

**BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE
PUEBLA.**



FACULTAD DE CIENCIAS DE LA COMPUTACIÓN.



Diseño de Bases de Datos.

Maestro: Carlos Armando Ríos Acevedo.

PRIMAVERA 2020

Objetivos:

- El alumno integrará los conceptos de análisis con el diseño de bases de datos relacionales.
- El alumno entenderá la necesidad de analizar la información como base para modelar una base de datos relacional de manera eficiente.
- Se estudiará y aplicará la normalización dentro del proceso de creación de la base de datos relacional.

MODULO II: Diseño de Bases de Datos.

Índice:	Páginas
1. Fases del diseño de bases de datos.	4
1.1 Recolección y análisis de requerimientos.	4
1.2 Diseño Conceptual.	10
1.3 Diseño lógico de la base de datos (transformación de modelo de datos).	12
1.4 Diseño Físico de la base de datos	13
2. Modelo Conceptual de Bases de Datos	14
2.1 Modelos de Datos.	14
2.2 Metodología del Diseño Conceptual.	16
2.3 Modelo Entidad-Relación.	20
2.3.1 Entidades y Conjunto de Entidades.	21
2.3.2 Relaciones y Conjunto de Relaciones.	22
2.3.3 Tipos de Relaciones.	24
2.3.4 Llaves Primarias.	27
2.3.5 Diagrama Entidad-Relación.	27
2.3.6 Reducción de diagramas E-R a tablas.	30
2.3.7 Generalización y especialización.	30
2.3.8 Agregación.	32
2.3.9 Diagrama E-R "Punto de Venta".	33
2.4 Otros Modelos de Datos.	35
3. Modelado Lógico de Bases de Datos	37
3.1 Introducción.	37
3.2 Objetivos del MR.	38
3.2.1 Estructura del Modelo Relacional.	39
3.2.2 Dominio y Atributo.	42
3.2.3 Claves.	42
3.2.4 Restricciones.	44
3.3 Álgebra Relacional.	49
3.4 Mapeo de los objetos del modelo conceptual al modelo relacional.	59
3.5 Teoría de la Normalización.	66
3.6 Dependencias Funcionales.	67
3.7 Apéndice	79

1. Fases del Diseño de Bases de Datos

1.1 Recolección y Análisis de Requerimientos

Ingeniería de Requerimientos

Facilita y agiliza el mecanismo apropiado para comprender lo que quiere el cliente, analizando necesidades, confirmando su viabilidad, negociando una solución razonable, especificando la solución sin ambigüedad, validando la especificación y gestionando los requisitos para que se transformen en un sistema operacional. Figura 1.1.

Existen 4 fases principales en la Ingeniería de Requerimientos:

1. Estudio de viabilidad.
2. Obtención y análisis de requerimientos.
3. Especificación de Requerimientos.
4. Validación de requerimientos.

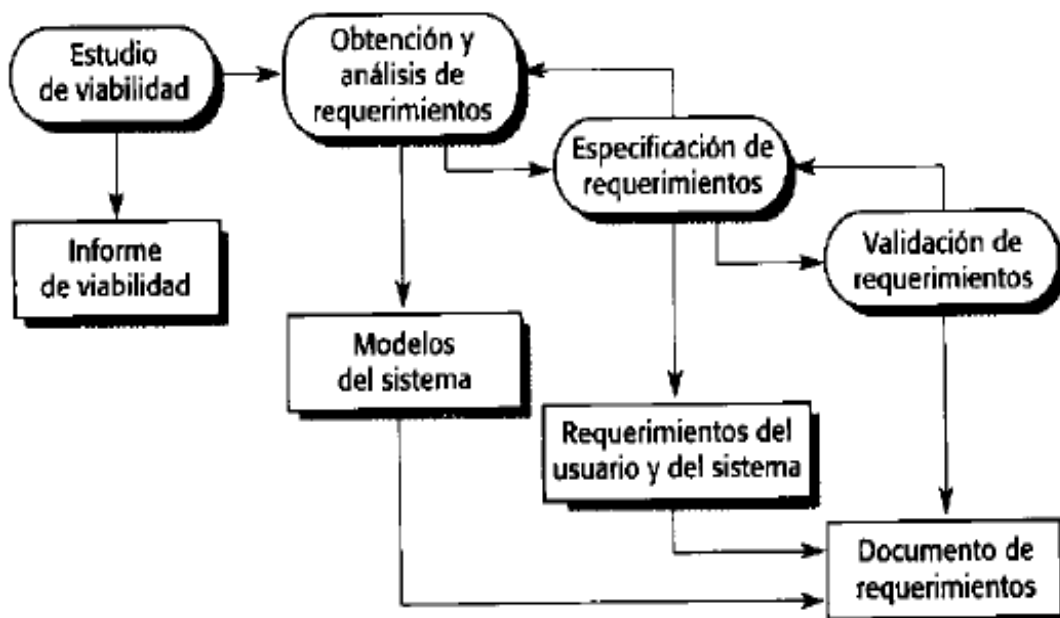


Figura 1.1. El proceso de Ingeniería de Requerimientos

Estudio de viabilidad.

Christel y Kang identifican una serie de problemas que nos ayudan a comprender por qué la obtención de requisitos es costosa.

- *problemas de alcance.* El límite del sistema está mal definido o los detalles técnicos innecesarios, que han sido aportados por los clientes/usuarios, pueden confundir más que clarificar los objetivos del sistema.
- *problemas de comprensión.* Los clientes/usuarios no están completamente seguros de lo que necesitan, no existe un total entendimiento del dominio del problema, existen dificultades para comunicar las necesidades al ingeniero del sistema, la omisión de información o por considerar que es «obvia», especificación de requisitos que están en conflicto con las necesidades de otros clientes/usuarios, o especificar requisitos ambiguos o poco estables.
- *Problemas de volatilidad.* Los requisitos cambian con el tiempo.

Obtención Análisis de Requisitos

Una vez recopilados los requisitos, el producto obtenido configura la base del *análisis de requisitos*. Los requisitos se agrupan por categorías y se organizan en subconjuntos, se estudia cada requisito en relación con el resto, se examinan los requisitos en su consistencia, completitud y ambigüedad, y se clasifican en base a las necesidades de los clientes/usuarios.

En muchas ocasiones los stakeholders (clientes, usuarios, patrocinadores etc.), solicitan reducciones en tiempo de desarrollo de aplicaciones informáticas, disminución de los costos originales presupuestados y/o incrementar el alcance del proyecto. Estas variables mencionadas: tiempo, alcance y costo se vuelven restricciones en el desarrollo de un proyecto de software y ocasionan en su mayor parte que el 80% de los desarrollos de software fracasen.

El ingeniero del sistema debe resolver estos conflictos a través de un proceso de negociación. Los clientes, usuarios y el resto de intervinientes deberán clasificar sus requisitos y discutir los posibles conflictos según su prioridad. Los riesgos asociados con cada requisito serán identificados y analizados. Se efectúan «estimaciones» del esfuerzo de desarrollo que se utilizan para valorar el impacto de cada requisito en el coste del proyecto y en el plazo de entrega. Utilizando un

procedimiento iterativo, se eliminarán requisitos, se combinan y/o modifican para conseguir satisfacer los objetivos planteados. Figura 1.2

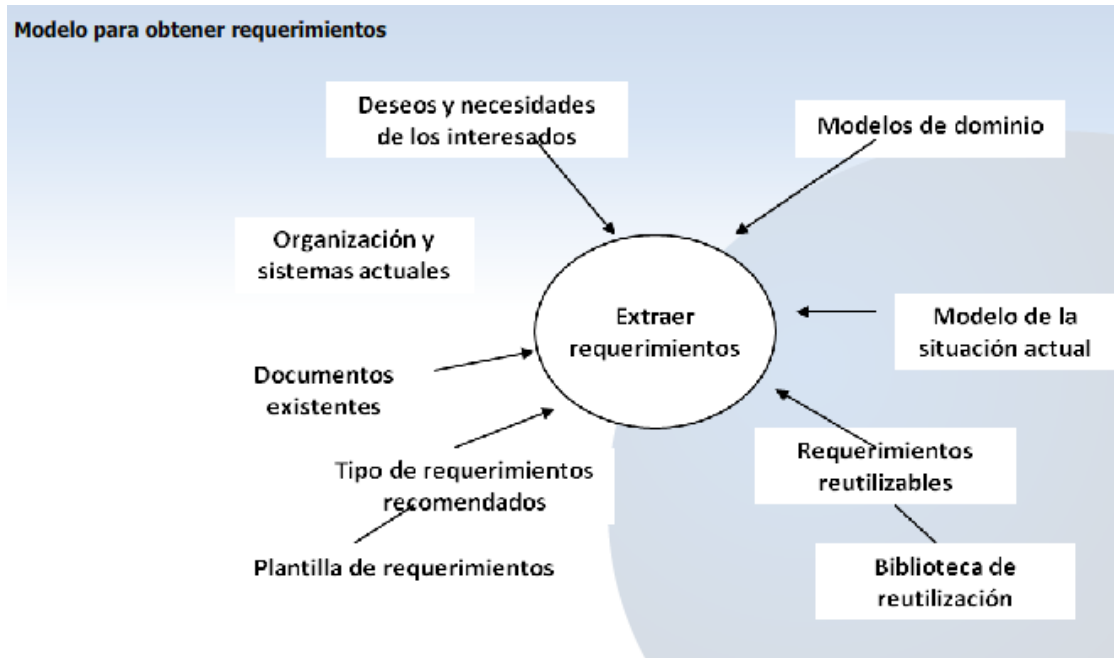


Figura 1.2. Modelo para obtener requerimientos.

Especificación de Requerimientos

La mayor parte de las veces, el cliente no tiene claro qué es lo que realmente necesita. Es el desarrollador el responsable de ayudar al cliente a entender y expresar sus necesidades para que el software las pueda satisfacer. Para identificar los requerimientos se consultan a los interesados a través de varias técnicas [Tomayco] como son:

- Observando el funcionamiento de la empresa.
- Entrevistando al personal de la empresa, concretamente, a aquellos que son considerados expertos en las áreas de interés.
- Revisar documentos o sistemas ya existentes que se pretenden mejorar.
- Utilizando cuestionarios para recoger información de grandes grupos de usuarios.
- Utilizando la experiencia adquirida en el diseño de sistemas similares.

Los requerimientos son el punto de partida para la construcción de un producto de software. Los Requerimientos de software son un área de la Ingeniería de Software dedicada al estudio de la adquisición, análisis, especificación, validación y administración de los requerimientos o necesidades de

un software. [SWEBOK 2001]. Es una fase fundamental para la construcción de software con calidad. Se entiende por calidad en un producto de software aquel que cumple con las necesidades del usuario.

La especificación de requerimientos es como un mapa para entender el problema a resolver. Los mapas son una ayuda a llegar más rápido al lugar deseado. Si no se tiene el mapa, se puede manejar a toda velocidad, pero no necesariamente se llegará a la meta.

Una característica de los requerimientos es que cambian constantemente por muchas razones: se modifican las necesidades del cliente, cambia el ambiente, la tecnología, etc. por lo que establecer los requerimientos es un proceso de negociación entre el cliente y los desarrolladores, donde ambos entienden y acuerdan el objetivo del software.

El texto deberá escribirse preferentemente en un lenguaje que entienda el cliente sin tecnicismos computacionales.

Coad propone una serie de estrategias al redactar ese texto para ayudar a aclarar el objetivo del software. Una de ellas es escribir una frase del tipo *"para ayudar a"* o *"para apoyar a"* que permiten saber el tipo de usuarios que tendrá el software y para qué se usará. Otra estrategia que propone es que el desarrollador *"se dé una vuelta por el ambiente de trabajo donde se usará el software"*, esto permite ver a los usuarios y el tipo de cosas que necesitan que el sistema haga para apoyarlos en su trabajo diario. Una estrategia más, indica que *"se haga una lista de características o cualidades"* que deberá tener el software, la que se puede escribir en orden de importancia.

Los requerimientos deben formularse de forma clara, precisa y no ambigua. Para eso pueden usarse varias técnicas al mismo tiempo: el lenguaje natural, que es claro para el cliente, pero ambiguo; el modelado gráfico, que es más claro para el desarrollador y no es ambiguo, pero puede no ser claro para el cliente; prototipo de interfaz de usuario, útil para ambos pues es una representación visual de lo que hará el software.

Hay dos tipos de requerimientos: los funcionales y los no funcionales.

Los **funcionales** incluyen:

- Las entradas y salidas de datos, los cálculos o las funciones del software.
- Los requerimientos funcionales de un sistema describen la funcionalidad o los servicios que se espera que éste provea. Estos dependen del tipo de software y del sistema que se desarrolle y de los posibles usuarios del software. Cuando se expresan como requerimientos del usuario,

habitualmente se describen de forma general, mientras que los requerimientos funcionales del sistema describen con detalle la función de éste, sus entradas y salidas, excepciones, etc.

Los **no funcionales son las características o restricciones que se le piden al software:**

- Necesidades de la interfaz externa como: tipo de usuario, hardware, software, comunicaciones, facilidad de uso por sus usuarios.
- Atributos del software tales como: eficiencia, disponibilidad, seguridad, conversión, portabilidad, mantenimiento.
- Restricciones del diseño: de formatos, de archivos, lenguajes, estándares, compatibilidad.
- Otros: base de datos, instalación, etc.

Los requerimientos no funcionales pueden ser:

- **Interfaz con el usuario:** descripción de las características que permitan al software apoyar al usuario en sus tareas.
- **Interfaz externa:** relación o conexión con otros sistemas.
- **Confiabilidad:** solicitud del desempeño respecto a seguridad, tolerancia a fallas y su recuperación.
- **Eficiencia:** límites de tiempo de respuesta.
- **Mantenimiento:** facilidad de comprensión y realización de modificaciones futuras.
- **Portabilidad:** facilidad de transferencia de un ambiente de ejecución a otro.
- **Restricciones de diseño y construcción:** las que imponga el cliente.
- **Legales y reglamentarios:** necesidades impuestas por leyes o reglamentos de otros.

Estos diferentes tipos de requerimientos se clasifican de acuerdo a sus implicaciones.

- **Requerimientos del Producto.** Especifican el comportamiento del producto; como los requerimientos de desempeño en la rapidez de ejecución del sistema y cuánta memoria se requiere; los de fiabilidad que fijan la tasa de fallas para que el sistema sea aceptable.
- **Requerimientos del Negocio.** Son importantes debido que reflejan los fundamentos del dominio de aplicación. Si estos requerimientos no se satisfacen, es imposible comprender las necesidades y funcionamiento del sistema.

- **Requerimientos del Usuario.** Describen los requerimientos funcionales y no funcionales de tal forma que sean comprensibles por los usuarios del sistema que no posean un conocimiento técnico detallado. Deben redactarse utilizando un lenguaje natural, representaciones y diagramas intuitivos sencillos. Evitar los tecnicismos.
- **Requerimientos del Sistema.** Establecen con detalle los servicios y restricciones del sistema. Son descripciones detalladas de los requerimientos del usuario. Sirven como base para definir el contrato de la especificación y, por lo tanto, debe ser una especificación completa y consistente del sistema. Son utilizados por los ingenieros de software como el punto de partida para el diseño del sistema. Deberán establecer lo que el sistema hará y no la manera en que se implementará.

Validación de Requerimientos.

Esta actividad comprueba la veracidad, consistencia y completitud de los requerimientos. Durante este proceso, inevitablemente se descubren errores en el documento de requerimientos. Se debe modificar entonces para corregir estos problemas.

El documento de especificaciones de requerimientos del software (SRS) es la declaración oficial de qué deben implementar los desarrolladores del sistema. Debe incluir tanto los requerimientos del usuario para el sistema como una especificación detallada de los requerimientos del sistema.

El documento de requerimientos tiene un conjunto diverso de usuarios que va desde los altos cargos de la organización que pagan por el sistema, hasta los ingenieros responsables de desarrollar el software. La diversidad de posibles usuarios significa que el documento de requerimientos tiene que presentar un equilibrio entre la comunicación de los requerimientos a los clientes, la definición de los requerimientos en el detalle exacto para los desarrolladores y probadores, y la inclusión de información sobre la posible evolución del sistema. Figura 1.3

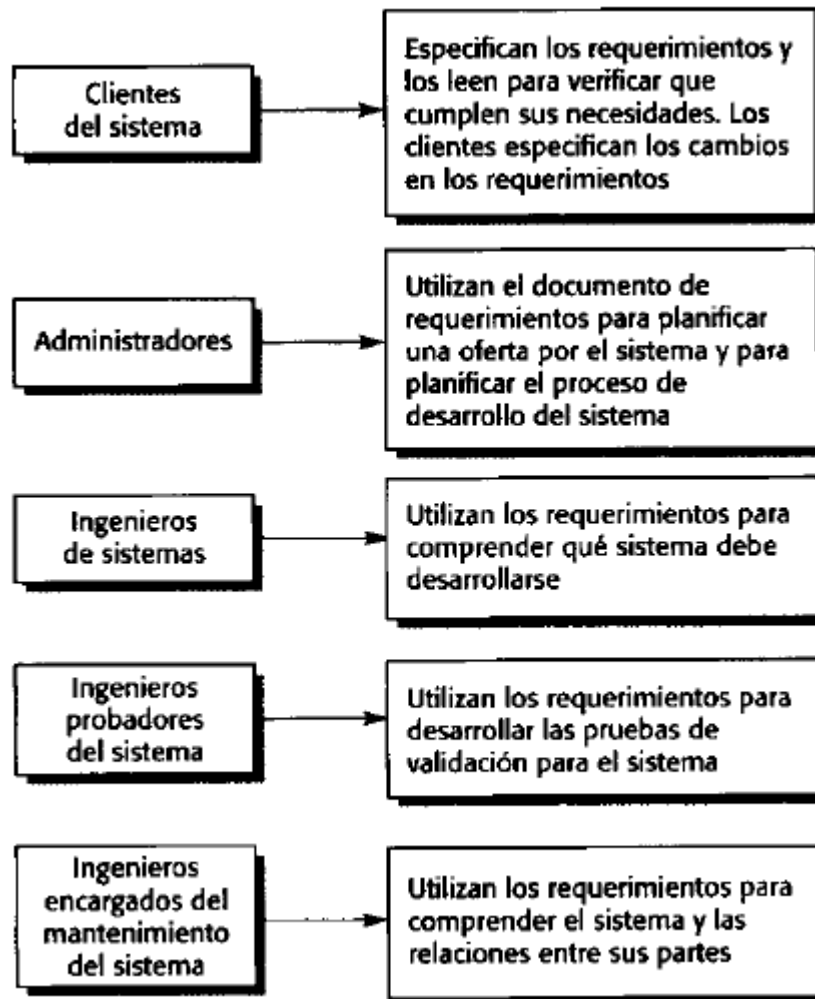


Figura 1.3. Usuarios de un documento de Requerimientos.

1.2. Diseño conceptual

En esta etapa se debe construir un esquema de la información que se usa en la empresa, de cualquier consideración física. A este esquema se le denomina esquema conceptual. Al construir el esquema, los diseñadores descubren la semántica (significado) de los datos de la empresa: encuentran entidades, atributos y relaciones. El objetivo es comprender:

- La perspectiva que cada usuario tiene de los datos
- La naturaleza de los datos,
- Independientemente de su representación física.

