pedido: información del encabezado (número de cliente, dirección de envío y fecha de pedido) y líneas de detalle (número de artículo, cantidad ordenada) las cuales son varias por pedido.

- Para cada artículo: plantas manufactureras, existencia de cada planta y descripción del artículo.

Como en el ejercicio 2 declare cualquier suposición que haga sobre los atributos y claves primarias usadas.

2.8. Cuestionario

1. Defina el modelo Entidad/Relación
2. Defina cada uno de los elementos del modelo Entidad/Relación
3. ¿Cuál es la diferencia entre entidad fuerte y débil?
4. ¿A qué se le llama clave primaria?
5. ¿Cuál es la primera fase del diseño de bases de datos y a que se refiere?
6. Defina UML
7. Defina los objetivos de UML
8. ¿A qué se refiere el modelado visual?
9. Mencione y explique las tres clases de bloques de construcción.

2.9. Bibliografía

[1] Silberschatz Abraham, Korth Henry F., Sudarshan (1988). **Fundamentos de bases de datos.** 3ª. Ed. Mc Graw Hill. España.

[2] Date, C.J. (2000). **Sistemas de bases de datos.** 7ª. Ed. Prentice Hall. México.

[3] Marqués, Mercedes (2002). **Diseño de bases de datos.** http://www3.uji.es/~mmarques/f47/apun. Universitat Jaume I.
Fecha de consulta: 12 de noviembre del 2004.

[4] Leyva, Juan. (2004). **UML.**
http://juanleyva.metricasweb.com/tutoriales/uml.htm.
Fecha de consulta: 7 de diciembre del 2004.

[5] Hernández Orallo, Enrique (2002). **ActaUML.pdf**.
http://dark.disca.upv.es/enheror/pdf/ActaUML.pdf. Universidad Politécnica de Valencia, España.
Fecha de consulta: 7 de diciembre del 2004.

[6] Mora, Francisco (2002). **UML.pdf.**
http://www.dccia.ua.es/dccia/inf/asignaturas/GPS/archivos/uml.pdf. Universidad de Alicante.
Fecha de consulta: 7 de diciembre del 2004.