

## CAPITULO I

### INTRODUCCION A LOS SISTEMAS DE BASES DE DATOS

#### Base de Datos

“Una base de datos es un conjunto de elementos de datos que se describe a sí mismo, con relaciones entre esos elementos, que presenta una interfaz uniforme de servicio” (14)

“Una base de datos está constituida por cierto conjunto de datos persistentes utilizado por los sistemas de aplicaciones de una empresa determinada.”(9)

De las referencias bibliográficas anteriores se puede concluir que:

Una Base de Datos se define como una colección o depósito de datos homogéneos entre los que existen relaciones lógicas, integrados con redundancia controlada y con una estructura de diseño que refleje las restricciones existentes en el mundo real, con el objetivo de satisfacer los requerimientos de información de múltiples usuarios de una empresa u organización, teniendo en cuenta que todos los procedimientos de inserción, actualización y recuperación comunes y bien determinados deben conservar la integridad, seguridad, consistencia y confidencialidad del conjunto de los datos.

Además una base de datos se define como un conjunto de datos almacenados de manera ordenada y sistemática en algún medio de almacenamiento de datos, desde el cual pueden ser recuperados, para que sirvan como soporte en la toma de decisiones.

Las bases de datos proporcionan la infraestructura requerida para los sistemas de apoyo a la toma de decisiones y para los sistemas de información estratégicos, ya que estos sistemas explotan la información

contenida en las bases de datos de la organización para apoyar el proceso de toma de decisiones o para lograr ventajas competitivas.

Por este motivo es importante conocer la forma en que están estructuradas las bases de datos y su manejo.

## Bases de Datos Automatizadas

Una base de datos automatizada es un conjunto de datos almacenados de manera ordenada y sistemática en algún soporte informático de almacenamiento de datos, desde el cual pueden ser recuperados, para que sirvan como soporte en la toma de decisiones.

Las bases de datos automatizadas pueden ser de 2 tipos:

Bases de datos automatizadas en Archivos Convencionales.

Bases de datos Relacionales.

(Ver Figura 1.1.)

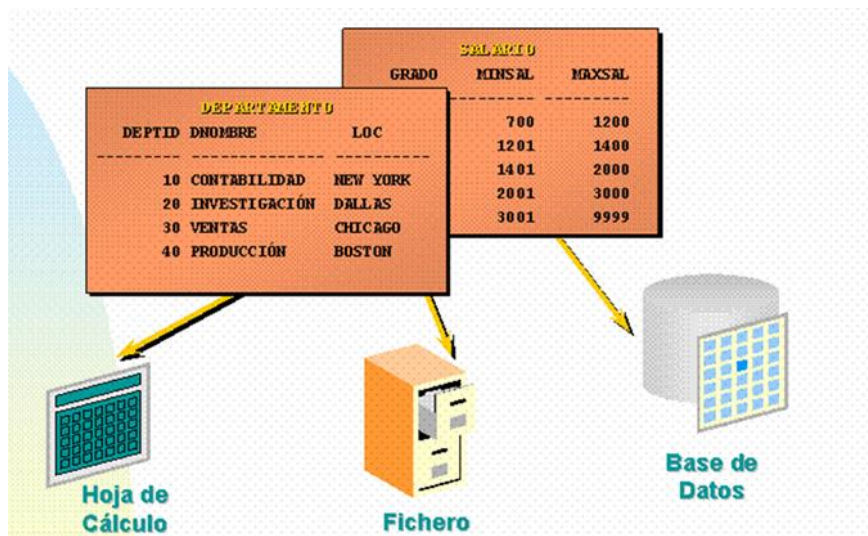


Figura 1.1. Tipos de Bases de Datos Automatizadas

## **Bases de Datos en Archivos Convencionales**

Las formas en las cuales pueden organizarse son archivos secuenciales o archivos directos. En los archivos secuenciales los registros están almacenados en una secuencia que depende de algún criterio definido. Por ejemplo, pueden almacenarse los registros de los empleados de la empresa de manera secuencial de acuerdo al departamento al que pertenecen o de acuerdo a su antigüedad. Si se desea consultar o modificar información, también es necesario buscar uno por uno en los registros hasta encontrarla.

Los archivos directos permiten acceder directamente un registro de información sin tener que buscar uno a uno por todos los registros del archivo, utilizando una llave de acceso dentro del archivo.

## **Bases de Datos Relacionales**

Una base de datos relacional es un conjunto, colección o depósito de datos almacenados en un soporte informático de acceso directo, en donde se especifican las relaciones entre los datos, de manera que la base de datos sea el fiel reflejo del mundo real.

Dada la importancia que tienen en el mundo real las interrelaciones entre los datos, es imprescindible que la base de datos sea capaz de almacenar éstas interrelaciones, al igual que hace con otros elementos (como las entidades y atributos), siendo ésta una diferencia esencial respecto a los ficheros donde no se almacenan las interrelaciones.

La redundancia de los datos debe ser controlada, de forma que no existan duplicidades perjudiciales ni innecesarias, y que las redundancias físicas, convenientes muchas veces a fin de responder a objetivos de eficiencia, sean tratadas por el mismo sistema, de modo que no puedan producirse incoherencias. Por tanto, un dato se actualizará lógicamente por el usuario de forma única, y el sistema se preocupará de cambiar físicamente todos

aquellos campos en los que el dato estuviese repetido, en caso de existir redundancia física.

### **Desventajas de las Bases de Datos en Archivos Convencionales**

- Muchos datos son usados repetidamente en múltiples aplicaciones.
- El sistema de base de datos convencionales requiere los mismos datos para ser guardados en múltiples archivos, lo que produce la redundancia de los datos.
- La redundancia de datos produce muchos problemas especialmente con la integridad de los datos.
- La inconsistencia de los datos ocurre con más frecuencia, cuando los mismos datos son almacenados en más de un lugar durante las operaciones de ingreso, actualización y eliminación.
- El sistema de archivos convencionales provee poca capacidad de compartir los datos, en este sistema los archivos de datos son implementados en unidades separadas haciendo dificultoso el compartimiento de los datos eliminando la utilización de múltiples aplicaciones.
- Un sistema de múltiples archivos con redundancia de datos hace más dificultoso el manejo y control de los datos.
- Si un administrador de datos quiere realizar algún cambio en la organización de los datos podría llegar a tener muchas dificultades.

### **Ventajas de una Base de Datos Relacional.**

Una Base de Datos Relacional:

- Provee de poderosas herramientas de manipulación a través de una gran variedad de comandos SQL (Structured Query Language), para sus operaciones.

- Reduce la redundancia de Datos. Los datos pueden ser almacenados y relacionados mediante la utilización de una gran variedad de constructores.
- Limitando la redundancia de los datos se puede ahorrar significativamente la cantidad de espacio en disco requerido.
- Las reglas de integridad de los datos pueden ser reforzadas en los datos contenidos en la Base de Datos Relacional.
- Puede soportar el uso de muchas personas y muchas diferentes aplicaciones.
- Los datos guardados en una Base de Datos Relacional pueden tener más que un uso y pueden ser compartidos por varios usuarios a la vez.
- Puede ser fácilmente modificada y acondicionada para crecer con nuevos requerimientos de información.
- Provee facilidad para su diseño, flexibilidad en cambios al momento de su diseño e independencia de la aplicación en la que se vaya a desarrollar.

### **Ventajas en el uso de Bases de Datos.**

- ✓ Globalización de la información. Permite a los diferentes usuarios considerar la información como un recurso corporativo que carece de dueños específicos.
- ✓ Eliminación de información redundante y duplicada
- ✓ Eliminación de información inconsistente. Si el sistema esta desarrollado a través de archivos convencionales por ejemplo, una cancelación deberá operarse tanto en el archivo de facturas del Sistema de Control de Cobranza como en el archivo de facturas del Sistema de Comisiones, y en el resto de archivos por medio de código.
- ✓ Permite compartir información. Varios sistemas o usuarios pueden utilizar una misma entidad.
- ✓ Permite mantener la integridad en la información. Solo se almacena la información correcta.

- ✓ Independencia de datos. La independencia de datos implica un divorcio entre programas y datos; es decir, se pueden hacer cambios a la información que contiene la base de datos o tener acceso a la base de datos de diferente manera, sin hacer cambios en las aplicaciones o en los programas.

## **Sistema de Bases de Datos Relacionales**

Un sistema de Base de Datos Relacionales puede satisfacer muchos importantes objetivos:

- Puede servir efectivamente en las diferentes funciones de una Empresa.
- Puede proveer información consistente y precisa.
- La cantidad de redundancia de datos puede ser minimizada.
- Son aprovechadas las utilidades de respaldo y recuperación de datos.
- La seguridad en los datos pueden ser aplicados y reforzados.
- Los programas de aplicación pueden ser desarrollados, cambiados y mantenidos rápido, económicamente y con menos personal experto.
- La organización física de los datos puede ser fácilmente establecida.
- El control y manejo centralizado de los datos es posible.

## **Sistemas de Gestión de Bases de Datos (SGBD)**

“Un SGBD (Sistema de Gestión de Bases de datos) es un conjunto de programas que va a permitir insertar, modificar, borrar y buscar eficazmente datos específicos entre un volumen masivo de información compartida por todos los usuarios de la base; pero también es una herramienta que va a permitir ordenar, buscar, reordenar y convertir datos.” (10)

El Sistema de Gestión de Bases de Datos (SGBD) es un conjunto de programas, procedimientos y lenguajes que actúa como un intermediario entre los usuarios y los datos proporcionando las herramientas necesarias y suficientes para realizar todo tipo de operaciones en una Base de Datos.