El uso de los datos a través de las áreas de aplicación. El esquema conceptual se puede utilizar para que el diseñador transmita a la empresa lo que ha entendido sobre la información que ésta maneja. Para ello, ambas partes deben estar familiarizadas con la notación utilizada en el esquema. La más popular es la notación del modelo entidad-relación. El esquema conceptual se construye utilizando la información que se encuentra en la especificación de los requisitos de usuario. El diseño conceptual es completamente independiente de los aspectos de implementación, como puede ser el Sistema Gestor de Bases de Datos (SGBD) que se vaya a usar, los programas de aplicación, los lenguajes de programación, el hardware disponible o cualquier otra consideración física. Durante todo el proceso de desarrollo del esquema conceptual éste se prueba y se valida con los requisitos de los usuarios. El esquema conceptual es una fuente de información para el diseño lógico de la base de datos. Figura 1.4

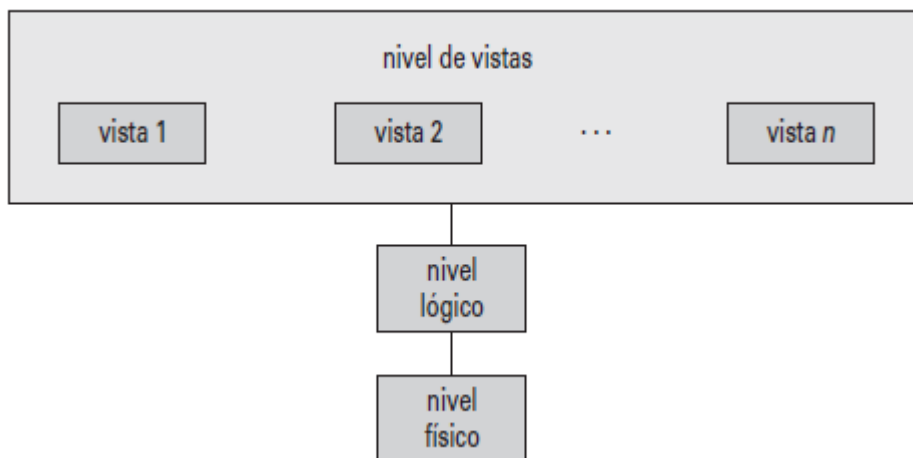


Figura 1.4. Los tres niveles de abstracción de bases de datos

1.3. Diseño lógico de la base de datos (transformación de modelo de datos)

El diseño lógico es el proceso de construir un esquema de la información que utiliza la empresa, basándose en un modelo de base de datos específico, independiente del SGBD concreto que se vaya a utilizar y de cualquier otra consideración física.

En esta etapa, se transforma el esquema conceptual en un esquema lógico que utilizará las estructuras de datos del modelo de base de datos en el que se basa el SGBD que se vaya a utilizar, como puede ser el modelo relacional, el modelo de red, el modelo jerárquico o el modelo orientado a objetos. Conforme se va desarrollando el esquema lógico, éste se va probando y validando con los requisitos de usuario.

La normalización es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional, ya que asegura que las relaciones (tablas) obtenidas no tienen datos redundantes.

El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos.

Tanto el diseño conceptual, como el diseño lógico, son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. Ambos se deben ver como un proceso de aprendizaje en el que el diseñador va comprendiendo el funcionamiento de la empresa y el significado de los datos que maneja. El diseño conceptual y el diseño lógico son etapas clave para conseguir un sistema que funcione correctamente. Si el esquema no es una representación fiel de la empresa, será difícil, sino imposible, definir todas las vistas de usuario (esquemas externos), o mantener la integridad de la base de datos. También puede ser difícil definir la implementación física o el mantener unas prestaciones aceptables del sistema. Además, hay que tener en cuenta que la capacidad de ajustarse a futuros cambios es un sello que identifica a los buenos diseños de bases de datos.

Por todo esto, es fundamental dedicar el tiempo y las energías necesarias para producir el mejor esquema que sea posible.

1.4. Diseño físico.

El diseño físico es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos.

Para llevar a cabo esta etapa, se debe haber decidido cuál es el SGBD que se va a utilizar, ya que el esquema físico se adapta a él. Entre el diseño físico y el diseño lógico hay una realimentación, ya que algunas de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico.

En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior. Concretamente, en el modelo relacional, esto consiste en:

- Obtener un conjunto de relaciones (tablas) y las restricciones que se deben cumplir sobre ellas.

2. Modelo Conceptual de Bases de Datos

2.1 Modelos de Datos.

La modelación de datos nos permite abstraer de problemas del mundo real los datos (objetos o entidades) involucrados y las relaciones que existen entre ellos.

Un modelo de datos es una serie de conceptos que puede utilizarse para describir un conjunto de datos y las operaciones para manipularlos. Hay dos tipos de modelos de datos: los modelos conceptuales y los modelos lógicos. Los modelos conceptuales se utilizan para representar la realidad a un alto nivel de abstracción. Mediante los modelos conceptuales se puede construir una descripción de la realidad fácil de entender. En los modelos lógicos, las descripciones de los datos tienen una correspondencia sencilla con la estructura física de la base de datos.

En el diseño de bases de datos se usan primero los modelos conceptuales para lograr una descripción de alto nivel de la realidad, y luego se transforma el esquema conceptual en un esquema lógico. El motivo de realizar estas dos etapas es la dificultad de abstraer la estructura de una base de datos que presente cierta complejidad. Un esquema es un conjunto de representaciones lingüísticas o gráficas que describen la estructura de los datos de interés.

Los modelos conceptuales deben ser buenas herramientas para representar la realidad, por lo que deben poseer las siguientes cualidades:

- **Expresividad:** deben tener suficientes conceptos para expresar perfectamente la realidad.
- **Simplicidad:** deben ser simples para que los esquemas sean fáciles de entender.
- **Minimalidad:** cada concepto debe tener un significado distinto.
- **Formalidad:** todos los conceptos deben tener una interpretación única, precisa y bien definida.

Es una colección de herramientas conceptuales para la descripción de datos, relaciones entre datos, semántica de los datos y restricciones de consistencia.

- Los modelos de datos describen las relaciones entre los datos que forman una base de datos.
- No se refieren en ningún momento a los valores específicos que un elemento de datos debe tomar.
- Tratan a los datos como grupos genéricos, que pueden tomar cualquier conjunto de valores específicos

Representación de Datos

- ✓ **Ítems/Entidades/Objetos** [sustantivos]:
Objetos que existen en el "mundo" y que son distinguibles de otros (un libro, un autor, un tema...).
- ✓ **Atributos** [adjetivos]:
Propiedades asociadas a un conjunto de entidades (ISBN, nombre...).
- ✓ **Relaciones/Conexiones/Asociaciones** [verbos]:
Conexiones semánticas entre dos conjuntos de entidades (escribe, trata...).

Un *esquema* es una descripción de una colección particular de datos usando un modelo de datos específico.

Un Sistema Manejador de Bases de Datos (SGBD) soporta un modelo de datos, que es usado para describir el esquema de la base de datos a utilizar.

Existen varios modelos de datos. Figura 2.1.

- Modelo Basado en Grafos (Jerárquico y de Red).
- Modelo Entidad-Relación (E-R).
- Modelo Entidad-Relación Extendido (EER).
- Modelo Orientado a Objetos.
- Modelo Multidimensional.
- Modelo Relacional.

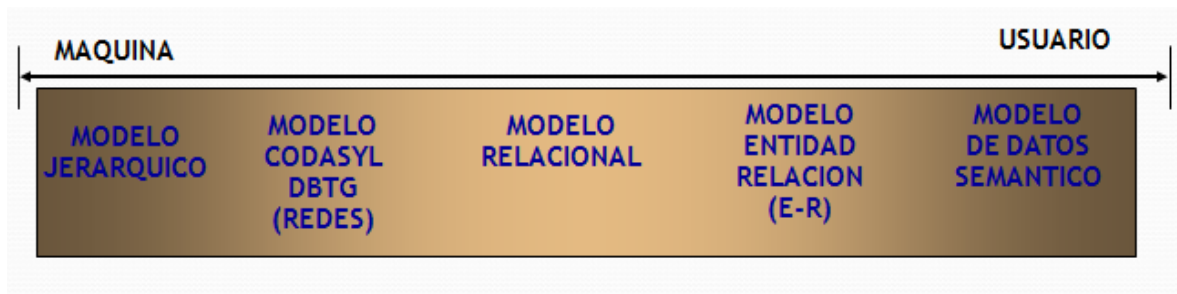


Figura 2.1 Modelos de Datos (Máquina-Usuario).

2.2 Metodología de Diseño Conceptual

El primer paso en el diseño de una base de datos es la producción del esquema conceptual. Normalmente, se construyen varios esquemas conceptuales, cada uno para representar las distintas visiones que los usuarios tienen de la información. Cada una de estas visiones suelen corresponder a las diferentes áreas funcionales de la empresa como, por ejemplo, producción, ventas, recursos humanos, etc.

Estas visiones de la información, denominadas vistas, se pueden identificar de varias formas. Una opción consiste en examinar los diagramas de flujo de datos, que se pueden haber producido previamente, para identificar cada una de las áreas funcionales. La otra opción consiste en entrevistar a los usuarios, examinar los procedimientos, los informes y los formularios, y también observar el funcionamiento de la empresa.

A los esquemas conceptuales correspondientes a cada vista de usuario se les denomina esquemas conceptuales locales. Cada uno de estos esquemas se compone de entidades, relaciones, atributos, dominios de atributos e identificadores. El esquema conceptual también tendrá una documentación, que se irá produciendo durante su desarrollo. Las tareas a realizar en el diseño conceptual son las siguientes:

1. Identificar las entidades.
2. Identificar las relaciones.
3. Identificar los atributos y asociarlos a entidades y relaciones.
4. Determinar los dominios de los atributos.

5. Determinar los identificadores.
6. Determinar las jerarquías de generalización (si las hay).
7. Dibujar el diagrama entidad-relación.
8. Revisar el esquema conceptual local con el usuario.

1. Identificar las entidades

En primer lugar hay que definir los principales objetos que interesan al usuario. Estos objetos serán las entidades. Una forma de identificar las entidades es examinar las especificaciones de requisitos de usuario. En estas especificaciones se buscan los nombres o los sintagmas nominales que se mencionan (por ejemplo: número de empleado, nombre de empleado, número de inmueble, dirección del inmueble, alquiler, número de habitaciones). También se buscan objetos importantes como personas, lugares o conceptos de interés, excluyendo aquellos nombres que sólo son propiedad desde otros objetos. Por ejemplo, se pueden agrupar el número de empleado y el nombre de empleado en una entidad denominada empleado, y agrupar número de inmueble, dirección del inmueble, alquiler y número de habitaciones en otra entidad denominada inmueble.

Otra forma de identificar las entidades es buscar aquellos **objetos** que existen por sí mismos. Por ejemplo, empleado es una entidad porque los empleados existen, sepamos o no sus nombres, direcciones y teléfonos. Siempre que sea posible, el usuario debe colaborar en la identificación de las entidades. A veces, es difícil identificar las entidades por la forma en que aparecen en las especificaciones de requisitos. Los usuarios, a veces, hablan utilizando ejemplos o analogías. En lugar de hablar de empleados en general, hablan de personas concretas, o bien, hablan de los puestos que ocupan esas personas.

2. Identificar las relaciones

Una vez definidas las entidades, se deben definir las relaciones existentes entre ellas. Del mismo modo que para identificar las entidades se buscaban nombres en las especificaciones de requisitos, para identificar las relaciones se suelen buscar las expresiones verbales (por ejemplo: oficina tiene empleados, empleado gestiona inmueble, cliente visita inmueble). Si las especificaciones de

requisitos reflejan estas relaciones es porque son importantes para la empresa y, por lo tanto, se deben reflejar en el esquema conceptual.

Una vez identificadas todas las relaciones, hay que determinar la **cardinalidad mínima y máxima** con la que participa cada entidad en cada una de ellas. De este modo, el esquema representa de un modo más explícito la semántica de las relaciones. La cardinalidad es un tipo de restricción que se utiliza para comprobar y mantener la calidad de los datos. Estas restricciones son aserciones sobre las entidades que se pueden aplicar cuando se actualiza la base de datos para determinar si las actualizaciones violan o no las reglas establecidas sobre la semántica de los datos.

Conforme se van identificando las relaciones, se les van asignando nombres que tengan significado para el usuario. En el diccionario de datos se anotan los nombres de las relaciones, su descripción y las cardinalidades con las que participan las entidades en ellas.

3. Identificar los atributos y asociarlos a entidades y relaciones

Al igual que con las entidades, se buscan nombres en las especificaciones de requisitos. Son atributos los nombres que identifican propiedades, cualidades, identificadores o características de entidades o relaciones.

4. Determinar los dominios de los atributos

El dominio de un atributo es el conjunto de valores que puede tomar el atributo. Un esquema conceptual está completo si incluye los dominios de cada atributo: los valores permitidos para cada atributo, su tamaño y su formato.

5. Determinar los identificadores

Cada entidad tiene al menos un identificador. En este paso, se trata de encontrar todos los identificadores de cada una de las entidades. Los identificadores pueden ser simples o compuestos. De cada entidad se escogerá uno de los identificadores como **clave primaria** en la fase del diseño lógico. Todos los identificadores de las entidades se deben anotar en el diccionario de datos.

6. Determinar las jerarquías de generalización

En este paso hay que observar las entidades que se han identificado hasta el momento. Hay que ver si es necesario reflejar las diferencias entre distintas ocurrencias de una entidad, con lo que surgirán nuevas subentidades de esta entidad genérica; o bien, si hay entidades que tienen características en común y que realmente son subentidades de una nueva entidad genérica. En cada jerarquía hay que determinar si es total o parcial y exclusiva o superpuesta.

7. Dibujar el diagrama entidad-relación

Una vez identificados todos los conceptos, se puede dibujar el diagrama entidad-relación correspondiente a una de las vistas de los usuarios. Se obtiene así un esquema conceptual local.

8. Revisar el esquema conceptual local con el usuario

Antes de dar por finalizada la fase del diseño conceptual, se debe revisar el esquema conceptual local con el usuario. Este esquema está formado por el diagrama entidad-relación y toda la documentación que describe el esquema. Si se encuentra alguna anomalía, hay que corregirla haciendo los cambios oportunos, por lo que posiblemente haya que repetir alguno de los pasos anteriores. Este proceso debe repetirse hasta que se esté seguro de que el esquema conceptual es una fiel representación de la parte de la empresa que se está tratando de modelar.

2.3 Modelo Entidad-Relación

El **modelo entidad-relación (E-R)** es un concepto de modelado de alto nivel para bases de datos, propuesto por Peter Chen en el año de 1976. Ver figura 2.2.



Figura 2.2 Peter Chen (M.I.T.) en los 70's.

Originalmente, el modelo entidad-relación sólo incluía los conceptos de entidad, relación y atributo. Más tarde, se añadieron otros conceptos, como los atributos compuestos y las jerarquías de generalización, en lo que se ha denominado **modelo entidad-relación extendido**.

(E-R) es una representación conceptual de la información. Mediante una serie de procedimientos se puede pasar del modelo E-R a otros, por ejemplo al modelo relacional. Los diagramas E-R son un lenguaje gráfico para describir conceptos. El modelo (E-R) es un modelo informal, sin embargo ayuda como teoría inicial para modelar estructuras de datos y asistir al diseñador para definir y entender las cosas significativas acerca de cuál información necesita ser conocida o manejada.

Para modelar un diagrama de E-R, se necesita experiencia y muy buen entrenamiento para lograr resultados correctos. A continuación se detallan algunos de los pasos para modelar Diagramas E-R:

- 1) Se parte de una descripción textual del problema (Especificación de Requerimientos de Software "SRS").
- 2) Se hace una lista de los sustantivos y verbos que aparecen.
- 3) Los sustantivos son posibles entidades o atributos.
- 4) Los verbos son posibles relaciones.
- 5) Analizando los párrafos se determina la cardinalidad de las relaciones y otros detalles.

- 6) Se elabora el diagrama entidad-relación.
- 7) Se completa el modelo con listas de atributos y una descripción de otras restricciones que no se reflejan en el diagrama.

2.3.1 Entidades y conjunto de entidades

Una **entidad** es una «cosa» u «objeto» en el mundo real que es distinguible de todos los demás objetos. Una entidad tiene un conjunto de propiedades, y los valores para algún conjunto de propiedades pueden identificar una entidad de forma unívoca. [Silberschatz, 2004].

Es todo objeto de datos que es diferenciable de otros objetos, ya sean abstractos o concretos.

Un **conjunto de entidades** es un conjunto de entidades del mismo tipo que comparten las mismas propiedades, o atributos. [Silberschatz, 2004].

Como un conjunto de entidades puede tener diferentes atributos, cada entidad se puede describir como un conjunto de pares (**atributo, valor**), un par para cada atributo del conjunto de entidades.

Entidades **fuertes y débiles**. Cuando una entidad participa en una relación puede adquirir un papel fuerte o débil. Una entidad débil es aquella que no puede existir sin participar en la relación, es decir, aquella que no puede ser unívocamente identificada solamente por sus atributos. Una entidad fuerte es aquella que sí puede ser identificada unívocamente. En los algunos casos, se puede dar que una entidad fuerte “preste alguno de sus atributos” a una entidad débil.

Atributos

Permite describir a una entidad. Los atributos describen propiedades o características de una entidad y/o relación. Los atributos describen información útil sobre las entidades. En particular, los atributos identificativos son aquellos que permiten diferenciar a una instancia de la entidad de otra distinta. Por ejemplo el atributo identificativo que diferencia a un alumno de otros es su matrícula de estudiante. Ver figura 2.3

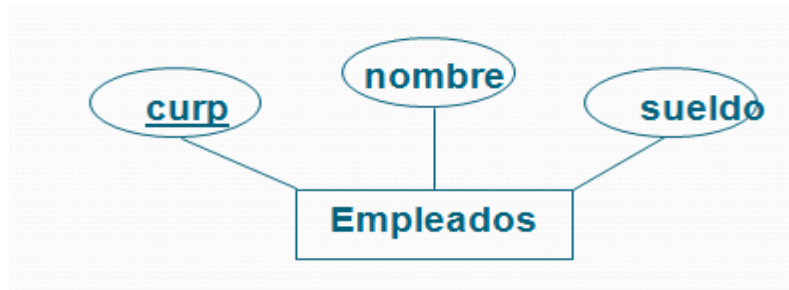


Figura 2.3. Entidad empleados con atributos.

Para cada atributo hay un conjunto de valores permitidos, llamados el **dominio**, o el **conjunto de valores**, de ese atributo.

Un atributo, como se usa en el modelo E-R, se puede caracterizar por los siguientes tipos de atributo:

- **Atributos simples.** Aquellos que no están divididos en subpartes. Por ejemplo apellido paterno, apellido materno etc.
- **Atributos compuestos.** Aquellos que se pueden dividir en atributos simples. Por ejemplo el atributo dirección se puede dividir en: Calle, Colonia, Número, Código Postal, etc.
- **Atributos Multivalorados.** Aquellos que tienen un conjunto de valores para una entidad específica. Por ejemplo el atributo cuenta de correo.
- **Atributos identificativos o Claves.** Son aquellos que permiten diferenciar a una instancia de la entidad de otra distinta. Por ejemplo el atributo identificativo que diferencia a un alumno de otros es su matrícula de estudiante.
- **Atributos Derivados.** El valor de este atributo se puede derivar o calcular a partir de otros atributos. Por ejemplo el atributo Importe se puede calcular multiplicando el atributo precio de venta por la cantidad de unidades compradas.

2.3.2 Relaciones y Conjunto de Relaciones

Es una correspondencia o asociación entre dos o más entidades. Cada relación tiene un nombre que describe su función. Las relaciones se representan gráficamente mediante rombos y su nombre aparece en el interior. Las entidades

que están involucradas en una determinada relación se denominan **entidades participantes**.

El número de participantes en una relación es lo que se denomina **grado** de la relación. Por lo tanto, una relación en la que participan dos entidades es una relación binaria; si son tres las entidades participantes, la relación es ternaria; etc. Ver figura 2.4.

Atributos en las relaciones.

Las relaciones también pueden tener atributos asociados. Se representan igual que los atributos de las entidades. Un ejemplo típico son las relaciones de tipo "histórico" donde debe constar una fecha u hora de emisión.

Relación: Describe la conexión o asociación existente entre dos o más entidades.

Por ejemplo

- EMPLEADOS trabajan en DEPARTAMENTOS es una relación BINARIA. Figura 2.4.

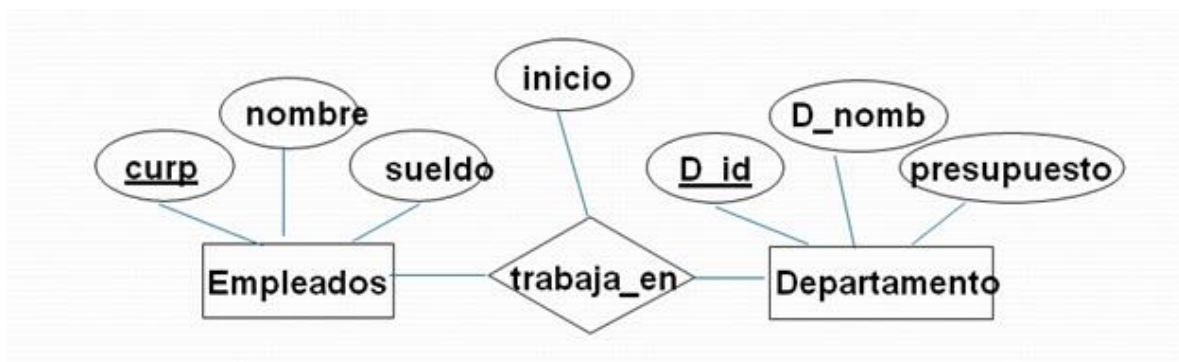


Figura 2.4 Ejemplo de relación Binaria.

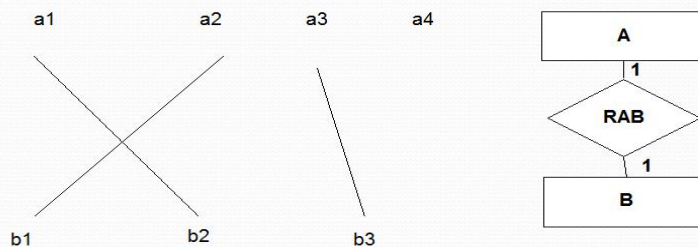
- ¿El empleado JOSÉ en cuántos departamentos trabaja?
- ¿En el departamento de CONTABILIDAD cuántos empleados trabajan?

2.3.3 Tipos de relaciones

La **cardinalidad** con la que una entidad participa en una relación especifica el número mínimo y el número máximo de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad. La participación de una entidad en una relación es obligatoria (total) si la existencia de cada una de sus ocurrencias requiere la existencia de, al menos, una ocurrencia de la otra entidad participante. Si no, la participación es opcional (parcial). Las reglas que definen la cardinalidad de las relaciones son las reglas de negocio.

Relaciones con cardinalidad 1:1

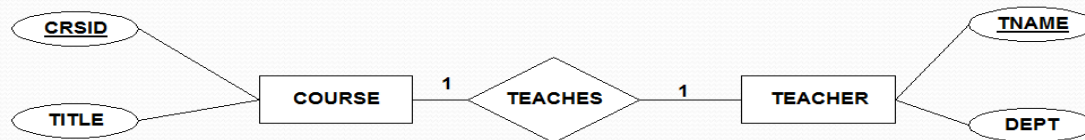
- Una instancia de la entidad **A** está asociada con **0** o **1** instancia de la entidad **B**
- Una instancia de la entidad **B** está asociada con **0** o **1** instancia de la entidad **A**



Ejemplo Relación 1:1

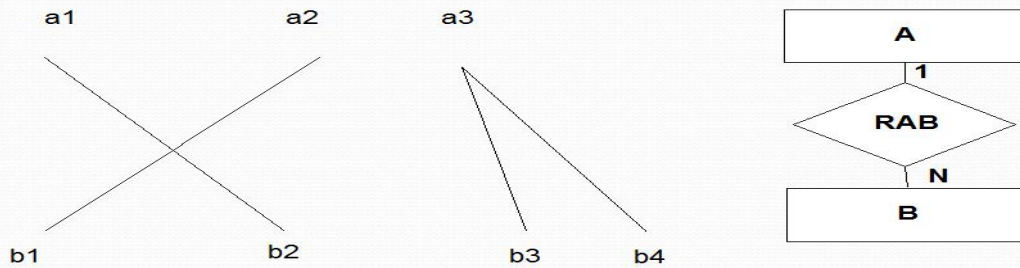
1 curso es impartido por **1 profesor**

1 profesor imparte **1 curso**



Relaciones con cardinalidad 1:N

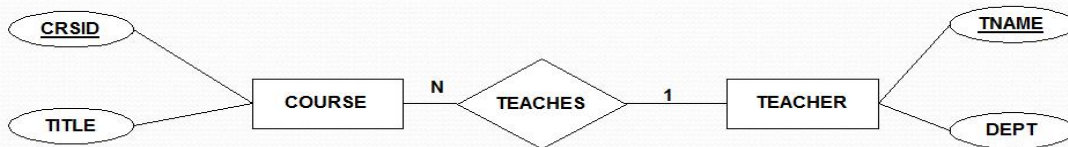
- Una instancia de la entidad **A** está asociada con **0** o **más** instancias de la entidad **B**
- Una instancia de la entidad **B** está asociada con **0** o **1** instancia de la entidad **A**



Ejemplo relación 1:N

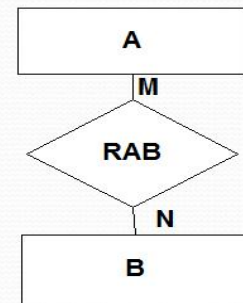
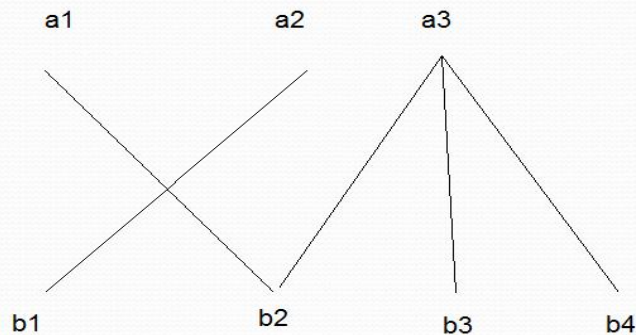
1 curso es impartido por máximo 1 profesor

1 profesor imparte CERO o MAS cursos



Relaciones con cardinalidad M:N

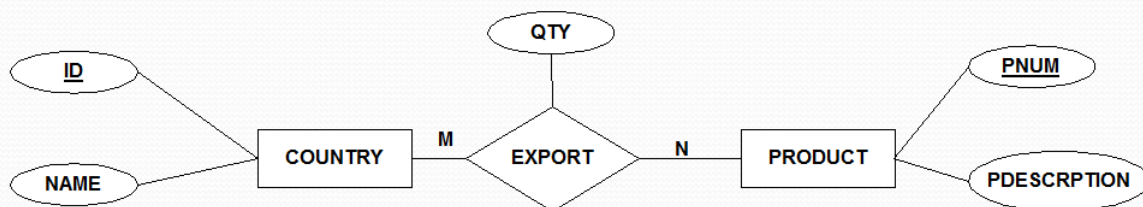
- Una instancia de la entidad **A** está asociada con **0 o más** instancias de la entidad **B**
- Una instancia de la entidad **B** está asociada con **0 o más** instancias de la entidad **A**



Ejemplo Relación M:N

1 país exporta **CERO o MAS** productos

1 producto es exportado por **CERO o MAS** países



1 país exporta **máximo N** productos

1 producto es exportado por **máximo N** países

2.3.4 Llaves Primarias

Una **clave** permite identificar un conjunto de atributos suficiente para distinguir las entidades entre sí. Las claves también ayudan a identificar unívocamente a las relaciones y así a distinguir las relaciones entre sí. [Silberschatz, 2004].

Por lo tanto, los valores de los atributos de una entidad deben ser tales que permitan *identificar unívocamente* a la entidad.

Una **superclave** es un conjunto de uno o más atributos que, tomados colectivamente, permiten identificar de forma única una entidad en el conjunto de entidades.

Llaves candidatas: llaves que posee una relación, número mínimo de atributos.

Una **Clave primaria** sirve para denotar una clave candidata que es elegida por el diseñador de la base de datos como elemento principal para identificar las entidades dentro de un conjunto de entidades. Buscando que posea el menor número de atributos, y que no pueda tener valores nulos.

2.3.5 Diagrama Entidad-Relación

Representa gráficamente y de manera lógica toda la información y como los datos se relacionan entre sí. **(E-R)** está basado en una percepción del mundo real consistente en objetos básicos llamados **entidades** y de **relaciones** entre estos objetos. Ver Figura 2.5.

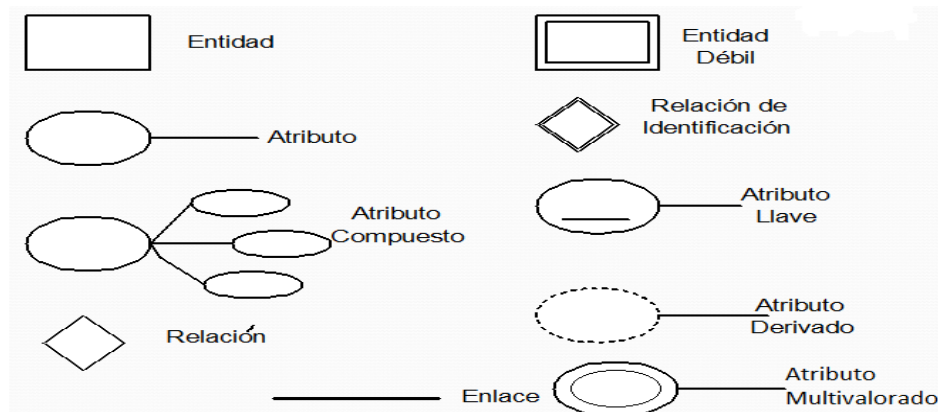


Figura 2.5. Elementos de E-R

Por ejemplo tenemos a continuación el diagrama entidad-relación para un punto de venta de abarrotes sencillo. Ver figura 2.6.

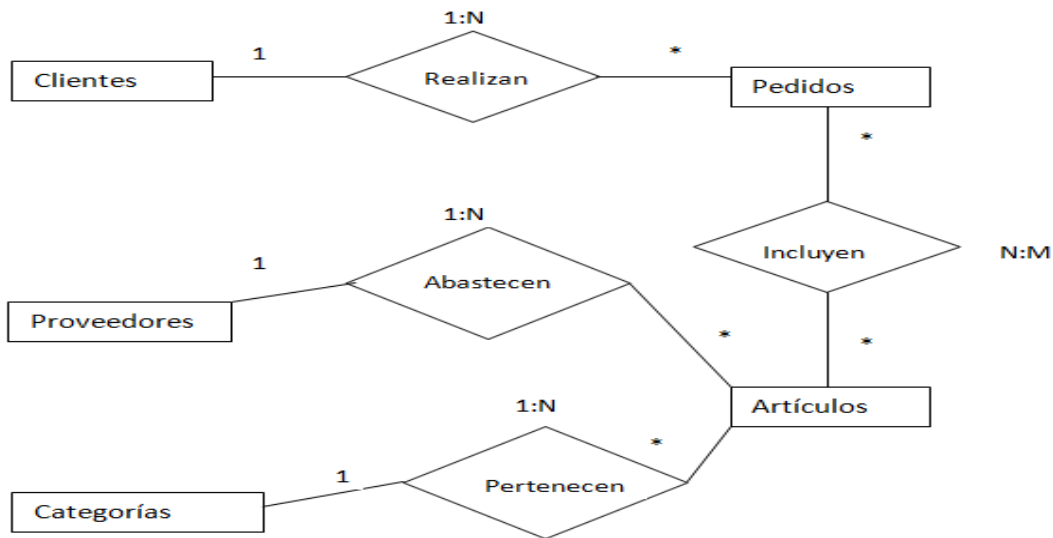


Figura 2.6 Diagrama entidad-relación para un punto de venta.

Notación E-R Alternativas. Ver Figura 2.7

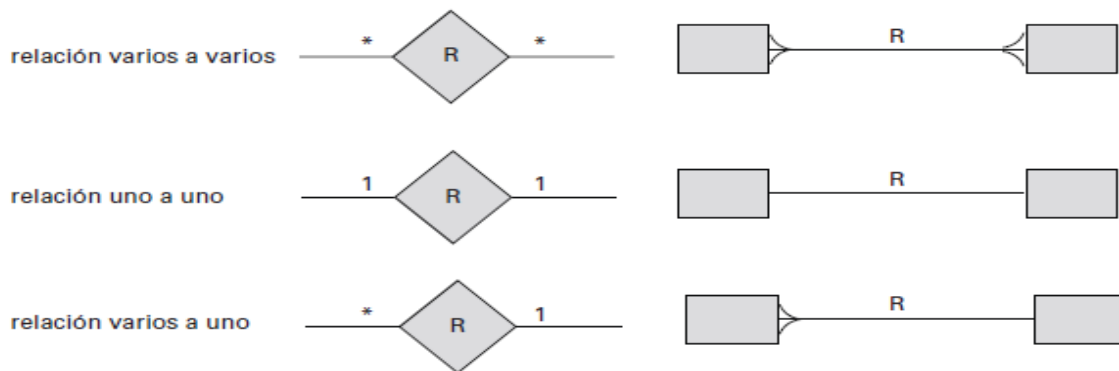
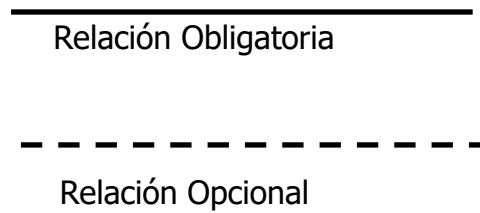
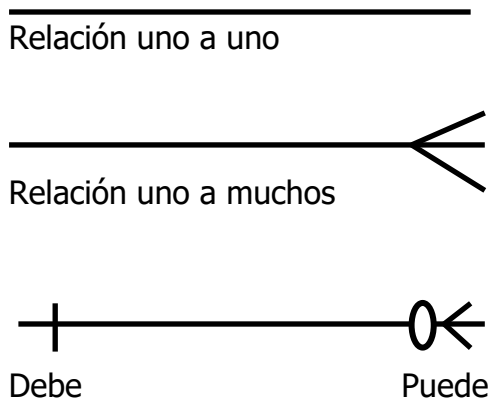


Figura 2.7 Notación E-R Alternativas

Otras notaciones a usarse en los diagramas de entidad-relación son:

1. Los rectángulos representan una entidad o conjunto de entidades.
2. Los círculos representan atributos. (ligados a sus conjuntos de entidades mediante flechas sin dirección).
3. Los rombos o rectángulos redondeados representan relaciones ligadas a las entidades por flechas no dirigidas, aunque muchas, con el fin de simplificar

el diagrama, no se dibuja el rectángulo y en su lugar solo se etiqueta a las flechas que representan la relación.



La técnica para llegar a un modelo entidad-relación, se puede iniciar con un prototipo inicial y gradualmente irlo depurando, ganando precisión e identificando los elementos necesarios para modelar el negocio. Al principio se puede identificar cada entidad con al menos los atributos principales. Figura 2.8

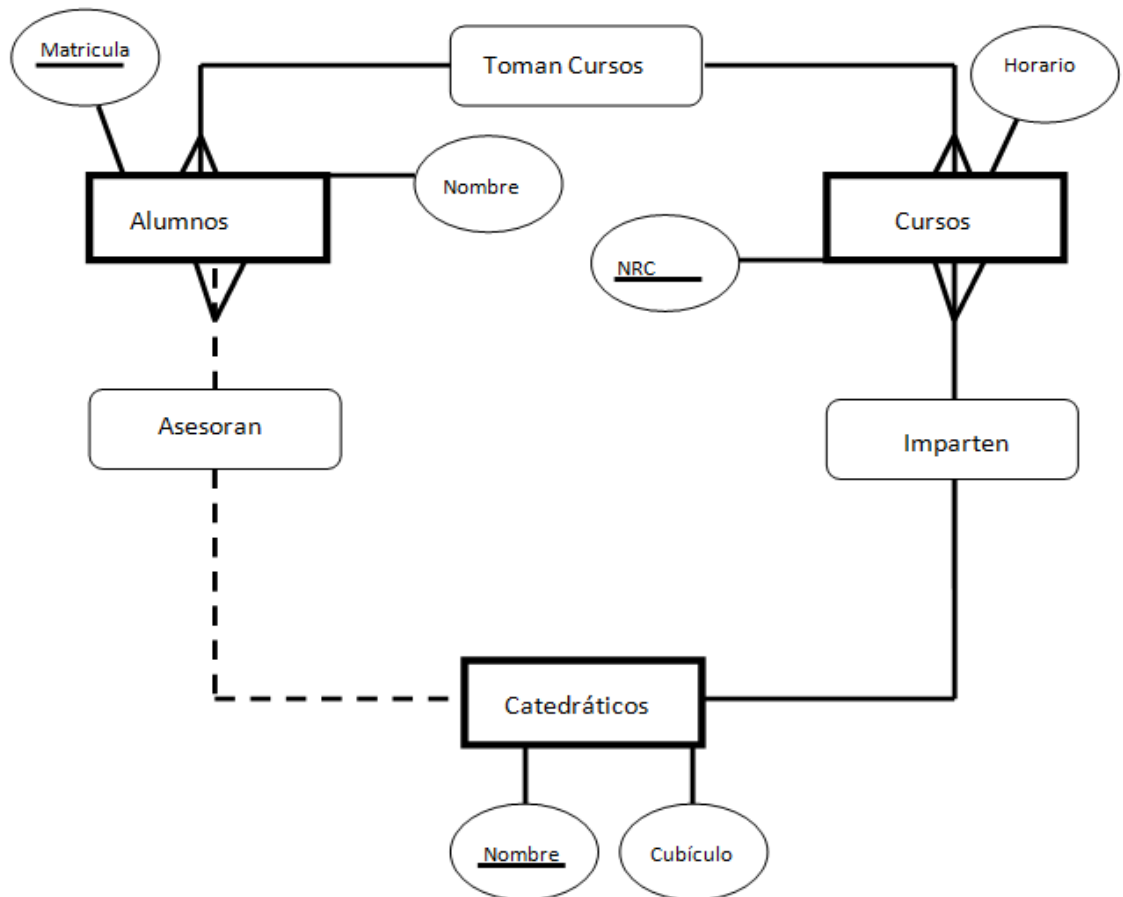


Figura 2.8. Diagrama entidad-relación

2.3.5 Reducción de diagramas E-R a tablas

Atributo:

Los atributos son empleados para identificar, describir, calificar ó expresar el estado de una entidad.

Toda entidad posee un atributo o combinación de atributos que se denominan "clave primaria" y que emplea para diferenciar cada instancia de los demás.

Adicionalmente los atributos pueden ser obligatorios u opcionales.