## Funciones del Sistema de Gestión de Bases de Datos

“Un Sistema de Gestión de Bases de Datos (SGBD) proporciona el método de organización necesaria para el almacenamiento y recuperación flexibles de grandes cantidades de datos.” (14)

“Las funciones mínimas de un SGBD son: la Definición de Datos, y la Manipulación de Datos.” (9)

El Sistema de Gestión de Bases de Datos (SGBD) es un conjunto de programas, procedimientos y lenguajes que proporcionan a los usuarios las herramientas necesarias y suficientes para crear, manipular, controlar y operar con una base de datos relacional. Por tanto, el SGBD actúa como un intermediario entre los usuarios y los datos. Debe cumplir una serie de funciones como descripción de los datos, de manera que debe permitir definir los registros, sus campos, sus relaciones de autorización, etc. Debe manipular los datos permitiendo a los usuarios insertar, suprimir, modificar y consultar datos de la base de datos y por último, debe permitir usar la base de datos, dando un interfaz adecuado a cada tipo de usuario.

El sistema manejador de bases de datos es la porción más importante del software de un sistema de base de datos. Un SGBD es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de alguna tarea específica.

En adición se puede decir que las funciones principales de un SGBD son:

- Crear y organizar la Base de Datos.
- Manejar sentencias de Definición de Datos.
- Establecer y mantener las trayectorias de acceso a la base de datos de tal forma que los datos puedan ser accedidos rápidamente.
- Manejar los datos de acuerdo a las peticiones de los usuarios.

- Registrar el uso de las bases de datos.
- Manejar sentencias de manipulación de datos.
- Facilitar el respaldo y recuperación de los datos.
- Controlar la concurrencia en la interacción entre los usuarios concurrentes para no afectar la consistencia de los datos.
- Permitir la creación y el uso de interfaces adecuadas de acceso a los datos de la base.

### **Niveles de Abstracción en un SGBD según ANSI SPARC**

El SGBD es el software encargado de realizar el ocultamiento de la información y crear las visiones de los datos para cada usuario. En las bases de datos aparece un nuevo nivel de abstracción que se ha denominado de diversas maneras: nivel conceptual, estructura lógico global, esquema, etc. Esta estructura intermedia pretende una representación global de los datos que se interponga entre las estructuras lógica y física y que sea independiente, tanto del equipo como de cada usuario en particular.

ANSI/SPARC es un grupo de normalización creado en 1969 para estudiar el impacto de los S.G.B.D. en los sistemas de información y cuyos resultados, publicados en 1975 propusieron el uso de tres niveles de descripción de datos.

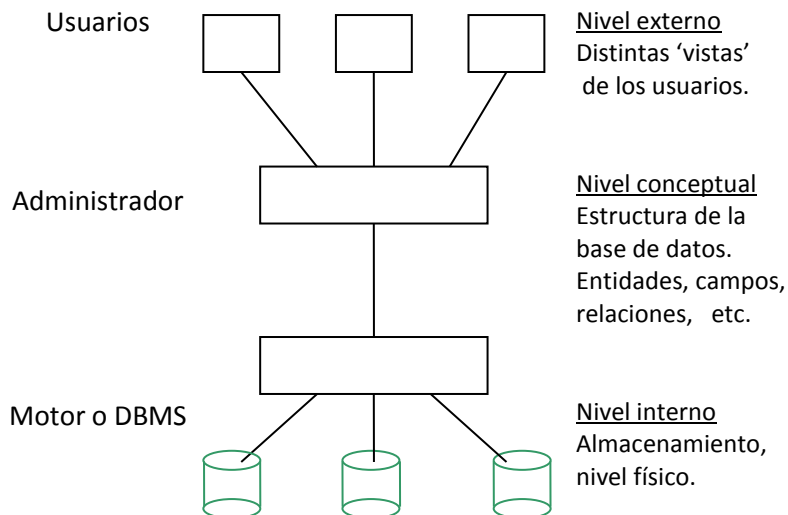
Nivel externo

Nivel conceptual

Nivel interno.

La representación de los niveles de visión se puede observar en la Figura 1.2.





**Figura1.2. Sistemas Gestores de Base de Datos (SGBD). Arquitectura ANSI-SPARC.**

- ❑ **Nivel externo o vistas de cada usuario.** Es el nivel más cercano al usuario y representa la percepción individual de cada usuario. Si bien los niveles interno y conceptual describen toda la BD, este nivel describe únicamente la parte de datos para un usuario o grupo de usuarios. Habrá usuarios que podrán acceder a más de un esquema externo y uno de éstos puede ser compartido por varios usuarios, se protege así el acceso a los datos por parte de personas no autorizadas. A la hora de construir un esquema externo:

- ? Se pueden omitir una o más entidades del sistema.
- ? Se pueden omitir uno o más atributos de una entidad.
- ? Se pueden omitir una o más relaciones entre los datos.
- ? Se pueden cambiar el orden de los atributos.

- ❑ **Nivel Conceptual.** En el se describen cuáles son los datos reales almacenados en la BD y que relaciones existen entre ellas. Este nivel lo definen los diseñadores/administradores de la BD que son los que deciden que información se guarda en la BD. Este nivel corresponde a la estructura organizacional de los datos obtenida al reunir los requerimientos de todos los usuarios, sin preocuparse de su organización física ni de las vías de acceso, ni del SGBD en el cual se va a implementar
  
- ❑ **Nivel interno o físico.** Se refiere al almacenamiento físico en el se describe cómo se almacenan realmente los datos en memorias secundarias, en qué archivos, su nombre y dirección. También estarán los registros, longitud, campos, índices y las rutas de acceso a esos archivos, es decir es la codificación interna de la base de datos.

**Importante:**

**“Para una BD habrá un único esquema interno, un único esquema conceptual, pero puede haber varios esquemas externos.”**

### **Objetivos de los S.G.B.D.**

En un ambiente multiusuario el S.G.B.D ofrece a la empresa un control centralizado de su información. Los objetivos que se plantean estos sistemas están relacionados con la intención de evitar los problemas que existían en los sistemas de información orientados a los procesos. Los principales objetivos son:

- Evitar la redundancia de los datos, eliminando así la inconsistencia de los mismos.
- Mejorar los mecanismos de seguridad de los datos y la privacidad. Podemos distinguir cuatro tipos de contextos para usar mecanismos de seguridad: seguridad contra accesos indebidos a los datos, seguridad contra accesos no autorizados a la BD, seguridad contra

destrucción causada por el entorno (fuego, inundación, robo, ...), seguridad contra fallos del propio sistema (fallos del hardware, del software, ...).

- Asegurar la independencia de los programas y los datos, es decir, la posibilidad de modificar la estructura de la base de datos (esquema) sin necesidad de modificar los programas de las aplicaciones que manejan esos datos.
- Mantener la integridad de los datos realizando las validaciones necesarias cuando se realicen modificaciones en la base de datos.
- Mejorar la eficacia de acceso a los datos, en especial en el caso de consultas imprevistas.

### **AutoEvaluación. Ejercicios Propuestos**

Escriba un concepto de Base de Datos.

Escriba un concepto de Base de Datos Relacional.

Indique 5 Ventajas de las Bases de Datos relacionales sobre los Sistemas de Archivos Convencionales.

Escriba 5 Funciones de un SGBD.

Grafique los Niveles de Visión según ANSI/SPARC.

**CAPITULO II****METODOLOGIA DE DISEÑO DE BASES DE DATOS****Análisis y Diseño de Bases de Datos**

Antes de diseñar una base de datos se debe establecer un proceso partiendo desde los casos reales, de manera que sea posible plasmar cada detalle de los mismos, en un sistema de información que sea utilizable y manipulable.

Este proceso se denomina modelamiento, y consiste en un conjunto de pasos que permitirán organizar los datos desde su forma más simple, tomando en cuenta las relaciones con otros datos y los procesos que los afectarán.

Para empezar este proceso de modelado, es necesario entender que son los modelos de abstracción de datos.

**Modelos para diseñar Bases de datos**

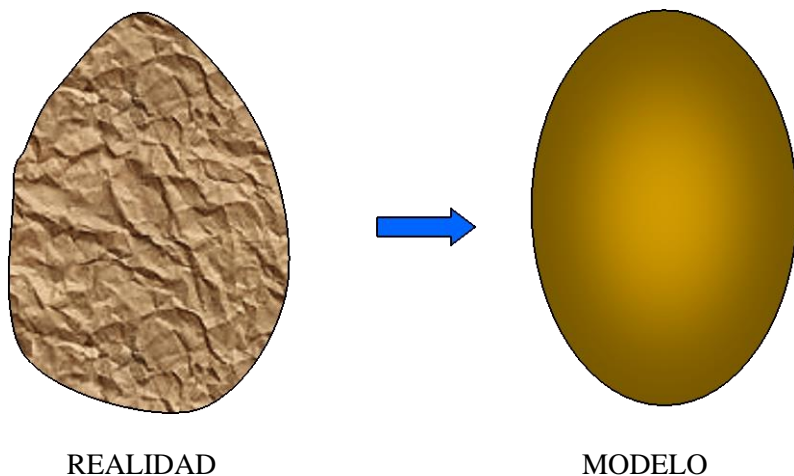
**Modelo:** Es una representación de la realidad que contiene las características generales de alguna entidad u objeto.

En base de datos, esta representación se la elabora de forma gráfica.

**Modelo de datos.** Es una colección de herramientas conceptuales que permiten describir los datos, sus propiedades, y las relaciones que existen con otros datos, manteniendo siempre la semántica asociada a los datos y restricciones de consistencia.

La imagen que se obtiene de un objeto del mundo real por medio de un modelo debe definirlo de manera total, con todas las características relevantes para el sistema.

**Abstracción.** Es la actividad por la cual se recogen las características comunes más relevantes y esenciales de un objeto de la realidad, para generar un modelo. Así lo muestra la Figura 2.1.



**Figura 2.1. Abstracción de la realidad para generar un modelo**

El modelo generado, deberá tener un comportamiento, desde el punto de vista de quién lo usará, semejante a la realidad que representa. Por esta razón el modelo deberá poseer tantos atributos de la realidad, como corresponda a su operación interna y a su relación con otros modelos.

### **Introducción al modelado de bases de datos**

Al diseñar un sistema de información o un proyecto de tecnología se debe tener en cuenta varios factores que intervienen en el desarrollo del mismo. El éxito del proyecto dependerá de la calidad con que se desarrollen todas las etapas que se identifiquen.

Algunas consideraciones se relacionan con reconocer ciertos componentes que participan en el diseño de una solución tecnológica, donde se incluye

el análisis, el diseño lógico y físico de la solución y posteriormente su implantación.

En el ámbito de los sistemas de información, se reconoce que el estudio de los procesos de negocio deberá desembocar prontamente en el establecimiento de un modelo lógico de datos, que refleje de la mejor forma posible la complejidad que puede llegar a adquirir el sistema real.

Cuando se estudia un proceso de negocio o una situación cualquiera, se presenta una importante diversidad de componentes o casos, unos muy frecuentes, otros menos y algunos eventuales.

Cuando se trata de crear un sistema que refleje este proceso, surge la pregunta ¿cuáles son los componentes que voy a considerar en el diseño?. Cuanto mayor sea el número de componentes, casos o situaciones, considerados, mayor será la complejidad del diseño. Particularmente, si deben considerarse las situaciones excepcionales, el diseño será muy complejo y por ende caro.

### **Metodología de Diseño de Bases de Datos Relacionales**

El diseño de una base de datos es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en subproblemas y se resuelve cada uno de estos subproblemas independientemente, utilizando técnicas específicas.

Es necesario tener claro que la calidad del producto final dependerá de la calidad de cada una de sus fases.

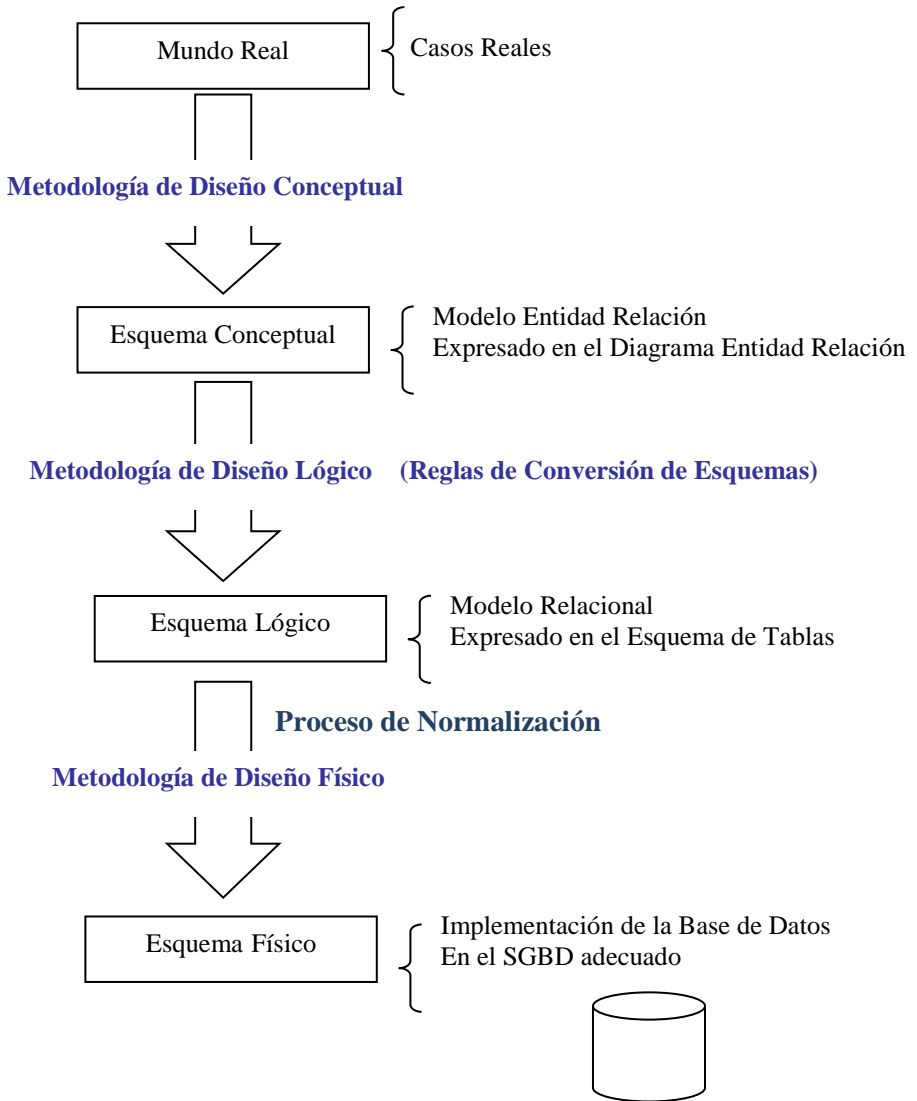
Así, la Metodología de Diseño de Bases de Datos Relacionales se divide en 3 metodologías de diseño, cada una de ellas definida por sus fases, así:

1. Metodología de Diseño Conceptual.
2. Metodología de Diseño Lógico.
3. Metodología de Diseño Físico.

Dentro de esta metodología es importante mencionar que cada paso siguiente, depende de la validez y calidad del paso anterior, es por eso que se debe ser extremadamente cuidadoso para no arrastrar errores desde las fases iniciales.

Además previo a la aplicación de la Metodología de Diseño Físico (Implementación), se debe aplicar un Proceso de Normalización, para minimizar la redundancia y maximizar el rendimiento de la Base de Datos.

Gráficamente la Metodología de Diseño de Bases de Datos Relacionales se expresa en la Figura 2.2.



**Figura 2.2. Metodología de Diseño de Bases de Datos Relacionales**



## 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 la documentación de la empresa, 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.

En general el diseño conceptual inicia desde los casos reales, es decir como están relacionados los datos en la realidad, cuales son sus características, sus propiedades, sus identificadores, sus relaciones, lo cual sumado a las especificaciones de requisitos de usuario, dan como su resultado el esquema conceptual de la base de datos.

El esquema conceptual es una descripción de alto nivel de la estructura de la base de datos, independientemente del SGBD que se vaya a utilizar para implementarla. Para obtener el esquema conceptual es necesario utilizar un modelo de abstracción de datos, denominado Modelo Conceptual.

**Modelo Conceptual.** Es un conjunto de herramientas conceptuales que permiten obtener una representación gráfica o semántica de la realidad a un alto nivel de abstracción. Mediante los modelos conceptuales se puede conocer el contenido de la información que tendrá la base de datos, pero no las estructuras de almacenamiento que se necesitarán para manejar esta información.

Un modelo conceptual debe permitir construir una descripción de la realidad fácil de entender representar, y de la manera correcta, por lo que es necesario que cumpla con las siguientes características:

- a) *Expresividad*: Debe tener suficientes conceptos para expresar perfectamente la realidad.
- b) *Simplicidad*: Debe ser simples para que los esquemas sean fáciles de entender.
- c) *Minimalidad*: Cada concepto debe tener un significado distinto.
- d) *Formalidad*: Todos los conceptos deben tener una interpretación única, precisa y bien definida.

El modelo conceptual utilizado dentro de esta metodología es el Modelo Entidad Relación.

### **Modelo Entidad Relación**

Denominado por sus siglas como: MER o ER.

El Modelo Entidad-Relación es el modelo conceptual más utilizado para el diseño conceptual de bases de datos. Fue introducido por Peter Chen en 1976. El Modelo Entidad-Relación está formado por un conjunto de herramientas conceptuales que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas.

Este modelo representa gráficamente a la realidad a través de entidades, relaciones, atributos, etc.

La principal herramienta de este modelo es el Diagrama Entidad Relación.

### **Diagrama Entidad Relación.**

Denominado por sus siglas como DER, es una técnica de representación gráfica, que incorpora información relativa a los datos y la relación

existente entre ellos, para poder así plasmar una visión del mundo real sobre un soporte informático. Sus características fundamentales son:

- Reflejan tan sólo la existencia de los datos sin expresar lo que se hace con ellos.
- Es independiente de las bases de datos y de los sistemas operativos
- Incluye todos los datos que se estudian sin tener en cuenta las aplicaciones que se van a tratar

### Conceptos fundamentales.

**Entidad.** Una entidad es una cosa u objeto que puede ser concreto o abstracto, que tiene características propias, y que sus datos representan interés para el sistema y sobre el que se recoge información la cual va a ser representada en un sistema de base de datos.

Las entidades pueden ser concretas o tangibles como: clientes, productos, estudiantes, autos, etc. o pueden ser abstractas o intangibles como sucesos por ejemplo: compras, ventas, pedidos, préstamos, etc.

En el DER las entidades se representan gráficamente mediante rectángulos y su nombre en plural aparece en el interior. Un nombre de entidad sólo puede aparecer una vez en el esquema conceptual. (Ver Figura 2.3).