- A los atributos que forman parte de la clave primaria se los identifica anteponiéndoles el signo de número (#).
- A los atributos obligatorios se les antepone el asterisco (*).
- A los atributos opcionales se les antepone un círculo (0).

2.3.7 Generalización y especialización

La generalización es el tipo de interrelación que existe entre un tipo de entidad y los tipos de entidad más específicos que dependen de él. En el mundo real es muy habitual la descomposición de un tipo de entidad, creándose de esta forma una jerarquía de tipos de entidad donde se puede distinguir un **supertipo** del cual dependen varios **subtipos**. La abstracción correspondiente a este tipo de interrelación entre entidades se denomina **ES_UN** ("IS_A", en inglés).

Así, por ejemplo, en la figura 2.9, el tipo de entidad DOCUMENTO es el supertipo de esta jerarquía, y constituye la generalización de los subtipos de entidades LIBRO y ARTÍCULO. Y a su vez estos últimos son una especialización del supertipo DOCUMENTO.

Una de las características más importantes de las jerarquías es la herencia, por la cual, los atributos de un supertipo son heredados por sus subtipos. Así, por ejemplo, en la figura 2.9 vemos que tanto un libro como un artículo son documentos, por lo que los tipos de entidad ARTÍCULO y LIBRO poseerán (heredarán) todos los atributos del tipo de entidad DOCUMENTO.

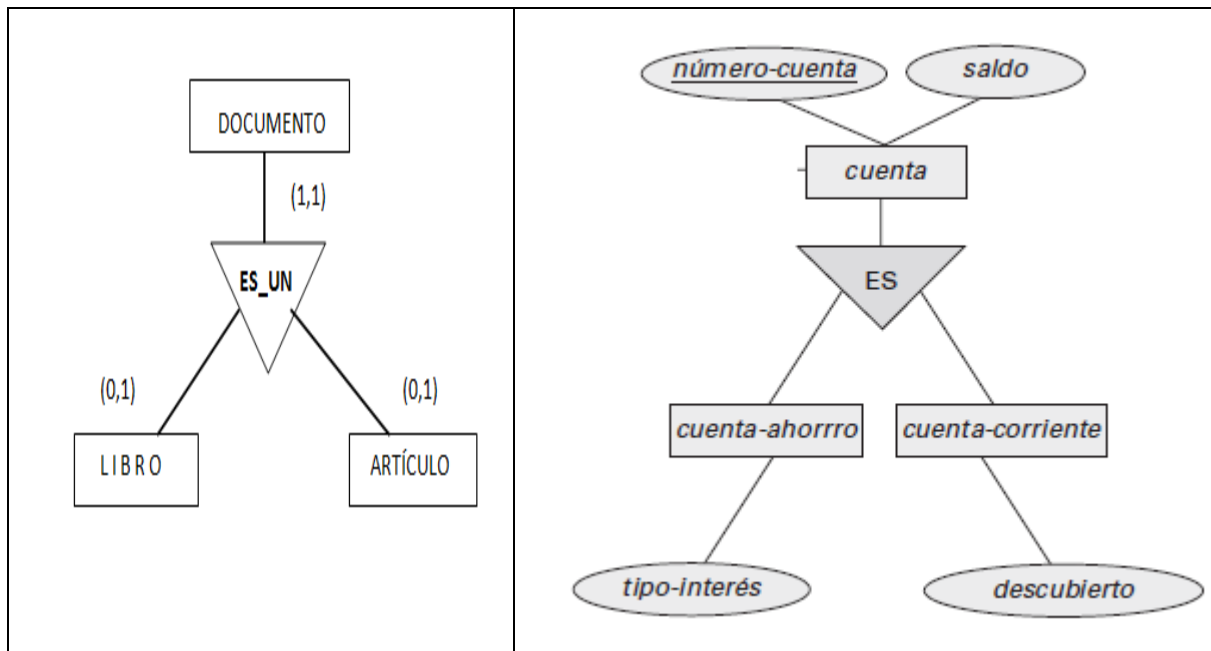


Figura 2.9 Ejemplos de Generalización y Especialización.

En la generalización, los atributos comunes a los subtipos (incluidos los identificadores) se asignan al supertipo, mientras que los atributos específicos se asocian al subtipo correspondiente. Del mismo modo, las interrelaciones que afectan a todos los subtipos se asocian al supertipo, dejándose para los subtipos las interrelaciones específicas en las que el correspondiente subtipo.

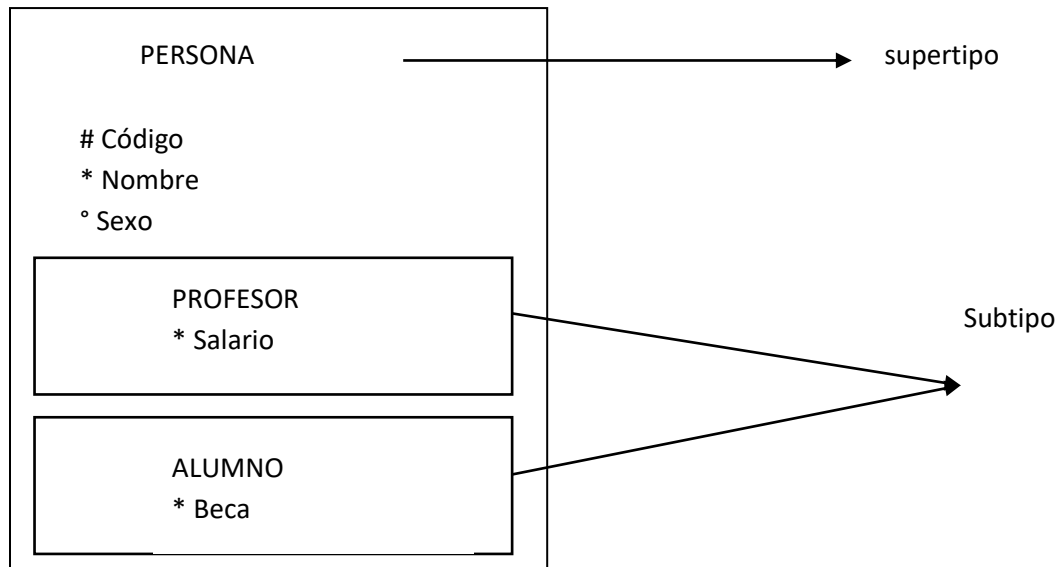
En un diagrama entidad-relación también puede agrupar las entidades en supertipo y en subtipo.

- Los supertipos agrupa a dos más entidades subtipo.
- Los subtipos heredan los atributos de las entidades supertipo.

Supertipo
<div>PROFESOR</div> <div> # Código * Nombre ° Sexo * Salario </div>

Subtipo
<div>ALUMNO</div> <div> # Código * Nombre ° Sexo * Beca </div>

- Cada subtipo puede tener relaciones propias independientes del supertipo.
- Los subtipos se representan como cajas dibujadas dentro de la caja del supertipo.



2.3.8 Agregación

La **agregación** es una abstracción a través de la cual las relaciones se tratan como entidades de nivel más alto. Así, para este ejemplo, se considera el conjunto de relaciones *trabaja-en* (que relaciona los conjuntos de entidades *empleado*, *sucursal* y *trabajo*) como un conjunto de entidades de nivel más alto denominado *trabaja-en*. Tal conjunto de entidades se trata de la misma forma que cualquier otro conjunto de entidades. Se puede crear entonces una relación binaria *dirige* entre *trabaja-en* y *director* para representar quién dirige la tarea. Figura 2.10.

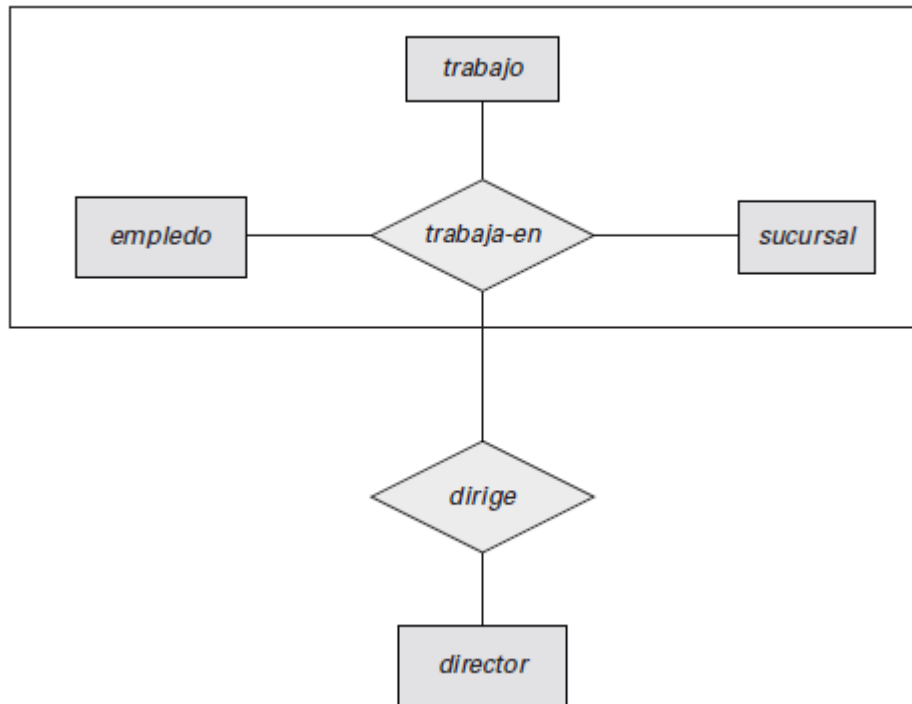


Figura 2.10 Diagrama E-R con agregación.

2.3.9 Ejemplo Diagrama E-R “Punto de Venta”

Una tienda de venta de abarrotes, desea tener una base de datos para facilitar la gestión de la información de sus clientes, facturas “pedidos”, productos, marcas, categorías y empleados.

La tienda cuenta con varios clientes, de los cuales almacena (nombre completo, dirección, RFC, correos electrónicos, números de teléfonos celulares, etc.). Los clientes reciben una o más facturas por cada una de sus compras. En el caso de las facturas la información de los clientes sólo se guardará por una sola ocasión.

Cada nota factura consta de la fecha de compra, folio de la nota, código de barras del producto, nombre del producto, precio de venta, unidades vendidas nombre del empleado “vendedor” que realizó la venta, además un pedido puede tener uno o más productos y un producto puede aparecer en uno o más pedidos. Cada producto pertenece a una sola marca y categoría; pero cada categoría o marca tiene uno o más productos. Además cada producto tiene un precio de compra y unidades en existencia.

Finalmente a los empleados se les almacenará los siguientes datos: nombre completo, dirección, código de empleado, correos electrónicos, números de teléfonos celulares, salario mensual, horario laboral. Figura 2.11

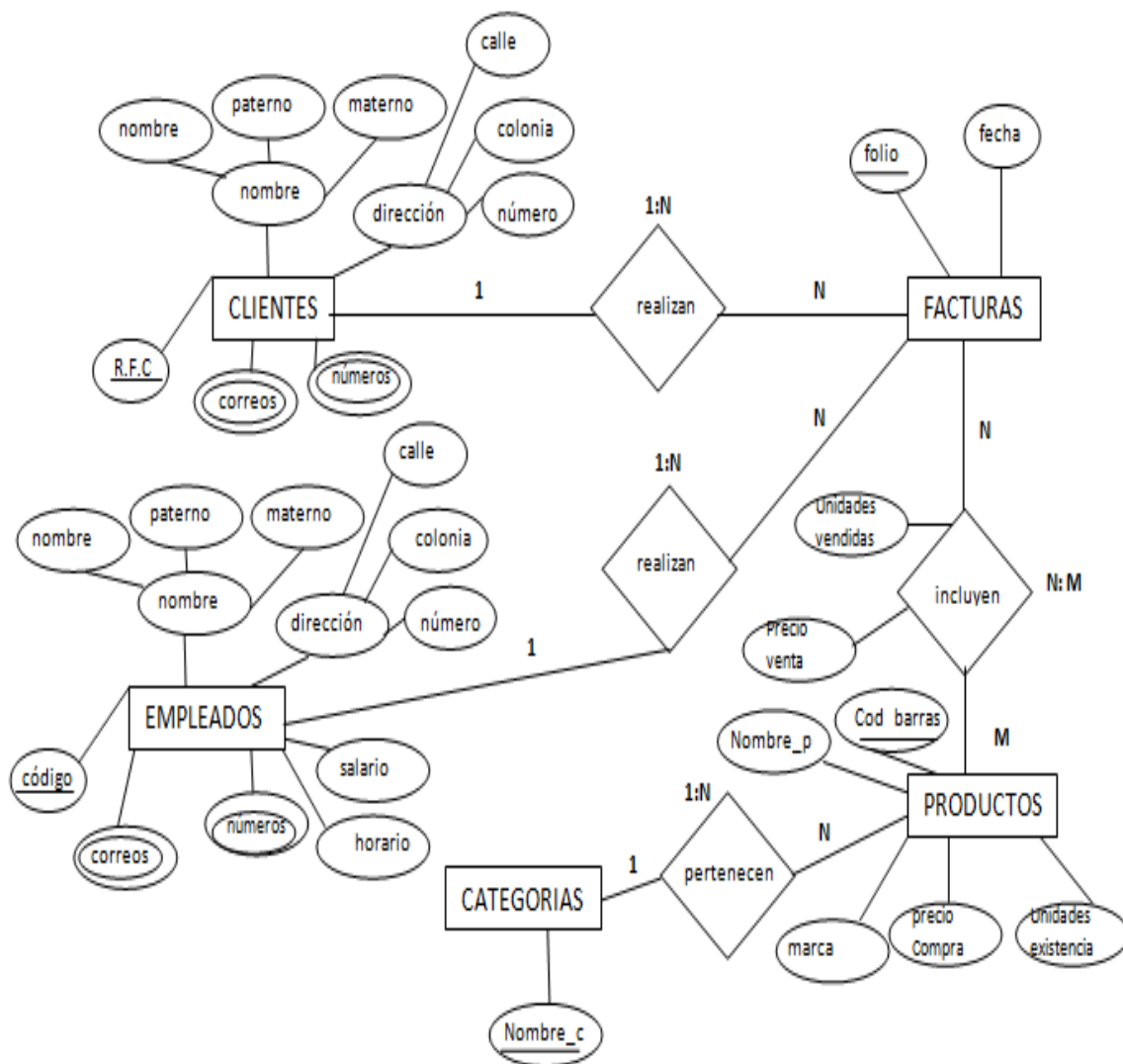


Figura 2.11. Diagrama Entidad-Relación para una tienda de abarrotes pequeña.

2.4 Otros Modelos de Datos (XML)

XML (eXtensible Mark up Language) es un lenguaje de representación de datos que permite etiquetarlos para descubrir su significado. Una de sus características distintivas es que permite crear conjuntos específicos de etiquetas (dialectos de XML) para describir toda clase de datos. Este lenguaje fue diseñado para un ambiente altamente distribuido como lo es Internet, haciendo posible el tener datos autos-descritos que pueden tratarse automáticamente y que son independientes de cualquier plataforma de hardware, ya que todo en XML es texto plano.

Representación de datos en XML

Los componentes básicos de XML son las etiquetas y los elementos. Las etiquetas (tags en inglés) son utilizadas para marcar secciones en los datos, para una aplicación bancaria estos podrían ser por ejemplo cliente, cuenta, etc. Un elemento define una sección de datos que se delimita entre una etiqueta de apertura y otra de cierre. Figura 2.12

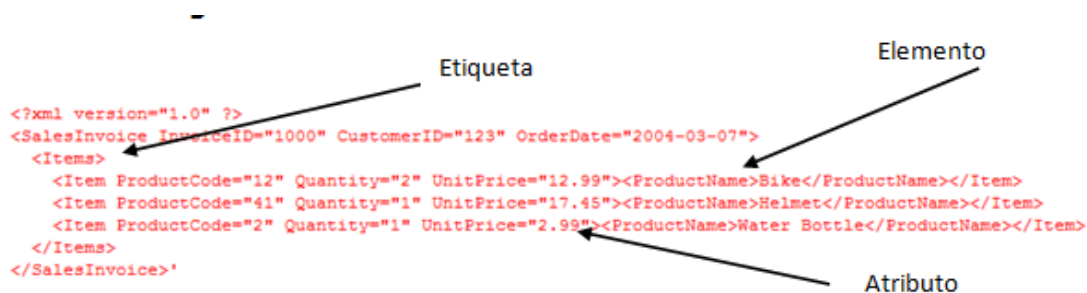


Figura 2.12 Representación de Datos XML

Los elementos pueden anidarse a varios niveles formando una jerarquía elemento/subelemento. Los elementos pueden tener atributos, los cuales se especifican por pares nombre=valor dentro de la etiqueta de inicio de un elemento. Un elemento puede tener varios atributos, pero cada atributo puede ocurrir solo una vez.

Una discusión frecuente es en qué momento se deben usar atributos o subelementos. Por ejemplo, la matrícula de un alumno se puede representar como un atributo:

```
<alumnos matricula="980019335">.....</alumnos>
```

O bien como un sub_elemento:

```
<alumnos>
  <matricula>980019335</matricula>
</alumnos>
```

En este caso, una sugerencia es usar atributos para los identificadores de los elementos y usar sub_elementos para el contenido.

Esquemas XML

El esquema de una base de datos restringe la información que puede ser almacenada y los tipos de datos de los valores almacenados. Los documentos XML no requieren tener un esquema asociado, pero son los esquemas los que hacen posible consultar e interpretar los datos. Para los XML, existen dos formas de especificar esquemas: la Document Type Definition (DTD) y el XML Schema. Figura 2.13.

```
<xsd:schema xmlns:xsd=http://www.w3c.org/2001/XMLSchema>
  <xsd:element name="bank" type="BankType"/>
  <xsd:element name="customer">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="customer-name" type="xsd:string"/>
        <xsd:element name="customer-street" type="xsd:string"/>
        <xsd:element name="customer-city" type="xsd:string"/>
        <xsd:element ref="account" minOccurs="0"
                      maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  ... definición del tipo account

  <xsd:complexType name="BankType">
    <xsd:element ref="customer" minOccurs="0"
                  maxOccurs="unbounded" />
  </xsd:complexType>
</xsd:schema>
```

Figura 2.13 Schema de un XML.

XPath y XQuery son lenguajes de interrogación que ofrecen múltiples funcionalidades. En el sitio web del consorcio W3C se encuentra información detallada de ambos.

3. Modelo Lógico de Bases de Datos

3.1 Introducción

Es una buena justificación para estudiar la teoría que hay tras el modelo relacional, la que da Hernández (1997):

“Muchas disciplinas (y sus metodologías de diseño asociadas) tienen algún tipo de base teórica. Los ingenieros industriales diseñan estructuras utilizando teorías de la física. Los compositores crean sinfonías utilizando conceptos de teoría de la música. La industria del automóvil utiliza teorías de la aerodinámica para diseñar automóviles con menor consumo. La industria aeronáutica utiliza las mismas teorías para diseñar alas de aviones que reduzcan la resistencia al viento.”

Estos ejemplos demuestran que la teoría es muy importante. La ventaja principal de la teoría es que hace que las cosas sean predecibles: nos permite predecir qué ocurrirá si realizamos una determinada acción. Por ejemplo, sabemos que si soltamos una piedra, caerá al suelo.”

Consideremos ahora el ejemplo de una base de datos relacional. Sabemos que si un par de tablas están relacionadas, podemos extraer datos de las dos a la vez, simplemente por el modo en que funciona la teoría de las bases de datos relacionales. Los datos que se saquen de las dos tablas se basarán en los valores coincidentes del campo que ambas tienen en común. Una vez más, nuestras acciones tienen un resultado predecible. El modelo relacional se basa en dos ramas de las matemáticas: la teoría de conjuntos y la lógica de predicados de primer orden. El hecho de que el modelo relacional esté basado en la teoría de las matemáticas es lo que lo hace tan seguro y robusto. Al mismo tiempo, estas ramas de las matemáticas proporcionan los elementos básicos necesarios para crear una base de datos relacional con una buena estructura, y proporcionan las líneas que se utilizan para formular buenas metodologías de diseño. Hay quien ofrece una cierta resistencia a estudiar complicados conceptos matemáticos para tan sólo llevar a cabo una tarea bastante concreta.

El modelo Relacional fue propuesto por el Dr. Edgar F. Codd de IBM en 1970 en el siguiente artículo: **"A Relational Model for Large Shared Data**

Banks, Communications of the ACM, June 1970". Este artículo revolucionó el área de las bases de datos por este trabajo, Edgar Codd obtuvo el premio Turing en 1981 (ACM Turing Award). Figura 3.1

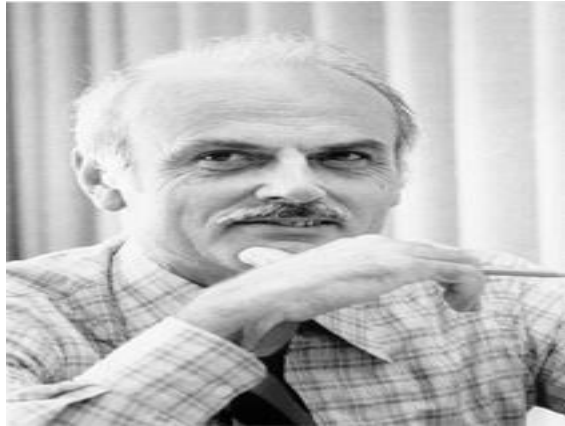


Figura 3.1 Edgar F. Codd.

El modelo relacional es ampliamente usado en los sistemas manejadores de bases de datos actuales. Obtiene su nombre del concepto matemático de *relación matemática*, donde cada entidad es representada por medio de una relación. La relación matemática puede verse como un conjunto de 'valores' diferentes de una entidad dada.

3.2 Objetivos del MR

El documento de Codd propone un modelo de datos basado en la teoría de las relaciones, en donde los datos se estructuran lógicamente en forma de relaciones "tablas", siendo un objetivo fundamental del modelo mantener la independencia de esta estructura lógica respecto al modo de almacenamiento y a otras características de tipo físico.

El trabajo publicado por Codd en ACM presentaba un nuevo modelo de datos que perseguía una serie de objetivos:

- **Independencia física:** el modo en que se almacenan los datos no debe influir en su manipulación lógica y, por tanto, los usuarios acceden a esos datos no han de modificar sus programas por cambios en el almacenamiento físico.

- **Independencia lógica:** Añadir, eliminar o modificar cualquier elemento de la base de datos no debe repercutir en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos (vistas).
- **Flexibilidad:** En el sentido de poder ofrecer a cada usuario los datos de la forma más adecuada a la correspondiente aplicación.
- **Uniformidad:** Las estructuras lógicas de los datos presentan un aspecto uniforme (tablas), lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.
- **Sencillez:** Las características anteriores, así como unos lenguajes de usuario muy sencillos, producen como resultado que el modelo de datos relacionales fáciles de comprender y de utilizar por parte del usuario final.

3.2.1 Estructura del Modelo Relacional

Una base de datos relacional representa a los datos mediante tablas bidimensionales, compuestas de renglones y columnas. Figura 3.2

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel
1	Chai	1	1	10 boxes x 20 b...	18.0000	39	0	10
2	Chang	1	1	24 - 12 oz bottles	19.0000	17	40	25
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.0000	13	70	25
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.0000	53	0	0
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.3500	0	0	0
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.0000	120	0	25
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.0000	15	0	10
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.0000	6	0	0
9	Mishi Kobe Niku	4	6	18 - 4 oz packages	26.0000	0	0	0
10	Ikura	4	8	32 - 4 oz jars	26.0000	0	0	0

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coff...
2	Condiments	Sweet and savo...
3	Confections	Desserts, candie...
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers...
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and b...
8	Seafood	Seaweed and fish

Figura 3.2. Tabla bidimensional.

En la tabla podemos distinguir su nombre, un conjunto de columnas, denominadas **atributos**, que representan propiedades de la tabla y que también están caracterizadas por su nombre, y un conjunto de filas llamadas **tuplas**, que contienen los valores que toma cada uno de los atributos para cada elemento de la relación.

La tabla por sí misma, y no los datos almacenados en ella, es a lo que se llama **relación**. Las relaciones (tablas) poseen un nombre que refleja su contenido

Ejemplo: Relación Products

Table - dbo.Products									
Diagram - LOR...ind.Diagram_0									
Summary									
ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	
1	Chai	1	1	10 boxes x 20 b...	18.0000	39	0	10	
2	Chang	1	1	24 - 12 oz bottles	19.0000	17	40	25	
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.0000	13	70	25	
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.0000	53	0	0	
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.3500	0	0	0	
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.0000	120	0	25	
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.0000	15	0	10	
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.0000	6	0	0	
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.0000	29	0	0	
10	Ikura	4	8	12 - 200 ml jars	31.0000	31	0	0	

Los datos almacenados en una tabla son llamados **ocurrencias o instancias** de la relación.

Las tablas están formadas por una colección de atributos.

Atributo:

- Describe a la relación, con referencia a los datos que almacena.
- Forman las columnas de una relación.
- Tiene un nombre que se relaciona con el tipo de datos que el atributo almacena.

En una tabla se puede distinguir una cabecera que define la estructura de la tabla: es decir, sus atributos con los dominios subyacentes, y un cuerpo que está formado por un conjunto de tuplas que varían en el tiempo. A continuación la

terminología relacional con la que corresponde a las tablas y a los ficheros. Figura 3.3.

RELACIÓN	TABLA	FICHERO
TUPLA	FILA	REGISTRO
ATRIBUTO	COLUMNA	CAMPO
GRADO	NÚMERO DE COLUMNAS	NÚMERO DE CAMPOS
CARDINALIDAD	NÚMERO DE FILAS	NÚMERO DE REGISTROS

Figura 3.3. Relación de terminología en MR.

Propiedades Importantes en las tablas “Relaciones”:

- No se permiten duplicados.
- Número finito de renglones y columnas.
- El orden de los renglones no es relevante.
- Un mismo valor puede aparecer varias veces en una columna.
- El grado (arity) de una tabla es el número de columnas o atributos
Grado debe ser ≥ 0 .
- Cardinalidad de una tabla $|T|$ es el número de tuplas.
- Cada renglón representa una tupla de R.
- Una tupla es un conjunto de atributos asociados y corresponde a un renglón de la tabla.

En el modelo relacional, las relaciones son representadas a través de archivos que se organizan por llaves, la organización física de tales archivos es usando el valor de la llave. El hecho de organizar los archivos a través de las llaves permite que las operaciones de inserción y borrado se realicen a través del valor de la llave. Un esquema relacional se conforma por una o más tablas, las cuáles dinámicamente pueden ser sujetas a inserción de nuevas tuplas, o de la eliminación de algunas de ellas, o bien de la modificación de los componentes de algunas de esas tuplas.

3.2.2 Dominio y Atributo.

Dominio:

- conjunto de valores permitido para un atributo.
- Es una restricción sobre la relación. La restricción es una regla que gobierna los datos que pueden ser almacenados en un atributo de una relación.
- Este conjunto finito de valores es homogéneo porque son todos del mismo tipo y atómicos porque son indivisibles en lo que al modelo se refiere.
- Todo dominio ha de tener un nombre, por el cual nos podemos referir a él, y un tipo de datos.

Un **atributo** es el nombre de una columna de una relación. En el modelo relacional, las relaciones se utilizan para almacenar información sobre los objetos que se representan en la base de datos. Una relación se representa gráficamente como una tabla bidimensional en la que las filas corresponden a registros individuales y las columnas corresponden a los campos o atributos de esos registros. Los atributos pueden aparecer en la relación en cualquier orden.

3.2.3 Claves.

Ya que en una relación no hay tuplas repetidas, éstas se pueden distinguir unas de otras, es decir, se pueden identificar de modo único. La forma de identificarlas es mediante los valores de sus atributos.

Cada relación posee una o varias llaves.

Llave:

- Es el conjunto mínimo de atributos que identifican de manera única a cada tupla en la relación.
- Una llave debe ser única.

- Pueden estar formadas por un solo atributo o por la concatenación de varios atributos.
- Para poder definir las es necesario conocer las reglas bajo las cuales operan los datos en la relación.

Clave Candidata: agrupación de atributos (puede ser sólo un atributo) que identifican sin ambigüedad y de forma unívoca todas las posibles tuplas de una tabla. Como mínimo, una clave debe tener un atributo. Como máximo, los que tenga la tabla. El único modo de identificar las claves candidatas es conociendo el significado real de los atributos, ya que esto permite saber si es posible que aparezcan duplicados. Sólo usando esta información semántica se puede saber con certeza si un conjunto de atributos forman una clave candidata. Figura 3.4

Clave Primaria: se escoge de entre varias claves candidatas. Usualmente se recomienda con el menor número posible de atributos (indexación). No permite nulos en algún atributo de la misma (regla de integridad de entidad).

Claves Alternativas: el resto de las claves candidatas no escogidas como clave primaria.

Clave Ajena: un atributo de una tabla (o agregación de ellos) puede ser la llave primaria de otra tabla. Además sirve como un mecanismo de relación y enlace de información. En una tabla no es obligatoria la existencia de claves ajenas. Atributo simple o compuesto, en una relación, que está definido en el mismo dominio que la llave primaria de otra relación. Una llave foránea no necesita tener el mismo nombre que su llave primaria, sólo contener datos equivalentes.

Table - dbo.Suppliers						
	SupplierID	CompanyName	ContactName	ContactTitle	Address	City
▶	1	Exotic Liquids	Charlotte Cooper	Purchasing Man...	49 Gilbert St.	London
	2	New Orleans Caj...	Shelley Burke	Order Administr...	P.O. Box 78934	New Orleans
	3	Grandma Kelly's ...	Regina Murphy	Sales Represent...	707 Oxford Rd.	Ann Arbor
	4	Tokyo Traders	Yoshi Nagase	Marketing Manager	9-8 Sekimai Mus...	Tokyo
	5	Cooperativa de ...	Antonio del Valle...	Export Administr...	Calle del Rosal 4	Oviedo

Figura 3.4. Llave Primaria

Llave Primaria

3.2.4 Restricciones.

Nulos

Cuando en una tupla un atributo es desconocido, se dice que es *nulo*.

Un nulo no representa el valor cero ni la cadena vacía, éstos son valores que tienen significado. El nulo implica ausencia de información, bien porque al insertar la tupla se desconocía el valor del atributo, o bien porque para dicha tupla el atributo no tiene sentido. Ya que los nulos no son valores, deben tratarse de modo diferente, lo que causa problemas de implementación. De hecho, no todos los SGBD relacionales soportan los nulos.

Regla de Integridad de la llave (Key Constraint):

- Cada tabla debe tener una llave que identifique de manera única cada tupla.

Regla de Integridad de la Entidad (Entity Integrity Constraint):

- La llave Primaria no puede ser Nula.

Regla de Integridad de Referencia (Referential Integrity Constraint):

- Una tupla de una tabla que referencia otra tabla DEBE referirse a una tupla existente en esa tabla.

La integridad referencial es una importante restricción semántica que viene impuesta por el mundo real, siendo el usuario quien la define al escribir el esquema relacional.

Por lo tanto, para cada clave ajena de la base de datos habrá que contestar a tres preguntas:

- **Regla de los nulos:** ¿Tiene sentido que la clave ajena acepte nulos?

•**Regla de borrado:** ¿Qué ocurre si se intenta borrar la tupla referenciada por la clave ajena?

- a) Restringir: no se permite borrar la tupla referenciada.
- b) Propagar: se borra la tupla referenciada y se propaga el borrado a las tuplas que la referencian mediante la clave ajena.
- c) Anular: se borra la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos).

•**Regla de modificación:** ¿Qué ocurre si se intenta modificar el valor de la clave primaria de la tupla referenciada por la clave ajena?

- a) Restringir: no se permite modificar el valor de la clave primaria de la tupla referenciada.
- b) Propagar: se modifica el valor de la clave primaria de la tupla referenciada y se propaga la modificación a las tuplas que la referencian mediante la clave ajena.
- c) Anular: se modifica la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos).

12 Reglas de CODD para los Sistemas Relacionales

1. **Representación de la información:** Toda información en una base de datos relacional debe representarse explícitamente a nivel lógico, y de manera única, por medio de valores en tablas. Podríamos decir que éste es el principio básico del modelo relacional.
2. **Acceso garantizado:** Todo dato debe ser accesible mediante una combinación de un nombre de tabla, un valor de su clave y el nombre de una columna. Es una forma de insistir en la obligatoriedad de la clave primaria.
3. **Tratamiento Sistemático de valores nulos:** Los valores nulos información desconocida o inaplicable, han de ser tratados sistemáticamente por el sistema, el cual ha de ofrecer las facilidades necesarias para su tratamiento.

4. **Catálogo activo en línea basado en el modelo relacional:** la representación de la metainformación (descripción de la base de datos “diccionario de datos”) debe ser igual a la de los otros datos, y su acceso debe poder realizarse por medio del mismo lenguaje relacional que se utiliza para los demás datos; es decir, el modelo de datos para la metainformación debe ser también el relacional.
5. **Sub-lenguaje de datos completo:** Debe existir un lenguaje que permita un completo manejo de la base de datos (definición de datos, definición de vistas, manipulación de datos, restricciones de integridad, autorizaciones y gestión de transacciones).
6. **Actualización de vistas:** Toda vista teóricamente actualizable debe poder ser actualizada por el sistema.
7. **Inserciones, modificaciones y eliminaciones de alto nivel:** Todas las operaciones de manipulación de datos (consulta, inserción, modificación y borrado) deben operar sobre conjunto de filas.
8. **Independencia física de los datos:** El acceso lógico a los datos debe mantenerse incluso cuando cambien los métodos de acceso o la forma de almacenamiento.
9. **Independencia lógica de los datos:** Los programas de aplicación no deben verse afectados por cambios realizados en las tablas que estén permitidos teóricamente y que preserven la información.
10. **Independencia de la integridad:** Las reglas de integridad de una base de datos deben ser definibles por medio del sublenguaje de datos relacional y habrán de almacenarse en el catálogo de la base de datos (metabase), no en los programas de aplicación. Las reglas deberán estar definidas en el catálogo. El SMBD deberá checar automáticamente el que estas reglas se vean cumplidas.
11. **Independencia de la distribución:** Debe existir un sublenguaje de datos que pueda soportar base de datos distribuidas sin alterar los programas de aplicación cuando se distribuyan los datos por primera vez o se redistribuyan éstos posteriormente. Si la BD es distribuida, deberá aparecer a los usuarios como si estuviera centralizada.

- 12. Regla de la no subversión:** Si un SGBD soporta un lenguaje de bajo nivel que permite el acceso fila a fila, éste no puede utilizarse para saltárselas reglas de integridad expresadas por medio del lenguaje de más alto nivel

Ejemplo Esquema Sucursal Empleado

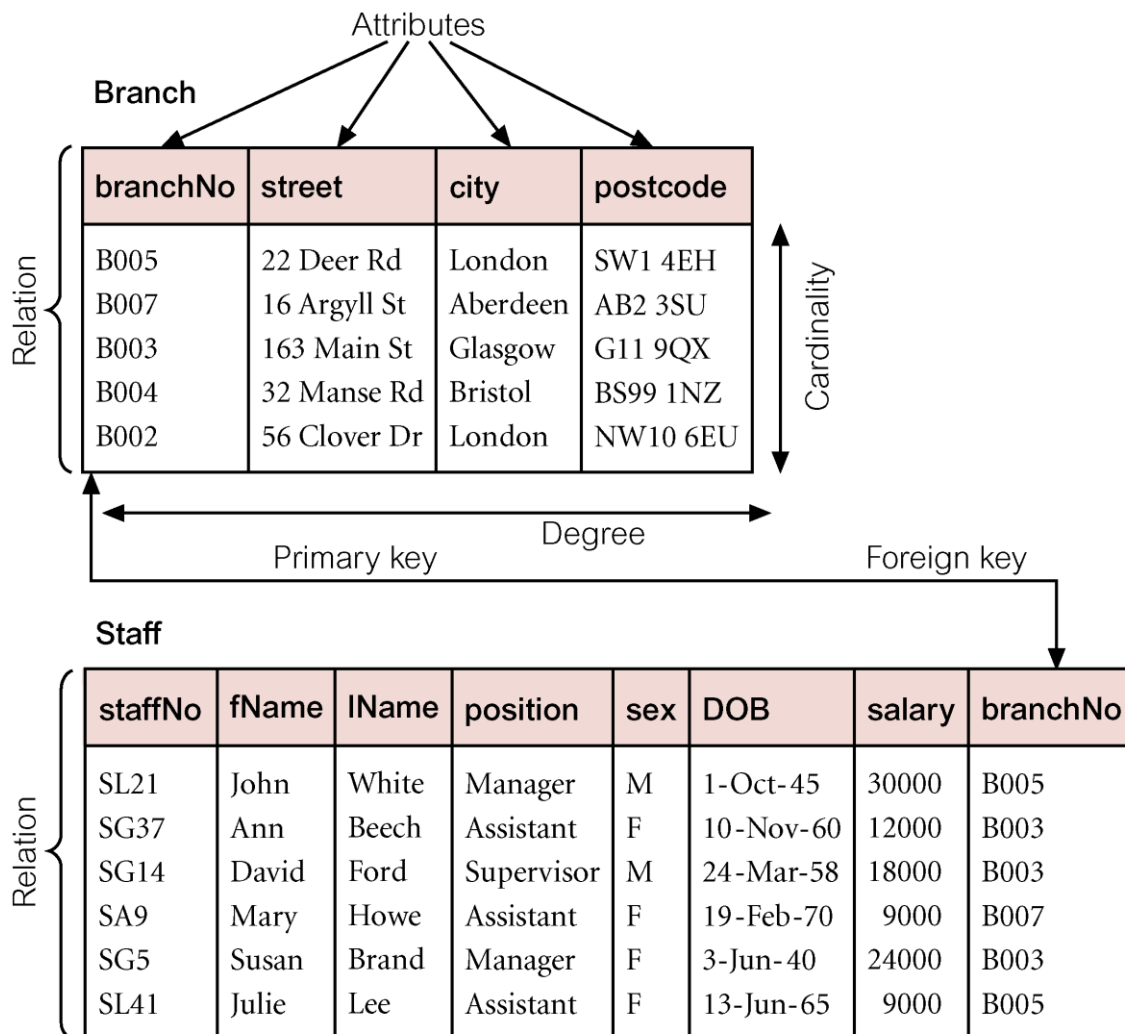


Figura 3.5 Esquema Sucursal Empleado

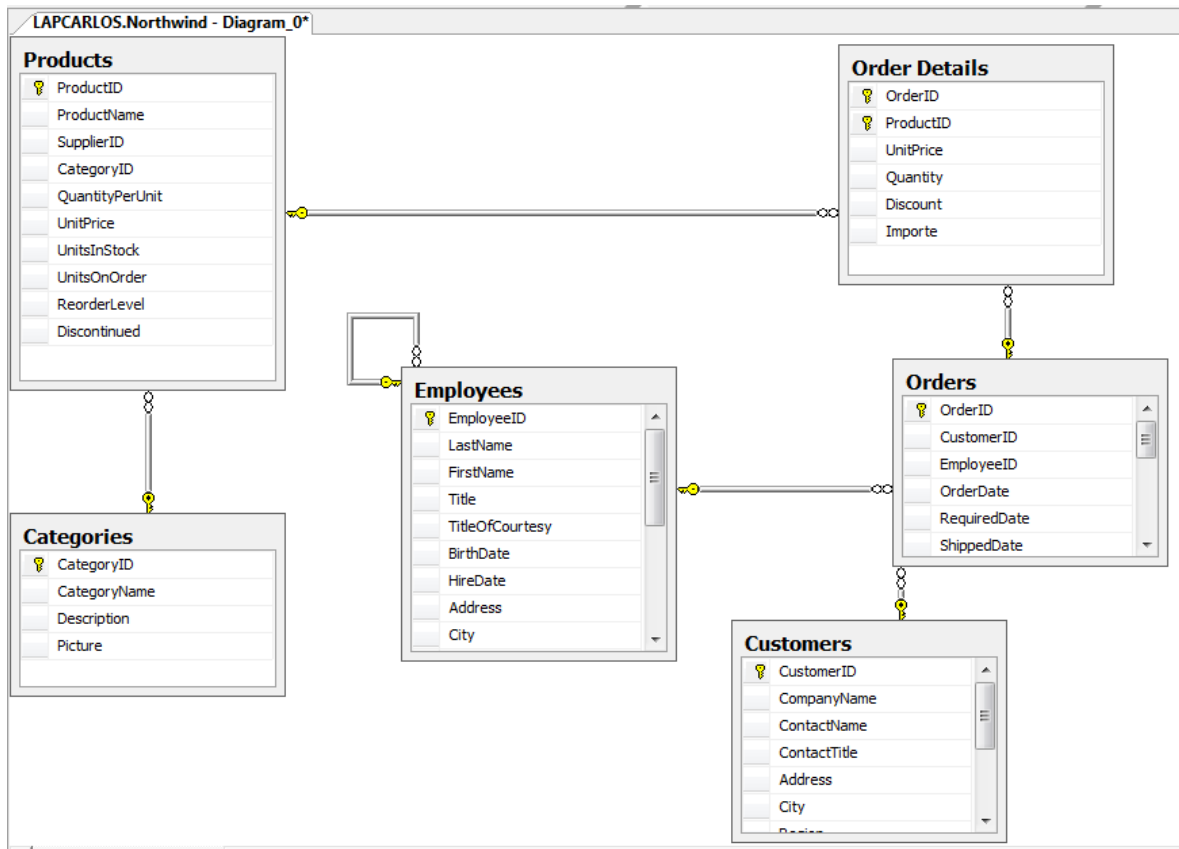


Figura 3.6 Esquema Relacional de Bases de Datos "NorthWind".