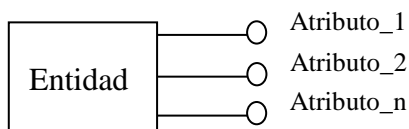
Nombre\_de\_la Entidad

### Figura 2.3. Representación de un Entidad

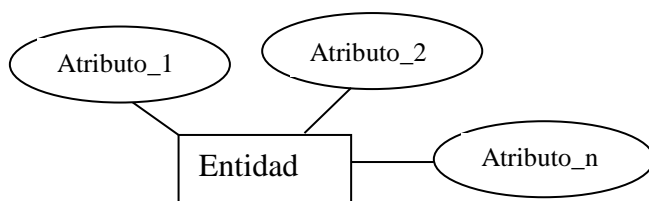
**Ocurrencias de una Entidad.** Constituye el número de veces que tiene esa entidad. Por ejemplo la entidad TAXIS, tendrá como número de ocurrencias la cantidad de taxis existen.

**Atributo:** Son las características de una entidad. El atributo es la unidad mínima e indivisible de información acerca de una entidad o una relación y sirve para identificar y describir a la misma. La determinación de los atributos que hay que incluir en el modelo es un problema del contexto de la base de datos, y se deben tomar decisiones basadas en el significado de los datos y en cómo se utilizarán.

En el DER se pueden representar los atributos mediante dos tipos de notaciones normalizadas, la notación de Codd (Ver Figura 2.4), y la notación de Chen (Ver Figura 2.5).



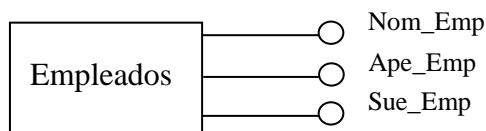
**Figura 2.4. Notación de Codd**



**Figura 2.5. Notación de Chen**

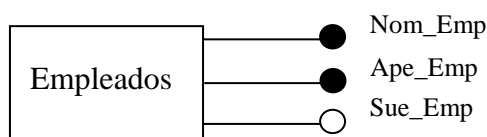
Por facilidad de implementación, se utilizará la notación de Codd.

Para nombrar a los atributos se puede utilizar la notación UML, es decir las 3 primeras letras del atributo, seguido de las 3 primeras letras de la entidad a la que referencia, iniciando cada una con mayúscula, o según notación internacional se puede utilizar las 3 primeras letras del atributo, guion bajo, y las 3 primeras letras de la entidad a la que referencia. Así.



**Figura 2.6. Ejemplo de Entidad con Atributos**

Si existen atributos que deben tener un valor obligatorio entonces estos se pintan, en caso de que el valor de un atributo no sea obligatorio, este no se pintará. Así.



**Dominio de un atributo:** Un dominio es el conjunto de valores legales y válidos que puede tomar un atributo.

La mayoría de las ocasiones se confunde el dominio con el tipo de dato, así que se ilustrará con un ejemplo la diferencia: si tomamos como atributo para una entidad persona su grupo sanguíneo, está claro que el tipo de dato que tendremos que usar será de tipo cadena de 3 caracteres **string(3)**; sin embargo, los únicos valores legales y válidos que puede tomar el atributo grupo sanguíneo, son los del conjunto {A+, A-, B+, B-, AB+, AB-, O+, O-}; este sería el dominio del atributo grupo sanguíneo. En caso de que un dato se encuentre en el tipo de dato, pero no se encuentre en el dominio, se habla de un dato sucio. Los datos sucios representan riesgo de inconsistencia en la base de datos y deben ser controlados..

**Tipos de Atributos.** Los atributos pueden ser estáticos, dinámicos, o calculados.

Un *atributo estático*, es aquel que no cambia con el tiempo.

Un *atributo dinámico* es aquel que puede cambiar con el tiempo.

Un *atributo calculado* es aquel que cambia a cada instante (edad, antigüedad) que se obtiene a partir del valor de uno o varios atributos, que no necesariamente deben pertenecer a la misma entidad o relación.

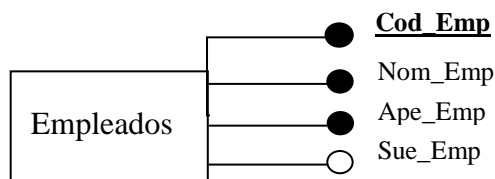
### Llaves o Claves.

**Llave principal (Clave Principal) (PRIMARY KEY):** Es un atributo o conjunto de atributos que permiten identificar de forma única a la ocurrencia de una entidad. **En el DER el atributo que es la llave principal, se subraya.**

**Clave ajena (Llave Foránea) (FOREIGN KEY):** Es un atributo que no es propio de la entidad, sino que se adiciona para mantener relación con otra entidad.

**Clave candidata:** Es el atributo o conjunto de atributos que pueden distinguir de forma unívoca una ocurrencia de una entidad. Puede haber varias claves candidatas para distinguir una misma entidad. Se elegirá como clave candidata aquel atributo que posea un dominio en el que se tenga valores únicos. Si esto no es posible, entonces se usará como clave candidata la combinación de varios atributos, de manera que esta combinación sí sea única.

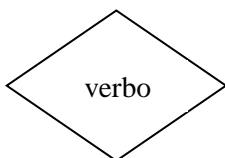
Una vez revisados estos conceptos, se puede generar el Diagrama Entidad Atributo. Para el ejemplo se tomará la entidad Empleado. (Ver Figura 2.7)



**Figura 2.7. Ejemplo de Diagrama Entidad-Atributo**

**Relación.** Es una asociación sin existencia propia que enlaza entidades. Por ejemplo: "curso" "tiene" "alumnos".

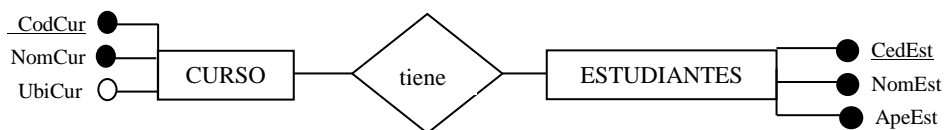
En el DER, las relaciones se representan gráficamente mediante rombos y opcionalmente en el interior un verbo alusivo a la relación. (Ver Figura2.8)



**Figura2.8. Representación de la Relación**

**Ligas.** En el Diagrama Entidad Relación, se representan en forma de líneas que unen a los diferentes símbolos.

En el Gráfico 2.9. se especifica un ejemplo de un Diagrama Entidad Relación para cursos y sus estudiantes, con los conceptos ya definidos.



**Grafico 2.9. Diagrama Entidad Relación Inicial**

### AutoEvaluación.

#### Ejercicios Propuestos:

En los siguientes casos reales.

Identifique las entidades, y sus correspondientes atributos.

1. Para un sistema de bases de datos Escolástico.
2. Para un sistema de bases de datos de Control de Matriculación Vehicular.
3. Para un sistema de Control de Transacciones Bancarias.

## Grados de una relación

Además de los atributos de cada entidad, un modelo de datos debe especificar las asociaciones existentes entre las entidades. Estas asociaciones son las relaciones entre las entidades. Por ejemplo, la frase "los clientes compran productos" nos dice que hay dos entidades, "Clientes" y "Productos", que están relacionadas por "comprar", pero no especifica ningún rango numérico entre sus ocurrencias.

La gran mayoría de las asociaciones son binarias, como "los clientes compran productos" o "los empleados venden productos". Entre las dos hay una asociación ternaria implícita: "los empleados venden productos a los clientes". Con las dos asociaciones binarias independientemente no podríamos saber a qué clientes se han vendido los productos que ha vendido un cierto empleado: en este caso necesitamos de la asociación ternaria.

Las asociaciones entre dos entidades cualesquiera pueden ser de tres tipos:

De Uno-a-uno.

De uno-a-muchos, y

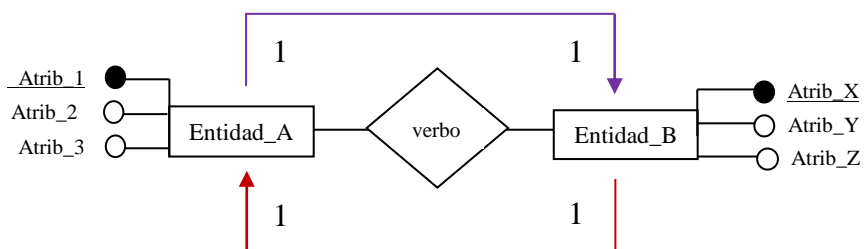
De muchos-a-muchos.

Relaciones Recursivas o Circulares.



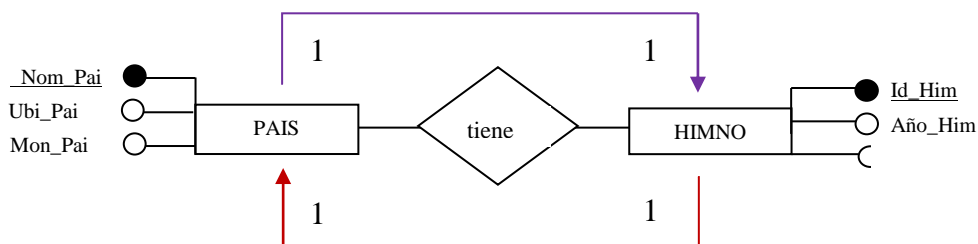
**Relación De Uno-a-Uno (One-to-One) (1 a 1).** Se da cuando a cada ocurrencia de la entidad A le corresponde una y solo una ocurrencia de la entidad B; y a cada ocurrencia de la entidad B le corresponde una y solo una ocurrencia de la entidad A.

Cuando definimos una relación como de uno-a-uno debemos asegurarnos de que se mantiene la asociación en todo momento, es decir desde la entidad A hacia la entidad B debe ser de uno a uno, y también desde la entidad B hacia la entidad A debe ser de uno a uno. Ver (Figura 2.10.).



**Figura 2.10. Esquema Relación de Uno a Uno**

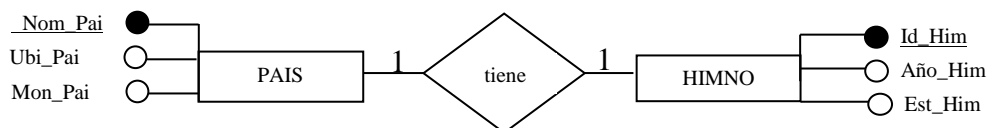
Por ejemplo: Si tomamos la relación: Un país tiene un himno, debemos asegurarnos que el análisis semántico indique que: En un país hay un solo himno, y que un himno puede serlo solo de un país en todo momento. (Ver Figura 2.11)



**Figura2.11. Ejemplo de la relación de Uno a Uno**

Cabe destacar que la relación Uno a Uno se da entre entidades, y es necesario no confundir entidades con atributos.

En el esquema Conceptual se debe especificar así: (Ver Figura 2.12)

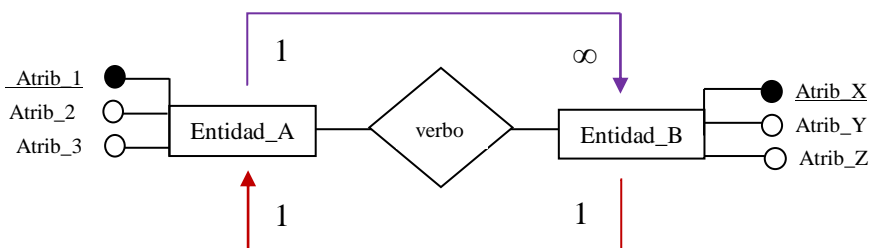


**Figura2.12. Esquema de la relación de Uno a Uno**

### AutoEvaluacion. Ejercicios Propuestos:

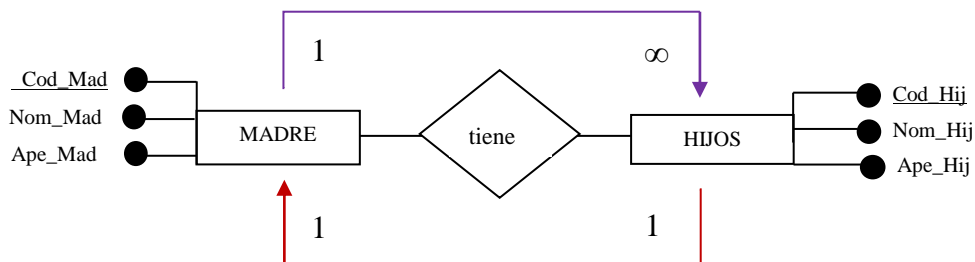
Indique y Grafique 5 relaciones De Uno a Uno de la realidad.

**Relación De Uno-a-muchos (One-to-Many) ( 1 a  $\infty$  ).** Se da cuando a cada ocurrencia de la entidad A le corresponden muchas ocurrencias de la entidad B; pero a cada ocurrencia de la entidad B le corresponde una y solo una ocurrencia de la entidad A. Cuando definimos una relación De uno-a-muchos debemos asegurarnos de que tomada la relación desde un lado sea uno a muchos, pero tomada la relación desde el otro lado sea uno a uno. Es decir desde la entidad A hacia la entidad B debe ser de uno a muchos, pero desde la entidad B hacia la entidad A debe ser de uno a uno. Ver (Figura 2.13.).



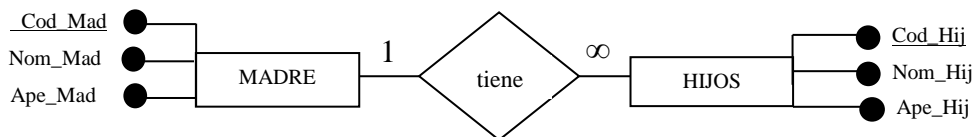
**Figura2.13. Relación de Uno a Muchos**

Por ejemplo: Si tomamos la relación: Una madre tiene muchos hijos, debemos asegurarnos que el análisis semántico indique que: Una madre puede tener muchos hijos, y que un hijo puede tener solo una madre. Nota. Más de uno, ya se considera muchos. (Ver Figura 2.14)



**Figura 2.14. Ejemplo de la relación de Uno a Muchos**

En el esquema Conceptual se debe especificar así: (Ver Figura 2.15)



**Figura 2.15. Esquema de la relación de Uno a Varios**

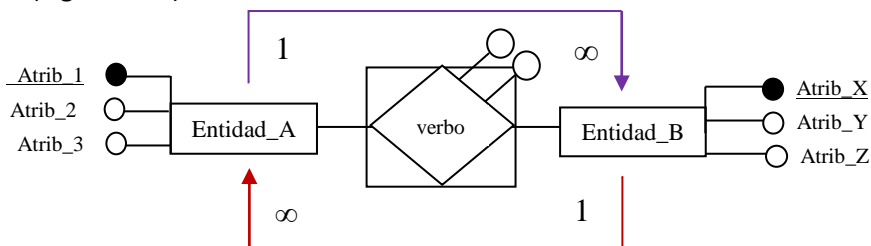
### AutoEvaluacion. Ejercicios Propuestos:

Indique y Grafique 5 relaciones De Uno a Muchos de la realidad.

**Relación De Muchos-a-Muchos (Many-to-Many) ( $\infty$  a  $\infty$ ).** Se da cuando a cada ocurrencia de la entidad A le corresponden varias ocurrencias de la entidad B; y a cada ocurrencia de la entidad B le corresponden varias ocurrencias de la entidad A.

Cuando definimos una relación De Muchos-a-Muchos debemos asegurarnos de que tomada la relación desde un lado sea uno a muchos, y

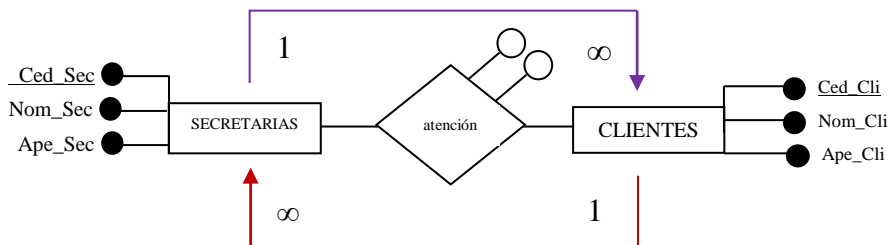
tomada la relación desde el otro lado también sea varios a muchos. Es decir desde la entidad A hacia la entidad B debe ser de uno a muchos, y desde la entidad B hacia la entidad A también ser de uno a muchos. La relación se rodea por un rectángulo, indicando que es una relación-tabla. Ver (Figura 2.16).



**Figura 2.16. Relación de Muchos a Muchos**

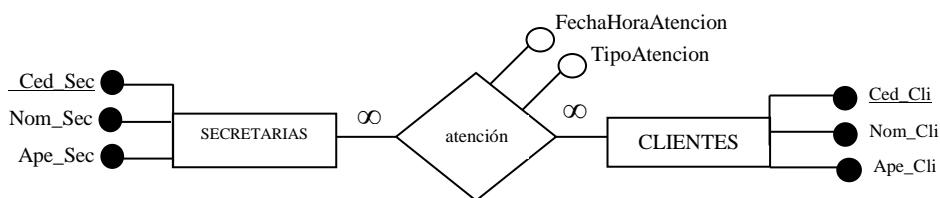
Por ejemplo si analizamos las atenciones en un banco, tendremos que: muchas secretarias atienden a muchos clientes, pero los datos de la atención son atributos de la relación-tabla.

En este análisis debemos fijarnos que: Una misma secretaria puede atender a varios clientes (pero en diferente momento, con un motivo distinto de atención), y un mismo cliente puede ser atendido por diferentes secretarias (así mismo en diferente momento, y con diferentes características de atención). Como se muestra en la Figura 2.17.



**Figura 2.17. Ejemplo de la relación de Varios a Varios**

En el esquema Conceptual se debe especificar así: (Ver Figura 2.18)



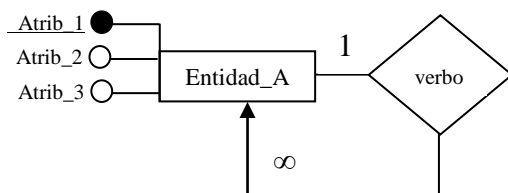
**Figura 2.18. Esquema de la relación de Varios a Varios**

### AutoEvaluacion. Ejercicios Propuestos:

Indique y Grafique 5 relaciones De Varios a Varios de la realidad.

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.

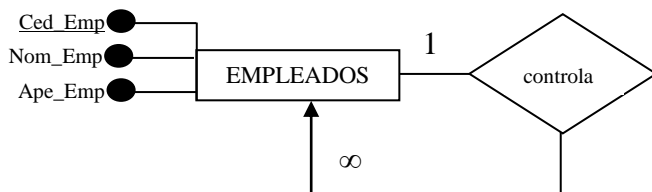
**Relaciones recursivas.** Es una relación donde la misma entidad participa más de una vez en la relación con distintos papeles. El nombre de estos papeles es importante para determinar la función de cada participación. (Ver Figura 2.19)



**Figura2.19. Esquema de una relación Recursiva**

Por ejemplo si tenemos una relación de un supervisor: Un empleado supervisor, controla a varios otros empleados, la relación es recursiva, porque el supervisor también es un empleado con atributos de empleado, pero con un cargo extra que es supervisor.