Ejercicio

The following tables form part of a database held in a relational DBMS:

Hotel	(<u>hotelNo</u> , hotelName, city)
Room	(<u>roomNo</u> , <u>hotelNo</u> , type, price)
Booking	(<u>hotelNo</u> , <u>guestNo</u> , <u>dateFrom</u> , dateTo, roomNo)
Guest	(<u>guestNo</u> , guestName, guestAddress)

- where Hotel contains hotel details and hotelNo is the primary key;
 Room contains room details for each hotel and (roomNo, hotelNo) forms the primary key;
 Booking contains details of the bookings and (hotelNo, guestNo, dateFrom) forms the primary key;
 and Guest contains guest details and guestNo is the primary key.

3.8 Identify the foreign keys in this schema. Explain how the entity and referential integrity rules apply to these relations.

3.3 Álgebra Relacional

Es un lenguaje FORMAL, PROCEDURAL, para realizar consultas en BD relacionales. Además el álgebra relacional proporciona un método paso a paso para obtener una relación que contenga los datos que constituyen la respuesta al problema planteado.

El álgebra relacional resulta una extensión de la teoría de conjuntos y consta de nueve operaciones fundamentales, cuatro de las cuales son muy similares a las de la teoría matemática. Las operaciones son:

1. Unión.
2. Intersección.
3. Diferencia.
4. Producto Cartesiano.
5. Restricción o selección
6. Proyección.
7. Reunión o Join.
8. División.
9. Asignación.

Sean r y s relaciones con Esquemas R y S , respectivamente, se definen 5 Operaciones básicas:

■ union: $r \cup s = \{t \mid t \in r \vee t \in s\}$

■ Diferencia : $r - s = \{t \mid t \in r \wedge t \notin s\}$

■ producto cartesiano : $r \times s = \{t \mid t = t_r t_s \text{ where } t_r \in r \wedge t_s \in s\}$

■ Selección: $\sigma_p(r)$

■ Proyección: $\pi_A(r)$

Esquema Relacional Ejemplo

Cse_majors(id, name, class)

Eee_majors(id, name, class)

Cse_profs(name, office)

Cse_courses(crsid, crstitle)

Teaches(tname, tcrsid)

Instancia de la BD

Cse_majors

Id	Name	Class
1111	Student1	Fr
2222	Student2	So
3333	Student3	Jr
4444	Student4	Sr
5555	Student5	Gr

Cse_profs

Name	Office
Prof1	Office1
Prof2	Office2

Teaches

tname	tcrsid
Prof1	PR1
Prof1	DB1
Prof2	IA1

Eee_majors

Id	Name	Class
2222	Student2	So
4444	Student4	Sr
6666	Student6	Sr

Cse_courses

Crsid	Crstitle
PR1	Programacion I
DB1	Bases de Datos I
IA1	Sistemas Inteligentes



Unión

$$r \cup s = \{t \mid t \in r \vee t \in s\}$$

- Las relaciones r y s deben ser COMPATIBLES en UNION
- 1) El grado de r y s debe ser el mismo (mismo número de atributos)
- 2) El dominio de atributo i^{th} de r debe ser el mismo dominio del atributo i^{th} de s.

- Se eliminarán las tuplas duplicadas puesto que se trata de un conjunto.

Ejemplo

Query: Obtener la información de los estudiantes cuya especialidad (major) sea Ciencias Computacionales O electrical engineering

Respuesta: cse_majors U eee_majors

Nuevo Schema: ID, NAME, CLASS

cse_majors U eee_majors

ID	NAME	CLASS
1111	Student1	FR
2222	Student2	SO
3333	Student3	JR
4444	Stuent4	SR
5555	Student5	GR
6666	Student6	SR

Diferencia.

$$r - s = \{t \mid t \in r \wedge t \notin s\}$$

Restricción: r y s DEBEN SER compatibles en UNION

Ejemplo

Query: Obtenga la información de los estudiantes con especialidad en Ciencias Computacionales y que no tengan especialidad en Ingeniería eléctrica (Electrical Engineering)

Respuesta: cse_majors – eee_majors

Schema: ID, NAME, CLASS

■ cse_majors – eee_majors

ID	NAME	CLASS
1111	Student1	FR
3333	Student3	JR
5555	Student5	GR

Producto Cartesiano

$r \times s = \{t \mid t = t_r t_s \text{ where } t_r \in r \wedge t_s \in s\}$

■ Sea n_r y n_s el número de tuplas en r y s

■ Hay $n_r * n_s$ tuplas en $r \times s$

■ El esquema de la relación formada por $r \times s$ es la concatenación de los esquemas de r y s , con sus respectivos nombres de atributos.

Query: Exprese el query que obtenga todas las combinaciones posibles de los profesores de ciencias computacionales con los cursos de ciencias computacionales.

Solución: cse_profs x cse_courses

Schema: NAME, OFFICE, CRSID, CRSTITLE

■ cse_profs x cse_courses

NAME	OFFICE	CRSID	CRSTITLE
Prof1	Office1	PR1	Programación I
Prof1	Office1	DB1	Bases de Datos I
Prof1	Office1	IA1	Sistemas Inteligentes
Prof2	Office2	PR1	Programación I
Prof2	Office2	DB1	Bases de Datos I
Prof2	Office2	IA1	Sistemas Inteligentes

Selección

$$\sigma_p(r)$$

Selecciona las tuplas de r que satisfacen el predicado P
 P es un predicado que expresa una condición sobre atributos de r .

Query: Escriba el query que liste la información de los estudiantes con clasificación SR (seniors) y que tengan su especialidad en Ciencias Computacionales

Respuesta: $\sigma_{\text{class}='SR'}(\text{cse_majors})$

Esquema: ID, NAME, CLASS

■ $\text{class}='SR'(\text{cse_majors})$

ID	NAME	CLASS
4444	Student4	SR

Proyección

$$\blacksquare \pi_A(r)$$

Regresa sólo los atributos dados en el conjunto A de la relación r, donde A es un subconjunto de los atributos de r.

- Query: Escriba el query que liste el nombre y el ID de los estudiantes cuya especialidad (major) es Ciencias Computacionales.

Solución: $\pi_{ID, NAME}(cse_majors)$

- Schema: ID, NAME

ID	NAME
1111	Student1
2222	Student2
3333	Student3
4444	Student4
5555	Student5

Operadores Complementarios

- Intersección
- Join
- Natural Join
- División

Intersección

- $r \cap s = r - (r - s)$
- Restricción: r y s deben ser compatibles en unión
- Query: Escriba el query que muestre la información de los estudiantes que tengan como especialidad (major) AMBAS Ciencias Computacionales e Ingeniería Eléctrica.
- Solución: $\text{cse_majors} \cap \text{eee_majors}$
- Schema: ID, NAME, CLASS

■ $\text{cse_majors} \quad \text{eee_majors}$

ID	NAME	CLASS
2222	Student2	SO
4444	Student4	SR

θ -JOIN

- $r \bowtie_{\theta} s = \pi_{\theta} (r \times s)$
 - Query: Escriba el query que muestre la información del profesor y de los cursos que imparte.
 - Solución: $\text{cse_profs} \bowtie_{\text{NAME=TNAME}} \text{teaches}$
- Schema: NAME, OFFICE, TNAME, TCRSID

■ cse_profs \bowtie NAME=TNAME teaches

NAME	OFFICE	TNAME	TCRSID
Prof1	Office1	Prof1	PR1
Prof1	Office1	Prof1	DB1
Prof2	Office2	Prof2	IA1

El θ -join más común es el operador de "igualdad" (=) y la operación se llama equi-Join. El ejemplo anterior es un equi-join.

En general la condición expresada en θ puede ser cualquier operación relacional (<, >, <=, >=, <>, =).

JOIN NATURAL

$$r \bowtie s = (R \cup S) (r \bowtie_{r.A_1=s.A_1} \bigwedge \dots \bigwedge_{r.A_n=s.A_n} S)$$

donde $R \cup S = \{A_1, \dots, A_n\}$

Es decir, $r \bowtie s$ es la relación que resulta de la proyección de $R \cup S$ de un θ -join donde el predicado requiere que $r.A=s.A$ para cada atributo A in $R \cap S$.

Considerando los siguientes esquemas

teaches (name, crsid), cse_profs(name, office),
cse_courses(crsid, crstitle)

Query: Escriba el query que obtenga la información de cursos y profesores que los imparten.

Answer: cse_profs \bowtie (cse_courses \bowtie teaches)

Esquema del resultado: NAME, OFFICE, CRSID, CRSTITLE

■ cse_profs \bowtie (cse_courses \bowtie teaches)

NAME	OFFICE	CRSID	CRSTITLE
Prof1	Office1	PR1	Programación I
Prof1	Office1	DB1	Bases de Datos I
Prof2	Office2	IA1	Sistemas Inteligentes

Outer Join

Natural Join

Solamente las tuplas R de que coinciden con las tuplas de S (and viceversa) aparecen en el resultado. Tuplas que no coinciden, son eliminadas del resultado.

Las tuplas con NULL en los atributos del JOIN son también eliminadas.

Left outer Join

Todas las tuplas de la primera relación r aparecen en el resultado de la operación r outer-join s . Si los valores campos del join coinciden, se colocan los valores correspondientes, y si no coinciden, se pone NULL en los atributos de s

Right outerjoin (r right outer-join s)

Todas las tuplas de la segunda relación s aparecen en el resultado

Full outerjoin (r full outer-join s)

Todas las tuplas r y s aparecen en el resultado

Ejemplo left outer join

Escriba el query que obtenga todos los nommbres de los empleados y el nombre del departamento que administran (SI es que administran un depto.)

Temp \leftarrow (employee left outer join (ssn=mgrssn) Department)

Result $\leftarrow \pi$ fname, minit, lastname,dname (Temp)

FNAME	MINIT	LNAME	DNAME
John	B	Smith	Null
Franklin	T	Wong	Research
Alicia	J	Zelaya	Null
Jennifer	S	Wallace	Administration
Ramesh	K	Narayan	Null
Joyce	A	English	Null
Ahmad	V	Jabbar	Null
James	E	Borg	Headquarters

Álgebra (Operadores Básicos)

- El álgebra relacional es un lenguaje de consultas procedural que aplica operadores específicos a una relación o relaciones con el objeto de obtener datos. Algebra relacional es un lenguaje formal basado en teoría de conjuntos.
- Hay cinco operadores básicos
 - Selección (select). La operación de selección $\sigma_{\theta}(r)$ regresa las tuplas de la relación r que satisfacen la condición de selección θ
 - Proyección (project). La operación de proyección $\pi_A(r)$ regresa los atributos de una relación r de acuerdo a la lista de atributos deba por el subíndice A
 - Unión (union). La operación binaria de union $r \cup s$ regresa la unión de las tuplas de las relaciones r y s . Las relaciones r y s deben de ser compatibles.
 - Diferencia (difference). La operación binaria de diferencia $r-s$ regresa las tuplas de la relación r que no aparecen en la relación s
 - Producto Cartesiano (cartesian product). La operación binaria de producto cartesiano $q \times r$ regresa todas las posibles combinaciones de las tuplas de las relaciones q y r

Álgebra (Operadores Adicionales)

- Operadores Adicionales
 - Intersección (intersection)
 - La operación binaria de intersección regresa aquellas tuplas aparecen en las dos relaciones que aparecen como operandos.
 - Combinación(join)
 - La operación de join esta definida como el producto cartesiano entre los operandos (relaciones) seguido de una selección.
 - Natural join
 - Una operación de natural join es una abreviación para una operación de join donde la condición de selección se asume como una conjunción de igualdades tal que los valores de atributos con el mismo nombre en ambos operandos son iguales.
 - División
 - Esta es una de las operaciones más complicadas del algebra relacional. Las consultas típicas que requieren de esta operación típicamente encuentran los valores en el primer operando que se relacionan con todos valores en el segundo operando.

3.4 Mapeo de los objetos del Modelo Conceptual al Modelo Relacional.

El paso de un esquema en el modelo E/R al relacional está basado en los tres principios siguientes:

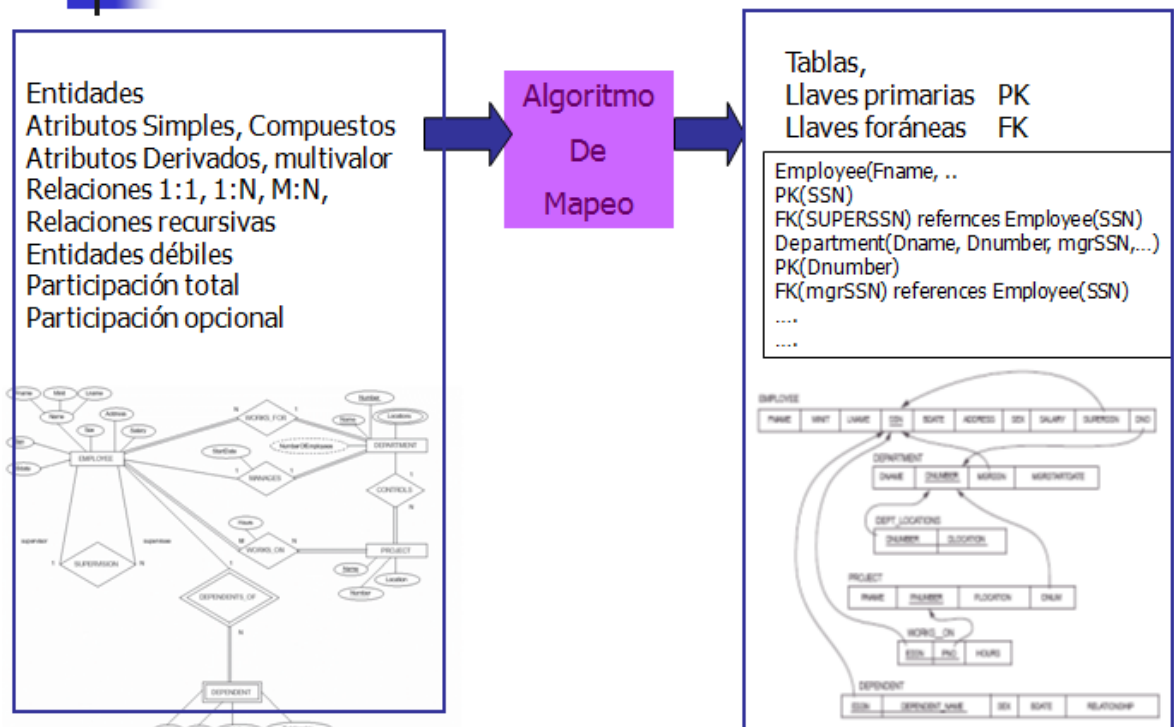
- ✓ Todo tipo de entidad se convierte en una relación.
- ✓ Todo tipo de interrelación N:M se transforma en una relación.
- ✓ Todo tipo de interrelación 1:N se traduce en el fenómeno de propagación de clave o bien se crea una nueva relación. Ver figura 3.7

El enfoque ER representa un diseño conceptual que representa una situación real. Aquí consideramos el mapeo de entidades y relaciones de un diagrama ER a las relaciones (tablas) del modelo de datos relacional. Aunque el mapeo es flexible de alguna forma, se definen varias heurísticas o reglas mapeo.



Input:ERD

Output: Esquema Relacional

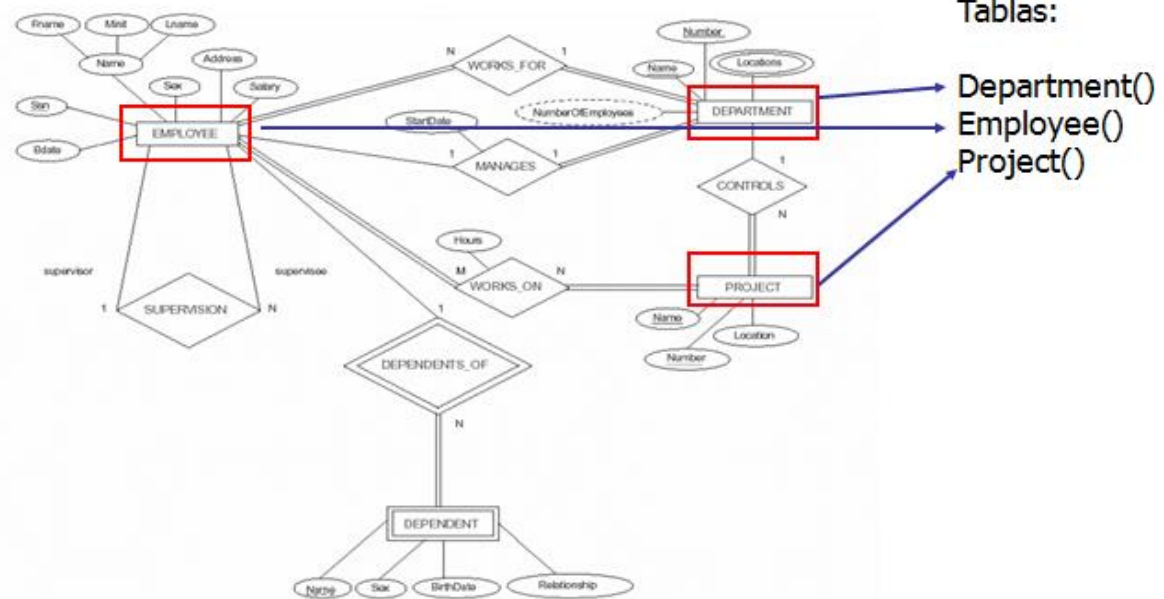


Reglas de Mapeo

- R1 Mapeo de Entidades.
- R2 Mapeo de Atributos Simples y Compuestos.
- R3 Mapeo de relaciones 1:1.
- R4 Mapeo de relaciones 1:N.
- R5 Mapeo de relaciones M:N.
- R6 Mapeo de atributos multivalor.
- R7 Mapeo de Entidades débiles.
- R8 Mapeo de relaciones recursivas.
- R9 Mapeo de atributos derivados.