En el esquema conceptual se debe especificar así: (Ver Figura 2.20)



**Figura2.20. Ejemplo de una relación Recursiva**

### **AutoEvaluacion. Ejercicios Propuestos:**

Indique y Grafique 5 relaciones Recursivas de la realidad.

## Cardinalidad de una relación

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.

La participación es No Obligatoria (opcional o parcial), si la existencia de cada una de sus ocurrencias no se requiere.

Las reglas que definen la cardinalidad de las relaciones son las reglas de negocio. A veces, surgen problemas cuando se está diseñado un esquema conceptual. Estos problemas, denominados *trampas*, suelen producirse a causa de una mala interpretación en el significado de alguna relación, por lo que es importante comprobar que el esquema conceptual carece de dichas trampas. En general, para encontrar las trampas, hay que asegurarse de que se entiende completamente el significado de cada relación. Si no se entienden las relaciones, se puede crear un esquema que no represente fielmente la realidad.

Otra de las trampas sucede cuando un esquema sugiere la existencia de una relación entre entidades, pero el camino entre una y otra no existe para algunas de sus ocurrencias. En este caso, se produce una pérdida de información que se puede subsanar introduciendo la relación que sugería el esquema y que no estaba representada.

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

## Modelo Entidad Relación Extendido (EER) (MERE)

El Modelo Entidad Relación Extendido pretende aportar soluciones a requerimientos un tanto más complejos no contemplados en el Modelo Entidad Relación propuesto por Codd.

Así se incorporan al modelo E-R dos nuevos elementos:

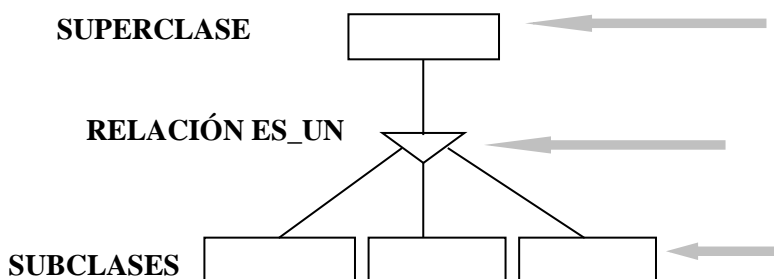
- Generalizaciones (Superclase/subclase)
- Agregaciones

### GENERALIZACIONES (Superclase/subclase)

Una **superclase** es todo tipo entidad sobre el que se definen subclases. Como se trata de entidades, se representan con un rectángulo.

Una **subclase** es un subconjunto del tipo entidad que tiene sentido en el minimundo ya que tiene atributos particulares. Como se trata de entidades, se representa al igual que ellas con un rectángulo.

La generalización una superclase y sus subclases, se define como tipo **ES\_UN**. Este tipo relación se representa a diferencia del resto de relaciones con un triángulo. El tipo ES\_UN define que una entidad no puede estar en dos subclases distintas y además, una entidad puede estar en dos o más superclases distintas. Como lo muestra la Figura2.21

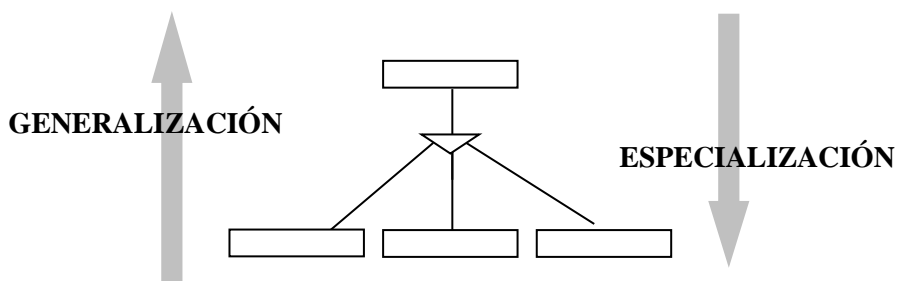


**Figura 2.21. Jerarquías de SuperClase/SubClase**



El proceso para determinar la necesidad de usar estos elementos puede producirse en dos sentidos:

- Especialización: es el proceso de definir un conjunto de subclases a partir de un tipo entidad.
- Generalización: es el proceso de suprimir las diferencias entre varios tipo entidad, identificando sus cualidades comunes.  
(Ver Figura 2.22)

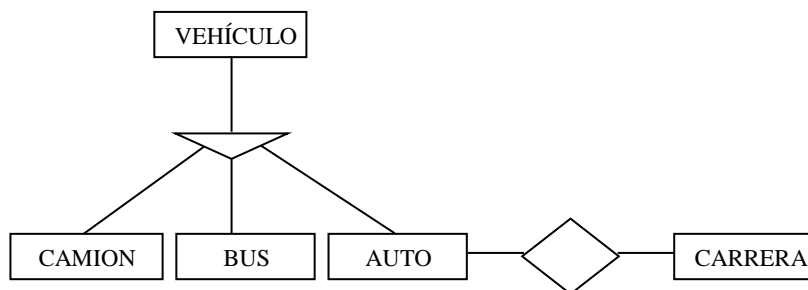


**Figura 2.22. Jerarquías de Generalización / Especialización**

El uso de estos elementos se justifica y recomienda en dos casos:

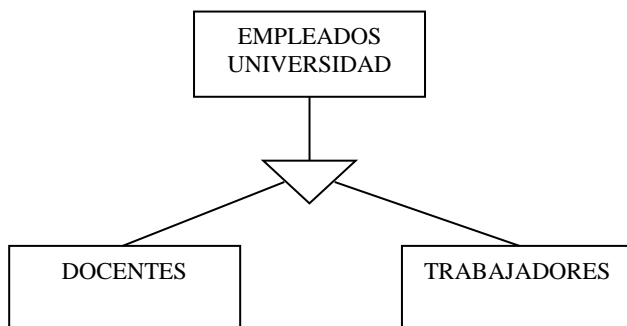
1. Cuando las subclases tienen atributos particulares que no tiene la superclase.
2. Cuando existen tipos relación en los que participan solo algunas subclases.

En el ejemplo (Ver Figura 2.23). La entidad camión, la entidad BUS, y la entidad AUTO cumplen con el tipo ES\_UN vehículo, y por tanto heredan los atributos del mismo, aunque un CAMIÓN tendrá sus propios atributos muy distintos a los de un BUS, y a su vez muy distintos a los atributos de un auto, además de que también van a cumplir funciones diferentes.

**Figura 2.23. Ejemplo de Generalización**

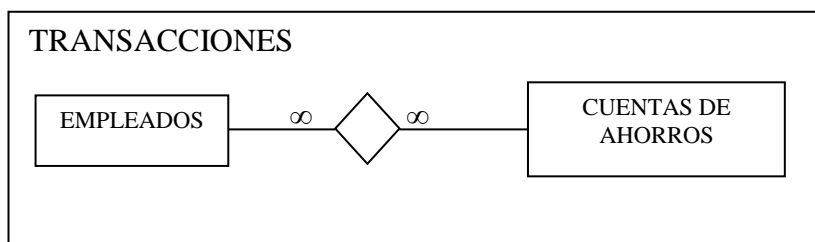
Ejemplo 2:

Una superclase/subclase para EMPLEADOS\_UNIVERSIDAD, indicará que en la superclase EMPLEADOS\_UNIVERSIDAD estarán los atributos comunes de todos los empleados de la universidad, y en las subclases estarán los atributos específicos de cada subclase, por ejemplo de los docentes se almacenan atributos muy diferentes a los que se almacenan de los trabajadores. Esta generalización se puede descomponer usando la técnica de especialización para generar las entidades apropiadas, con atributos no redundantes, así se muestra en la Figura 2.24.

**Figura 2.24. Ejemplo de Generalización/Especialización**

## AGREGACIONES

Agregación es un elemento que nos permite relacionar una relación con otra entidad. Hasta el momento solo se podían relacionar entidades. Este nuevo elemento permite relacionar relaciones con atributos propios (resultado de asociaciones de varios-a-varios) con entidades, o relaciones entre si siempre que el caso lo requiera. Por Ejemplo. Una agregación con la relación de varios a varios entre Empleados y Cuentas, en un banco creando una nueva superentidad llamada Transacciones, la cual tendrá sus propios atributos, así (Ver Figura 2.25):



**Figura 2.25. Ejemplo de Agregación**

### AutoEvaluación. Ejercicios Propuestos

Indique y Grafique 5 Generalizaciones.

Indique y Grafique 5 Agregaciones.

## Pasos de la Metodología de Diseño Conceptual

Con los conceptos anteriores, en esta etapa se debe construir un esquema de la información que se usa en la empresa, independientemente de cualquier consideración física. A este esquema se le denominará 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, que 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 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.

Para construir el Esquema Conceptual o Modelo Conceptual (Modelo Entidad Relación), que es el producto final de esta Metodología de Diseño Conceptual, es necesario aplicar un conjunto de pasos muy cuidadosamente, e ir generando la documentación correspondiente, la misma que se irá produciendo durante el desarrollo de la Metodología.

Los pasos a realizar en el diseño conceptual son las siguientes:

1. Identificar las entidades.
2. Identificar los atributos y asociarlos a entidades.
3. Identificar las relaciones.
4. Determinar los dominios de los atributos.
5. Determinar los identificadores y claves primarias.
6. Determinar las jerarquías de generalización (si existen).
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 representan interés para el usuario. Estos objetos son 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 propiedades de 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.

Para unirlo aún más, los usuarios usan, muchas veces, sinónimos y homónimos. Dos palabras son sinónimos cuando tienen el mismo significado. Los homónimos ocurren cuando la misma palabra puede tener distintos significados dependiendo del contexto.