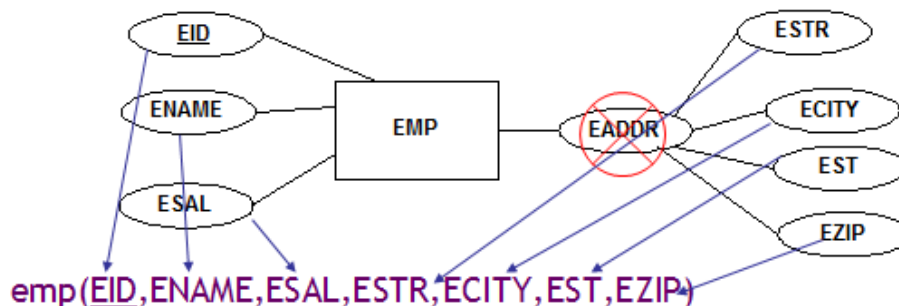
REGLA 1. MAPEO DE ENTIDADES

- Por cada entidad no débil, crear una tabla



Regla 2. Atributos Simples y Compuestos

- Cada Atributo Simple y cada nodo hoja de los atributos compuestos es una **columna en R**.
- Atributos de la tabla
 - Los atributos simples de la entidad y los componentes simples de atributos compuestos.
- Llave Primaria (Primary Key)
 - Llave primaria de la entidad.

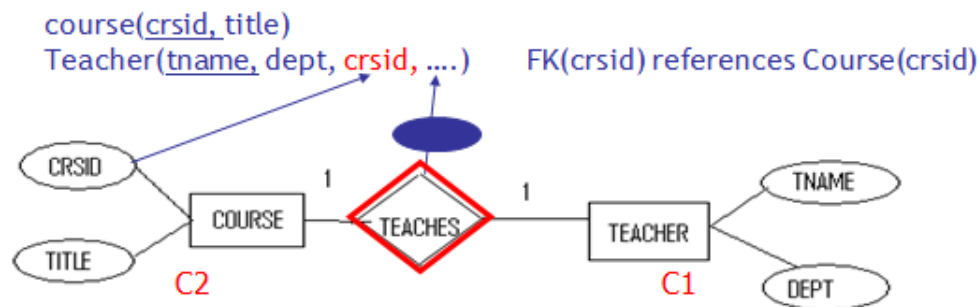


Regla 3. Mapeo de Relaciones 1:1

Relación de 1:1 entre clases C1 y C2 , seleccionar una de las clases (C1) e incluir la llave de C2 en C1 como una llave foránea. Incluir los atributos simples de la asociación entre ambas clases en C1

Agregar a una de las tablas seleccionada (C1):

1. Atributos correspondientes a la llave primaria de la otra entidad involucrada en la relación (relationship). En este caso C2 o course
2. Si existen Atributos de la relación (relationship) C1:C2 se ponen en la tabla correspondiente a C1, TEACHER en este ejemplo.



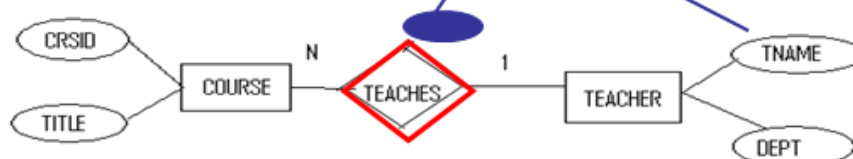
Nota: Si es posible, seleccionar a la entidad que tiene participación total (total participation) en la relación (relationship)

Regla 4. Mapeo de Relaciones 1: N

- Agregar a la relación o tabla (relation) de la entidad en el lado muchos (many) del tipo de relación (relationship) lo siguiente:

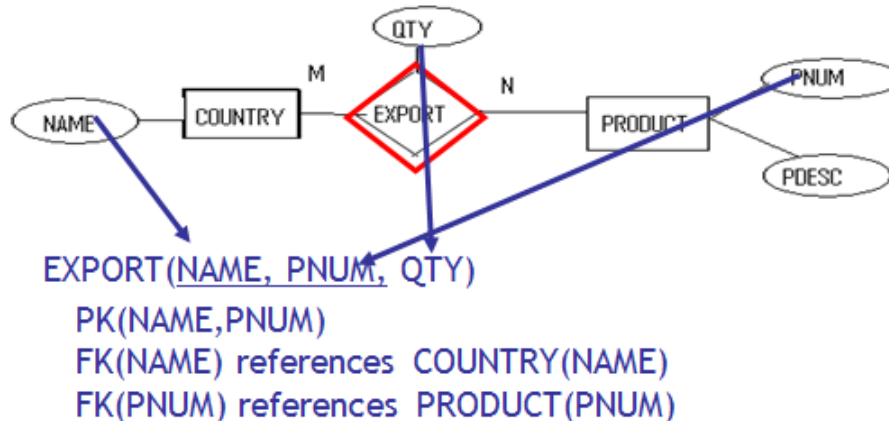
- Los Atributos que forman la **llave primaria** de la entidad en el lado uno del tipo de relación(relationship)
- Si existen atributos de la relacion (relationship), se ponen en la tabla que tiene N

- course(CRSID, TITLE, **TNAME**,)
FK(TNAME) references teacher(TNAME)
- teacher(TNAME, DEPT)



Regla 5. Mapeo de Relaciones N: M

- **R5. Relaciones N:M** entre clases C1 y C2, crear una relación R para representar la relación entre clases. Incluir las llaves de C1 y C2 y todos los atributos simples de la relación entre las clases.
- **Llave Primaria**
 - Combinación de los atributos que forman las llaves primarias de las entidades involucradas en la relación (relationship).



Regla 6. Mapeo de atributos multivalor.

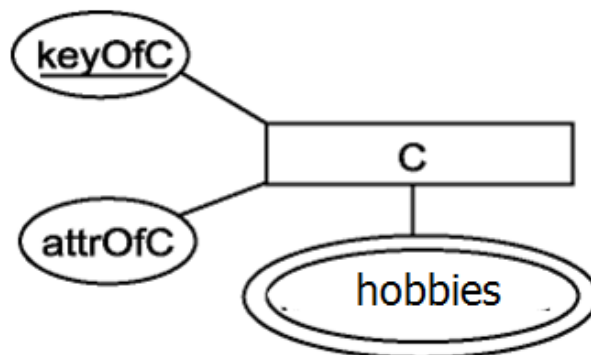
Crear una tabla para el atributo multivalor que incluya la llave de la entidad:

C(keyOfC, attrOfC)

hobbies(keyOfC, hobby)

FK(keyOfC) references C(keyOfC)

La llave de la relación *Hobbies* es una llave compuesta que consiste de la llave de la clase y el atributo multivalor. Notar que la restricción de integridad referencial se debe mantener



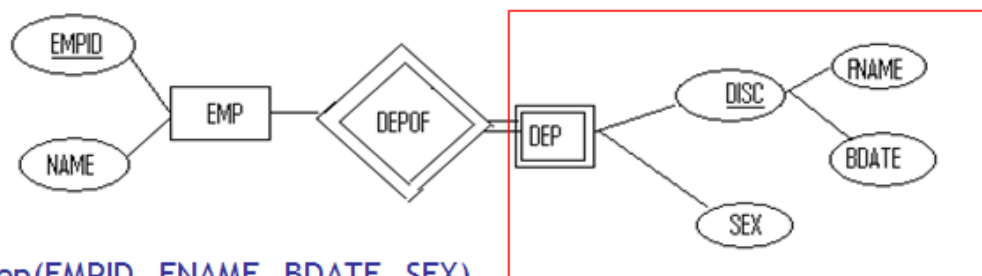
Regla 7. Mapeo de Entidades débiles.

- **Atributos.**

Atributos simples y componentes simples de atributos compuestos de la misma entidad y los atributos correspondientes a la llave primaria de la entidad dueña (strong entity).

- **Llave Primaria (Primary Key)**

Combinación de los atributos de la llave primaria de la entidad dueña y el discriminador (llave parcial [partial key]) de la entidad débil.



dep(EMPID, FNAME, BDATE, SEX)

Regla 8. Mapeo de relaciones recursivas.

- R8 Utilizar las reglas R3, R4 y R5 de acuerdo a la cardinalidad 1:1, 1:N o M:N. La llave foránea hace referencia a la misma tabla.

- Tabla(llave, attr2, attr3, llave foránea, attrel)
PK(llave)
FK(llave foránea) references Tabla(llave)

Regla 9. Atributos Derivados.

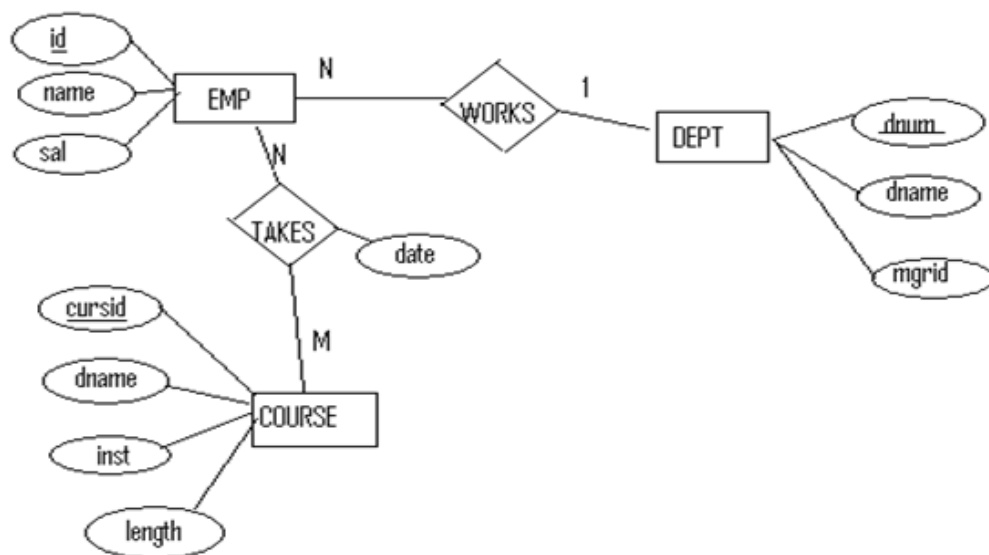
Los atributos derivados no se representan en el modelo relacional. Se deben codificar dentro de la aplicación para calcularse.

Reglas de Mapeo

- R1 Mapeo de Entidades
 - Crear tabla
- R2 Mapeo de Atributos Simples y Compuestos
 - Columnas en la tabla
- R3 Mapeo de relaciones 1:1
 - Llave foránea en una de las tablas
- R4 Mapeo de relaciones 1:N
 - Llave foránea del lado N
- R5 Mapeo de relaciones M:N
 - Crear una tabla con 2 llaves foráneas
- R6 Mapeo de atributos multivalor
 - Una nueva tabla con PK y atributo multivalor
- R7 Mapeo de Entidades débiles
 - Una nueva tabla mas la llave primaria de entidad dueña
- R8 Mapeo de relaciones recursivas
 - Llaves foráneas que referencian a la misma tabla
- R9 Mapeo de atributos derivados
 - No se representan en el modelo

Ejercicio de Mapeo

- Mapea el diagrama ER para cursos de entrenamiento de empleados al modelo relacional.



Solución

```
EMP( id, name, sal, dnum)
    PK(id)
    FK(dnum) references DEP(dnum)
DEP( dnum, dname, mgr)
    PK(dnum)
COURSE (cursid,dname, inst, length)
    PK(cursid)
TAKES (id, cursid)
    PK (id, cursid)
    FK(id) references EMP(id)
    FK(cursid) references COURSE(cursid)
```

3.5 Teoría de la Normalización.

El diseño de una base de datos relacional se puede realizar mediante la metodología de Diagrama E-R, a fin de obtener un esquema conceptual; en una segunda fase, se transforma dicho esquema al modelo relacional mediante las correspondientes reglas de transformación. Aunque esta primera aproximación produce un esquema estructurado y con poca redundancia, el éxito se fundamenta en la experiencia del que realiza este modelado; sin embargo, siempre es conveniente aplicar un conjunto de reglas, conocidas como teoría de la normalización, que nos permite asegurar que un esquema relacional cumple con ciertas propiedades.

La normalización es el conjunto de reglas a aplicar para reducir en forma sistemática una relación a un conjunto de relaciones más pequeñas que son equivalentes y más deseables que la relación original. Además, la normalización es una técnica que nos permite obtener estructuras de datos más eficientes.

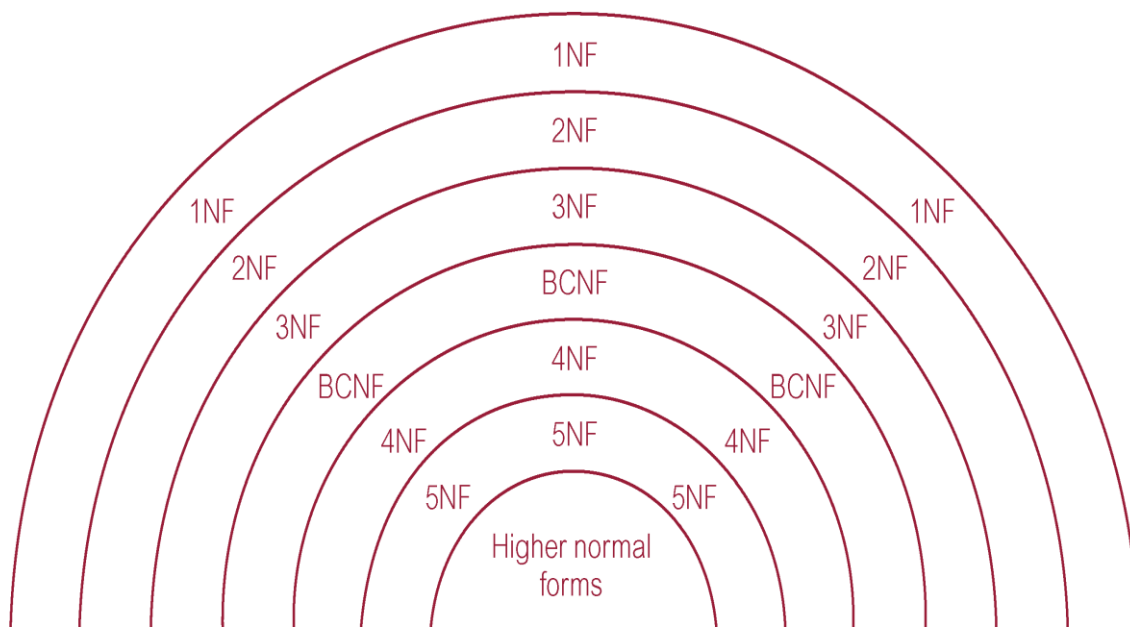
La normalización está basada en el concepto de formas normales cada forma normal tiene un conjunto de reglas que deben ser verificadas (1FN, 2FN, 3FN).

Estas formas normales están anidadas, es decir que para que una relación esté en 3FN debe haber pasado por 2FN y está por 1FN.

Los objetivos principales de la normalización son:

- 1) Eliminar cierto tipo de redundancia.
- 2) Evitar anomalías de actualización e inserción.
- 3) Simplificar las restricciones de integridad.
- 4) Producir un diseño que sea fácil de entender intuitivamente y constituya una buena base para el futuro.

Relaciones entre las formas Normales



Conceptos Usados en la normalización

3.6 Dependencias Funcionales

Concepto principal asociado con la normalización.

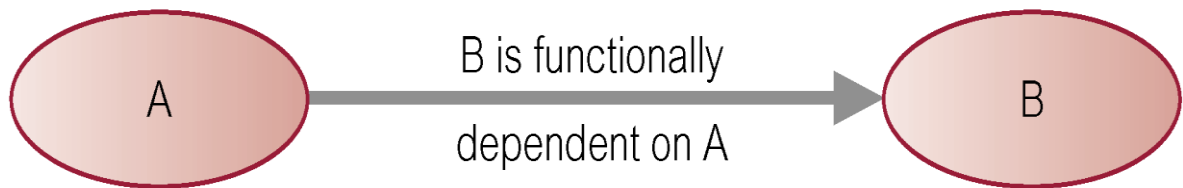
Dependencia Funcional (Functional Dependency)

Describe una relación entre atributos

A, B son atributos de la relación R,

$A \rightarrow B$ se lee: A determina los valores de B

B es funcionalmente dependiente A si cada valor de A en R está asociado con exactamente un valor de B en R.

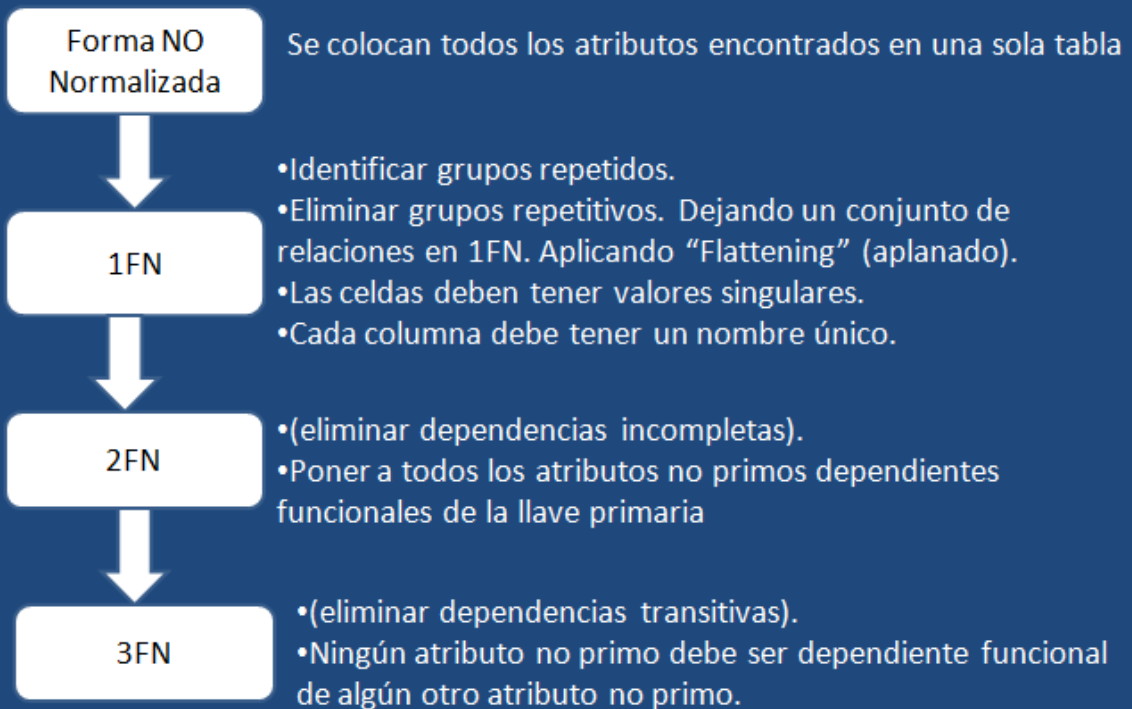


Claves o llaves

Claves o llaves. Es el atributo que le da la diferencia a cada tabla, este atributo hace que no tengamos tuplas o filas repetidas.

MATRICULA	NOMBRE	APELLIDO PATERNO
200812345	ARTURO	RIOS
200712345	PEDRO	LOPEZ
200323445	ADRIANA	AGUILAR

Pasos del Proceso de Normalización



PRIMERA FORMA NORMAL (1FN)

DEFINICIÓN:

"Una relación está en 1FN si cumple la propiedad de que sus dominios no tengan elementos que, a su vez, sean conjuntos".

La esencia de la 1FN, es que un registro no incluye ningún grupo repetitivo.

EJEMPLO DE PRIMERA FORMA NORMAL (1FN)

CASO: diseño de la base de datos para la automatización del control de los pedidos de productos.

Se cuenta con el siguiente documento para solicitar los productos:

FECHA:	01/06/2004	PEDIDO NRO.:	123456	
		PROVEEDOR NRO.:	75621	
		NOMBRE PROVEEDOR:	J. PEREZ	
		DIRECCION	LINCE	
DESEAMOS ENVIEN:				
Número Producto	Descripción	Precio Unitario	Cantidad	Subtotal
969715	Televisor	600	1	600
439124	Antena	20	10	200
439126	Espiga	10	10	100
IMPORTE TOTAL:				900

Se presenta un formulario de pedidos de productos a proveedores, el siguiente paso es llevar a la forma normal 1, donde se va a listar todos los atributos encontrados en el formulario y se va a seleccionar un atributo clave.

Primera Forma Normal (1FN)

Paso 1

✓ Se efectúa un listado de atributos:

NumPed : Número del pedido

FchPed : Fecha en que se realiza el pedido

NumProv : Número del proveedor

NomProv : Nombre del proveedor

DirProv : Dirección del proveedor

NumProd : Número del producto

DescProd : Descripción del producto

PreUniProd : Precio unitario del producto

CantPed : Cantidad de unidades del producto que se solicita

SubTtlProd : Monto a pagar por concepto de ese producto

MtoTtlPed : Monto a pagar por todo el pedido

✓ Indicando la clave.

Una vez que escogemos el atributo NumPed como el atributo clave por ser único. Procedemos a identificar grupos repetitivos.

Primera Forma Normal (1FN)

Paso 2

- ✓ Se verifica si existen grupos repetidos:


NumPed :Número del pedido
FchPed :Fecha en que se realiza el pedido
NumProv :Número del proveedor
NomProv :Nombre del proveedor
DirProv :Dirección del proveedor

NumProd :Número del producto
DescProd :Descripción del producto
PreUniProd :Precio unitario del producto
CantPed :Cantidad de unidades del producto que se solicita
SubTtlProd :Monto a pagar por concepto de ese producto

MtoTtlPed :Monto a pagar por todo el pedido

- ✓ Si no existen la relación está en 1FN, de lo contrario:

Grupo
Repetitivo.
Líneas de
productos
pedidos



Cómo podemos notar en la figura anterior se encuentran grupos repetitivos, por lo tanto no están en primera forma normal, la cual dice que debemos remover grupos repetitivos.

Primera Forma Normal (1FN)

Paso 3

Se eliminan los grupos repetidos, creándose dos relaciones:

- ✓ Una relación para los campos que sean únicos; es decir, se dejan en la **relación original** sólo los atributos que no son repetitivos:

PEDIDO (**NumPed**, FchPed, NumProv, NomProv,
DirProv, MtoTtlPed)

- ✓ Una relación para los grupos repetitivos; es decir, se extraen en una **nueva relación** los atributos repetitivos, además de la clave primaria de la relación original:

PED-PROD (**NumPed**, **NumProd**, DescProd,
PreUniProd, CantPed, SubTtlProd)

Notemos que en la primera relación se encuentran atributos no repetitivos, y tiene un solo atributo clave, sin embargo en la segunda relación donde se encuentran los grupos repetitivos la clave primaria está compuesta por dos atributos NumPed y NumProd

Primera Forma Normal (1FN)

Paso 3

- ✓ Una relación para los campos que sean únicos; es decir, se dejan en la **relación original** sólo los atributos que no son repetitivos:

Relación Original					
PEDIDO					
NumPed	FchPed	NumProv	NomProv	DirProv	MtoTtlPed
123456	01/06/2004	75621	J. PEREZ	LINCE	900

- ✓ Una relación para los grupos repetitivos; es decir, se extraen en una **nueva relación** los atributos repetitivos, además de la clave primaria de la relación original:

Nueva Relación					
PED-PROD					
NumPed	NumProd	DescProd	PreUniProd	CantPed	SubTtlProd
123456	969715	Televisor	600	1	600
123456	439124	Antena	20	10	200
123456	439126	Espiga	10	10	100

Nuestro ejercicio ya se encuentra en primera forma normal, pero presenta problemas en las operaciones de inserción, eliminación y actualización.

Primera Forma Normal (1FN)

Consideraciones

1. **Creación:** La información sobre un nuevo producto no se puede insertar, si no hay un pedido que lo incluya.
2. **Eliminación:** Supresión de una línea de pedido que sea la única que pida un producto, implica perder la información del producto.
3. **Modificación:** Por cada línea de pedido en la que se solicite determinado producto se tiene una ocurrencia en PED-PROD, que repite la información sobre éste. Si cambia algún atributo del producto, entonces es necesario hacer muchas actualizaciones.

Entonces para resolver este problema será necesario aplicar las siguientes fases de normalización (2FN, 3FN) a este modelo para eliminar los problemas que presenta.

Para Continuar debemos entender el concepto de Dependencia Funcional Completa: esto quiere decir que cada atributo no llave debe depender completamente de la llave clave, no parcialmente.

Por ejemplo tenemos la relación:

DEFINICION PREVIAS

Dependencia Funcional Completa:

~~Ejemplo en la relación PROV_PROD:~~

CANT es funcional y completamente dependiente de S,P
(S=producto), P=pedido)

PROV_PROD		
PNRO	SNRO	CANT
P2	S1	2
P2	S2	4
P2	S3	4
P3	S3	4
P3	S5	2
P4	S2	2
P4	S4	3
P4	S5	4

Así por lo tanto la segunda forma nos dice:

Segunda Forma Normal (2FN)

DEFINICION

Una relación R se dice que está en 2FN si además de estar en 1FN, cualquiera de sus atributos no primarios (aquel que no forma parte de la clave) tienen una dependencia funcional completa con la clave primaria de R.

Es decir, todos sus atributos dependen de la clave completa y no sólo de una parte de ésta

Entonces, este segundo paso se aplica sólo a relaciones con claves compuestas!!!

Por lo tanto de las dos relaciones que tenemos aún sólo la segunda tiene una llave compuesta:

De las dos relaciones que tenemos :

PEDIDO (NumPed, FchPed, NumProv, NomProv,
DirProv, MtoTtlPed)

PED-PROD (NumPed, NumProd, DescProd,
PreUniProd, CantPed, SubTtlProd)

**Una relación que esté en 1FN y que no tenga
claves compuestas, está ya en 2FN**

Continuando con la segunda forma normal vemos claramente que los atributos CantPed y SubTtlProd dependen funcional y completamente de NumPed y NumProd, sin embargo:

Segunda Forma Normal (2FN)

Consideramos la relación de clave compuesta:

PED-PROD (NumPed, NumProd, DescProd,
PreUniProd, CantPed, SubTtlProd)

- Esta relación **NO** está en 2FN, pues **DescProd** y **PreUniProd** no dependen funcional y completamente de la clave (NumPed, NumProd).

Por lo tanto se crea una nueva relación para los atributos que dependan de cada parte de la clave:

Segunda Forma Normal (2FN)

Se crea una relación para todos los atributos que dependen funcional y completamente de la clave:

PED-PROD (NumPed, NumProd, CantPed, SubTtlProd)

Se crea otra relación para los atributos que dependan de cada parte (subconjunto) de la clave.

PRODUCTO (NumProd, DescProd, PreUniProd)

Entonces ahora tenemos las siguientes relaciones:

Segunda Forma Normal (2FN)

PEDIDO (NumPed, FchPed, NumProv, NomProv, DirProv, MtoTtlPed)

PED-PROD (NumPed, NumProd, CantPed, SubTtlProd)

PRODUCTO (NumProd, DescProd, PreUniProd)

Sin embargo tenemos problemas con la relación **PEDIDO**:

Segunda Forma Normal (2FN)

Pero aún tenemos problemas en este caso que son similares a los vistos, pero con la relación **PEDIDO** y, específicamente, cuando se trata de insertar, eliminar o modificar la información de proveedores.

SOLUCION: 3FN!!!!

Por lo tanto ahora debemos aplicar la 3FN:

Tercera Forma Normal (3FN)

DEFINICION

Una relación R está en 3FN si está en 2FN y si, y sólo si, los atributos no claves son independientes de cualquier otro atributo no clave primaria.

Esto es lo mismo que decir que se deben eliminar las dependencias entre atributos no claves. Es decir que los atributos deben depender sólo de la clave y de ningún otro atributo de la relación.

Aplicando la 3FN a la relación PEDIDO nos queda:

Tercera Forma Normal (3FN)

Paso 1

- ✓ Se crea una relación para los atributos no claves que no dependen transitivamente de la clave primaria

Esta relación:



Quedaría así:

PEDIDO (NumPed, FchPed, NumProv, MtoTtlPed)

Luego creamos una nueva relación llamada Proveedor

Tercera Forma Normal (3FN)

Paso 2

- ✓ Y una relación para los atributos no claves que dependen transitivamente de la clave primaria a través de otro atributo o conjunto de atributos no clave primaria (que no son parte de la clave primaria.) La clave primaria de la relación así formada será el atributo o conjunto de atributos a través de los cuales existe la dependencia transitiva:

De esta relación:

PEDIDO (NumPed, FchPed, NumProv, NomProv, DirProv, MtoTtlPed)

Surgiría:

PROVEEDOR (NumProv, NomProv, DirProv)

Finalmente obtenemos las siguientes relaciones:

Tercera Forma Normal (3FN)

Se obtienen las siguientes relaciones:

PEDIDO (NumPed, FchPed, NumProv, MtoTtlPed)

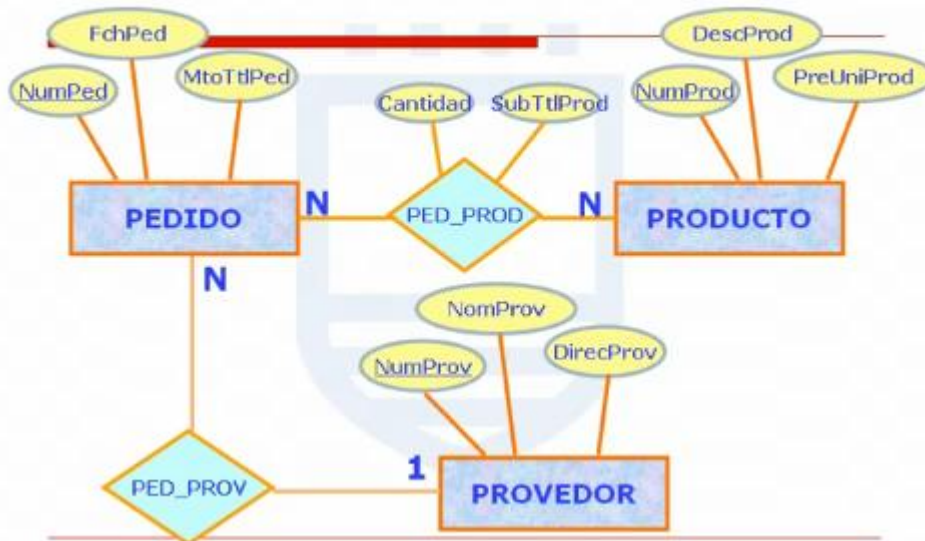
PED-PROD (NumPed, NumProd, CantPed, SubTtlProd)

PRODUCTO (NumProd, DescProd, PreUniProd)

PROVEEDOR (NumProv, NomProv, DirProv)

Luego lo pasamos a DER

DER del documento normalizado



APÉNDICE

EJERCICIOS DEL CAPITULO DOS.

Instrucciones: Lea con atención el siguiente problema, y luego modele E-R

Ejercicio 1. PARQUES NATURALES

Una comunidad autónoma (CA) puede tener varios parques naturales. En toda comunidad autónoma existe un y sólo un organismo responsable de los parques. Un parque puede estar compartido por más de una comunidad autónoma.

Un parque natural se identifica por un nombre, fue declarado en una fecha, se compone de varias áreas identificadas por un nombre y caracterizadas por una determinada extensión. Por motivos de eficiencia se desea favorecer las consultas referentes al número de parques existentes en cada comunidad y la superficie total declarada parque natural en cada CA.

En cada área forzosamente residen especies que pueden ser de tres tipos: vegetales, animales y minerales. Cada especie tiene una denominación científica, una denominación vulgar y un número inventariado de individuos por área. De las especies vegetales se desea saber si tienen floración y en qué período se produce ésta; de los animales se desea saber su tipo de alimentación (herbívora, carnívora u omnívora) y su período de celo anual; de las minerales se desea saber si se trata de cristales o de rocas.

Ejercicio 2. EMPRESA DE PROYECTOS INFORMÁTICOS

Una empresa de consultoría desea crear una base de datos para facilitar la gestión de los proyectos informáticos que desarrolla para sus empresas clientes. Los requisitos se muestran a continuación:

La empresa desarrolla proyectos de los que almacena su código, nombre, cliente para el que se desarrolla el proyecto, una breve descripción, presupuesto, número de horas totales estimadas, fecha inicio y fecha fin. Cada proyecto se compone de una serie de fases identificadas por un número de secuencia en cada proyecto. Cada fase se caracteriza, además, por su nombre, fecha de comienzo, fecha de fin y estado en que se encuentra (en curso o finalizada).

Los empleados de la empresa (código, DNI, nombre, dirección, titulación y años de experiencia) están asignados a los proyectos que desarrolla la empresa.

Interesa almacenar a los empleados que son jefes de proyecto junto con su dedicación total en horas prevista a cada proyecto así como el coste de su participación en euros, teniendo en cuenta que cada proyecto lo lidera un único jefe de proyecto. En cuanto a los informáticos que participan en los proyectos, se quiere conocer los que son programadores, así como el número de horas totales previstas dedicadas en cada proyecto y el coste en euros. Además se almacenará los lenguajes de programación que dominan cada uno de los programadores.

En cada fase de un proyecto se generan una serie de productos (software) sobre los que se quiere guardar información. Cada producto previsto para una fase tiene un código, un nombre, una descripción, una versión, si está finalizado o no y tiene como participantes a varios programadores, pero solo un jefe de proyecto.

Ejercicio 3. PUBLICACIONES

El departamento de Informática de una Universidad, necesita una base de datos para almacenar información concerniente a los proyectos de investigación tanto actuales como pasados en los que trabajan los profesores y así poder llevar una gestión más eficiente. La información que se desea almacenar corresponde a los siguientes supuestos:

En el departamento los profesores participan en proyectos de investigación caracterizados por un código de referencia único, por un nombre, un acrónimo, un presupuesto total, el programa de I+D que lo financia, una fecha de inicio y una fecha de finalización y una breve descripción de los objetivos del proyecto.

En los proyectos trabajan profesores del departamento durante un periodo de tiempo, es decir, una fecha de inicio y una fecha de fin, pudiendo ocurrir un profesor trabaje en el mismo proyecto en varias épocas (f_{ini} , f_{fin}) diferentes. Un profesor se identifica por su nombre y apellidos y se caracteriza por su despacho y teléfono y puede trabajar en varios proyectos simultáneamente y en un proyecto de investigación trabajan varios profesores.

Los proyectos de investigación producen una serie de publicaciones sobre las que también interesa guardar información. Una publicación se caracteriza por un número en secuencia dentro de cada proyecto de investigación y se guardará el título y los profesores que la han escrito; las publicaciones son de dos tipos, publicaciones en congresos y publicaciones en revista; de las primeras se almacenará el nombre del congreso, su tipo (nacional o internacional), la fecha de

inicio y de fin, el lugar de celebración, país y la editorial que ha publicado las actas del congreso (si es que se han publicado); de las publicaciones en revista interesa saber el nombre de la revista, la editorial, el volumen, el número y las páginas de inicio y fin.

Ejercicio 4. ADMINISTRACIÓN DE FINCAS

Una firma de abogados dedicada a la administración de fincas desea tener una base de datos para facilitar la gestión de la información de sus clientes, es decir las distintas comunidades de vecinos que administra. La información que debe contener la BD concierne a los aspectos que se describen a continuación.

La firma tiene varios abogados y cada uno de ellos ejerce de administrador de una o más comunidades de vecinos, por lo que cobra a cada una de ellas unos honorarios anuales. Una comunidad de vecinos es gestionada por un único administrador (Nombre, DNI y No de Colegiado). Las funciones de un administrador, sobre las que en este caso interesa guardar información, consisten en llevar la contabilidad de la comunidad, gestionando los recibos que pagan los vecinos mensualmente, así como los pagos a distintas compañías que proporcionan algún servicio a la comunidad (limpieza, ascensores, seguridad, luz, etc.).

De las empresas que tienen contratadas las distintas comunidades de vecinos (por ejemplo, Iberdrola, Unión Fenosa, OTIS, etc.) se guarda su nombre CIF, dirección, teléfono y una persona de contacto. Además interesa tener estas compañías agrupadas en distintos sectores (luz, seguridad, ascensores, etc.).

De cada comunidad de vecinos gestionada por la firma de abogados interesa almacenar un código identificador, su nombre, calle, código postal y población. Cada comunidad consta de una serie de propiedades que pueden ser de tres tipos (vivienda particular, local comercial y oficina). Cada propiedad se caracteriza por un número de portal, planta y letra, un nombre y apellidos del propietario con su dirección completa. Y un teléfono de contacto.

Si la vivienda es particular se guardará el número de habitaciones de que dispone; si es un local comercial se almacenará el tipo de comercio que se desarrolla en él y el horario (en caso de que esté en uso); si es una oficina se guardará la actividad a la que se destina.

Bibliografía

- Pressman Roger S., "Ingeniería del Software un enfoque práctico", sexta edición, Mc Graw Hill, 2006.
- Adoración de Miguel Mario Piattini., "Fundamentos y modelos de Bases de Datos", cuarta edición, Alfaomega RA-MA, 2003.
- Dolores Cuadra, Elena Castro, Ana Ma. Iglesias., "Desarrollo de Bases de datos: casos prácticos desde el análisis a la implementación", segunda edición, alfaomega RA-MA, 2008.
- Date C.J., "Introducción a los sistemas de bases de datos", Vol 4, Addison Wesley, 2003.
- Sommerville Ian., "Ingeniería del Software", novena edición, Addison Wesley, 2008.
- Baca Urbina Gabriel., "Evaluación de Proyectos", quinta edición, Mc Graw Hill, 2005.
- Silverschatz Abraham., "Fundamentos de Bases de Datos", sexta edición Mc Graw Hill, 2006.