No siempre es obvio saber si un objeto es una entidad, una relación o un atributo. Por ejemplo ¿cómo se podría clasificar matrimonio? Pues de cualquiera de las tres formas. El análisis es subjetivo, por lo que distintos diseñadores pueden hacer distintas interpretaciones, aunque todas igualmente válidas. Todo depende de la opinión y la experiencia de cada uno. Los diseñadores de bases de datos deben tener una visión selectiva y clasificar las cosas que observan dentro del contexto de la empresa u organización. A partir de unas especificaciones de usuario es posible que no se pueda deducir un conjunto único de entidades, pero después de varias iteraciones del proceso de análisis, se llegará a obtener un conjunto de entidades que sean adecuadas para el sistema que se ha de construir.

Conforme se van identificando las entidades, se les dan nombres que tengan un significado y que sean obvias para el usuario. Los nombres de las entidades y sus descripciones se anotan en el diccionario de datos. Cuando sea posible, se debe anotar también el número aproximado de ocurrencias de cada entidad (cardinalidad). Si una entidad se conoce por varios nombres, éstos se deben anotar en el diccionario de datos como alias o sinónimos.

## 2. Identificar los atributos y asociarlos a entidades

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.

Lo más sencillo es preguntarse, para cada entidad y cada relación, ¿qué información se quiere saber de ...?. La respuesta a esta pregunta se debe encontrar en las especificaciones de requisitos. Pero, en ocasiones, será necesario preguntar a los usuarios para que aclaren los requisitos. Desgraciadamente, los usuarios pueden dar respuestas a esta pregunta que también contengan otros conceptos, por lo que hay que considerar sus respuestas con mucho cuidado.

Al identificar los atributos, hay que tener en cuenta si son simples o compuestos. Por ejemplo, el atributo dirección puede ser simple, teniendo la dirección completa como un solo valor: 'Cevallos 54 y Mera'; o puede ser un atributo compuesto, formado por la calle principal ('Cevallos'), el número ('54') y la calle secundaria ('Mera'). El escoger entre atributo simple o compuesto depende de los requisitos del usuario. Si el usuario no necesita acceder a cada uno de los componentes de la dirección por separado, se puede representar como un atributo simple. Pero si el usuario quiere acceder a los componentes de forma individual, entonces se debe representar como un atributo compuesto.

También se deben identificar los atributos derivados o calculados, que son aquellos cuyo valor se puede calcular a partir de los valores de otros atributos. Por ejemplo, el número de empleados de cada oficina, la edad de los empleados o el número de inmuebles que gestiona cada empleado. Algunos diseñadores no representan los atributos derivados en los esquemas conceptuales. Si se hace, se debe indicar claramente que el atributo es derivado y a partir de qué atributos se obtiene su valor. Donde hay que considerar los atributos derivados es en el diseño físico.

Cuando se están identificando los atributos, se puede descubrir alguna entidad que no se ha identificado previamente, por lo que hay que volver al principio introduciendo esta entidad y viendo si se relaciona con otras entidades.

Es muy útil elaborar una lista de atributos e ir eliminándolos de la lista conforme se vayan asociando a una entidad o relación. De este modo, uno se puede asegurar de que cada atributo se asocia a una sola entidad o relación, y que cuando la lista se ha acabado, se han asociado todos los atributos.

Hay que tener mucho cuidado cuando parece que un mismo atributo se debe asociar a varias entidades. Esto puede ser por una de las siguientes causas:

- Se han identificado varias entidades, como *director*, *supervisor* y *administrativo*, cuando, de hecho, pueden representarse como una sola entidad denominada *empleado*. En este caso, se puede escoger entre introducir una jerarquía de generalización, o dejar las entidades que representan cada uno de los puestos de empleado.
- Se ha identificado una relación entre entidades. En este caso, se debe asociar el atributo a una sola de las entidades y hay que asegurarse de que la relación ya se había identificado previamente. Si no es así, se debe actualizar la documentación para recoger la nueva relación.

Conforme se van identificando los atributos, se les asignan nombres que tengan significado para el usuario. De cada atributo se debe anotar la siguiente información:

- Nombre y descripción del atributo.
- Alias o sinónimos por los que se conoce al atributo.
- Tipo de dato y longitud.



- Valores por defecto del atributo (si se especifican).
- Si el atributo siempre va a tener un valor (si admite o no nulos).
- Si el atributo es compuesto, qué atributos simples lo forman.
- Si el atributo es derivado y, en su caso, cómo se calcula su valor.
- Si el atributo es multievaluado.

### 3. 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.

Pero sólo interesan las relaciones que son necesarias. En el ejemplo anterior, se han identificado las relaciones *empleado gestiona inmueble* y *cliente visita inmueble*. Se podría pensar en incluir una relación entre empleado y cliente: *empleado atiende a cliente*, pero observando las especificaciones de requisitos no parece que haya interés en modelar tal relación.

La mayoría de las relaciones son binarias (entre dos entidades), pero no hay que olvidar que también puede haber relaciones en las que participen más de dos entidades, así como relaciones recursivas.

Es muy importante repasar las especificaciones para comprobar que todas las relaciones, explícitas o implícitas, se han encontrado. Si se tienen pocas entidades, se puede comprobar por parejas si hay alguna relación entre ellas. De todos modos, las relaciones que no se identifican ahora se suelen encontrar cuando se valida el esquema con las transacciones que debe soportar.

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.

#### **4. Determinar los dominios de los atributos**

El dominio de un atributo es el conjunto de valores legales y válidos que puede tomar el atributo. Por ejemplo el dominio de los números de oficina son las cadenas de hasta tres caracteres en donde el primero es una letra y el siguiente o los dos siguientes son dígitos en el rango de 1 a 99; el dominio de los números de teléfono son cadenas de 9 dígitos.

Un esquema conceptual está completo si incluye los dominios de cada atributo: los valores permitidos para cada atributo, su tamaño y su formato. También se puede incluir información adicional sobre los dominios como, por ejemplo, las operaciones que se pueden realizar sobre cada atributo, qué atributos pueden compararse entre sí o qué atributos pueden combinarse con otros. Aunque sería muy interesante que el sistema final respetara todas estas indicaciones sobre los dominios, esto es todavía una línea abierta de investigación.

Toda la información sobre los dominios se debe anotar también en el diccionario de datos.

## **5. Determinar los identificadores y claves primarias**

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.

## **7. Dibujar el Diagrama Entidad-Relación**

Una vez identificados todos los conceptos, se puede dibujar el Diagrama Entidad-Relación correspondiente, obteniendo así el esquema conceptual.

## **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 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.

### **AutoEvaluación: Ejercicios Propuestos**

Aplicando la Metodología de Diseño Conceptual, Generar el Esquema Conceptual (Diagrama Entidad Relación), para los siguientes casos reales:

1) Una cooperativa de taxis brinda servicio a diferentes hospitales de la región. En la cooperativa trabajan los choferes de los taxis, que se caracterizan por su número de licencia, nombre y años de servicio. A la cooperativa pertenecen varios taxis, pero un taxi puede pertenecer a una sola cooperativa. De los taxis se conoce la placa, la marca, el modelo, el año. Un taxi puede ser conducido por diferentes choferes, en diferentes momentos, pero se da que un chofer siempre es asignado al mismo taxi, y no puede conducir otro taxi. Un mismo taxi puede servir a varios hospitales de la región, y en un hospital se requiere el servicio de muchos taxis. De la cooperativa se conoce el nombre, la dirección y el número de taxis que son afiliados a la cooperativa, de los hospitales se conoce el nombre, tipo, y dirección. La base de datos debe permitir conocer cual taxi sirvió a cual hospital en un momento determinado.

2) Se desea diseñar una BDD para el movimiento mercantil de un organismo. Se conoce que empresas importadoras traen varios tipos de productos desde diferentes lugares de origen. Un mismo producto puede ser traído por varias importadoras, y una importadora trae diferentes tipos de productos. De cada empresa importadora se conoce el código, el nombre, y el tipo de empresa. De cada producto se conoce el número que lo identifica, la descripción y el precio unitario. Las empresas importadoras reparten los productos traídos a los distribuidores, cabe destacar que una empresa reparte a varios distribuidores, pero un distribuidor siempre es provisto de productos por la misma empresa importadora. El distribuidor

a su vez entrega los productos a varios almacenes terminales, pero un almacén terminal siempre recibe del mismo distribuidor, los almacenes terminales entregan los productos a diferentes comerciantes minoristas, y un mismo comerciante minorista puede adquirir los productos desde varios almacenes terminales. Finalmente los productos son entregados a los clientes por parte de los comerciantes minoristas, como es lógico, muchos clientes adquieren diferentes productos, y lo pueden hacer, comprándolo a cualquier comerciante minorista. De cada distribuidor se conoce el código, nombre, dirección. De cada almacén terminal se conoce el número, nombre, descripción, dirección. De cada comerciante minorista se conoce el número de CI, Nombre, y el tipo de comerciante. De los clientes finales se conoce el RUC, y el nombre. De la base de datos se desea filtrar el nombre del comerciante que atendió a un determinado cliente.

3) Representar mediante un DER las relaciones de maternidad, paternidad y matrimonio, que pueden existir entre hombres y mujeres. Tanto de los hombres, como de las mujeres, y los hijos se conoce: el código, el nombre y la edad.

4) Una empresa que se dedica al desarrollo de Software desea diseñar una BD para mantener información sobre sus empleados para su asignación a proyectos de forma eficiente. En particular se desea guardar la siguiente información: datos personales de cada empleado es decir CI, nombres y apellidos edad, fecha de ingreso a la empresa e información de los estudios académicos realizados por cada uno, indicando la fecha de finalización de los estudios. Cada empleado podrá tener, en función de su experiencia previa, uno o varios proyectos asignados, y a su vez a un determinado proyecto pueden ser asignados varios empleados. Además, se guarda información de los distintos proyectos con los que trabaja la empresa, como el número de expediente del proyecto, su nombre, costo y su descripción.

La base de datos debe permitir conocer el tiempo que un empleado le dedica aun determinado proyecto.

5) Una base de datos para una pequeña empresa debe contener información acerca de clientes, artículos y pedidos. Hasta el momento se registran los siguientes datos en documentos varios:

Para cada cliente: Número de cliente, Direcciones de envío, Saldo, Límite de crédito, Descuento.

Para cada artículo: Número de artículo, Existencias de ese artículo, Descripción del artículo.

Además, se ha determinado que se debe almacenar la información de las fábricas que reparten los artículos. Se usará: Número de la fábrica y Teléfono de contacto.

Y se desean conocer cuántos artículos (en total) provee la fábrica.

Modelar el diagrama ER si se conoce que varios clientes compran varios artículos, y que muchas empresas entregan un mismo artículo, pero una empresa entrega varios artículos, es decir que las empresas reparten muchos artículos para que sean vendidos a muchos clientes.

6) El Ministerio de Defensa desea diseñar una Base de Datos para llevar un cierto control de los soldados que realizan el servicio militar. Los datos significativos a tener en cuenta son:

- Un soldado se define por su código de soldado, su nombre y apellido, y su fecha de graduación.
- Existen varios cuarteles, cada uno se define por su código de cuartel, nombre y ubicación.
- Hay que tener en cuenta que existen diferentes Cuerpos del Ejército (Infantería, Artillería, Armada, ....etc), y cada uno se define por un código de Cuerpo y denominación.
- Los soldados están agrupados en compañías, siendo significativa para cada una de éstas, el número de compañía y la actividad principal que realiza.· Se desea controlar los puestos que ocupan los soldados (guardias, vigías, cuarteros, ...), y los puestos se definen por el código del puesto y descripción.

- Un soldado pertenece a un único cuerpo y a una única compañía durante todo el servicio militar. A una compañía pueden pertenecer soldados de diferentes cuerpos, no habiendo relación directa entre compañías y cuerpos.
- Los soldados de una misma compañía pueden estar destinados en diferentes cuarteles, es decir, una compañía puede estar ubicada en varios cuarteles, y en un cuartel puede haber varias compañías. Eso si, un soldado sólo esta en un solo cuartel.
- Un soldado ocupa varios puestos a lo largo de la milicia. Un mismo puesto puede ser ocupado por más de un soldado (con independencia de la compañía), siendo significativa la fecha de ingreso y salida del puesto.

7) Un concesionario de automóviles desea informatizar su gestión de ventas de vehículos. En particular, se quiere tener almacenada la información referente a los clientes que compran en el concesionario, los vehículos vendidos, así como los vendedores que realizan las distintas ventas. Para ello se tendrá en cuenta que:

El concesionario dispone de un catálogo de vehículos definidos por su marca, modelo, cilindraje y precio.

Cada uno de los modelos dispondrá de accesorios adicionales (aire acondicionado, pintura metalizada, etc.). Las opciones vienen definidas por un nombre y una descripción. Hay que tener en cuenta que una opción puede ser común para varios modelos variando sólo el precio en cada caso.

En cuanto a los clientes, la información de interés es el nombre, CI, dirección y teléfono, lo mismo que para los vendedores.

Los clientes pueden ceder su coche usado en el momento de comprar un vehículo nuevo. El coche usado vendrá definido por su marca, modelo, matrícula y precio de tasación. Es importante conocer la fecha en la que el cliente realiza esta sesión. Se desea saber qué vendedor ha vendido qué modelo a qué cliente. También la fecha de la venta y la matricula del nuevo vehículo. Es importante así mismo saber las opciones que el cliente ha elegido para el modelo que compra.

## **Metodología de Diseño Lógico**

Una vez obtenido el esquema conceptual, el siguiente paso es obtener el diseño lógico. El diseño lógico inicia desde el esquema conceptual, al cual se le aplica un conjunto de reglas de conversión, para obtener como resultado un esquema lógico. Un esquema lógico es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD.

### **Modelos Lógicos**

Son modelos que se utilizan para especificar los datos a un nivel de estructura lógica general, como un nivel de descripción más alto que el modelo conceptual.

Un modelo lógico es un lenguaje usado para especificar esquemas lógicos. Los modelos lógicos más reconocidos son el modelo relacional, el modelo de red, y modelo jerárquico.

El mas aceptado por su afinidad, para construir bases de datos relacionales es el Modelo Relacional.

### **Modelo Relacional.**

Fue introducido por Codd, muy a finales de los sesenta, de la teoría de las relaciones en el campo de las bases de datos supuso un importante paso en la investigación de los SGBD, suministrando un sólido fundamento teórico para el desarrollo, dentro de este enfoque relacional, de nuevos productos. 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 tablas relacionadas, 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 (1970), presentaba un nuevo modelo de datos que perseguía una serie de objetivos, que se pueden resumir en los siguientes.

- ✓ **Independencia física:** Es decir, el modo en el que se almacenan los datos no influye en su manipulación lógica y, por tanto, los usuarios que acceden a esos datos no tienen que modificar sus programas por cambios en el almacenamiento físico.
- ✓ **Independencia lógica:** Esto es, que el añadir, eliminar o modificar objetos de la base de datos no repercute en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos (vistas).
- ✓ **Flexibilidad:** En el sentido de poder presentar a cada usuario los datos de la forma en que éste prefiera.
- ✓ **Uniformidad:** Las estructuras lógicas de los datos presentan un aspecto uniforme, 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 relacional sea fácil de comprender y de utilizar por parte del usuario final.

Para conseguir los objetivos citados, Codd introduce el concepto de "relación" o tabla relacional como una estructura básica del modelo. Todos los datos de la base de datos se representan en forma de relaciones.

La ventaja del modelo relacional es que los datos se almacenan, al menos conceptualmente, de un modo en que los usuarios entienden con mayor facilidad. Los datos se almacenan como tablas y las relaciones entre las

filas y las tablas son visibles en los datos. Este enfoque permite a los usuarios obtener información de la base de datos sin asistencia de sistemas profesionales de administración de información.

En el enfoque relacional es sustancialmente distinto de otros enfoques en términos de sus estructuras lógicas y del modo de las operaciones de entrada/salida. En el enfoque relacional, los datos se organizan en tablas llamadas tablas relacionales o relaciones, cada una de las cuales se implanta como un archivo. En terminología relacional una fila en una relación representa un registro o una entidad; Cada columna en una relación representa un campo o un atributo.

Así, una relación se compone de una colección de entidades(o registros) cuyos propietarios están descritos por cierto número de atributos predeterminados implantados como campos.

### Conceptos Fundamentales

**Tabla:** Es la organización de los datos en forma de filas y columnas, en donde la intersección de una fila con una columna debe ser un valor atómico (indivisible). Una fila se llama tupla, y cada columna se llama campo. La tabla (llamada también tabla relacional o relación) es el elemento básico en el modelo relacional y se representa así: (Ver Figura 2.26)

#### Nombre

Atributo 1	Atributo 2	.....	Atributo n	
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	Tupla 1
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	Tupla 2
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	.
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	.
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	Tupla n

**Figura 2.26. Esquema de una Tabla Relacional**

En la tabla anterior podemos distinguir un conjunto de columnas, denominadas atributos, que representan propiedades de la misma y que están caracterizadas por un nombre; y un conjunto de filas llamadas tuplas que son las ocurrencias de la relación.

El número de filas de una relación se denomina cardinalidad de la relación y el número de columnas es el grado de la relación. Por ejemplo tomaremos la tabla Autor. (Ver Figura 2.26.)

## AUTOR

Nombre	Nacionalidad	Institucion
Pepe	España	O.N.U.
John	EE.UU.	O.M.S.
Pierre	Francia	N.A.S.A.

**Figura 2.26. Ejemplo de Tabla Relacional.**

**Tabla relacional:** Es una tabla que debe cumplir las siguientes características:

- ✓ Debe estar identificada por un nombre.
- ✓ No puede haber filas duplicadas, es decir, todas las tuplas tienen que ser distintas.
- ✓ El orden de las filas es irrelevante.
- ✓ No puede haber 2 columnas con el mismo nombre.
- ✓ El orden de las columnas es irrelevante.
- ✓ Los valores de las columnas deben pertenecer al dominio de cada atributo.
- ✓ La tabla es plana, es decir, en el cruce de una fila y una columna sólo puede haber un valor atómico e indivisible
- ✓ Debe tener una llave primaria.

**Tabla Base.** Es aquella cuya definición y estructura se encuentran creadas físicamente en la base de datos.

**Tabla Vista.** Es una tabla ficticia cuya definición y tuplas se obtiene a partir de una o más tablas base. Sus características son:

- Sus columnas se obtienen a partir de varias tablas base
- Pueden estar definidas a partir de otras vistas
- Sus datos se obtienen como resultado de una consulta a la base de datos
- Se puede almacenar su estructura

**Dominio.** Es un conjunto finito de valores legales y válidos que puede tener una columna.

Todo dominio debe tener un nombre por el cual nos podamos referir a él y un tipo de datos; así el tipo de datos del dominio "nacionalidades" es una cadena de caracteres de longitud 10.

El dominio "nacionalidades" tiene valores: España, Francia,... Si descompusiéramos España en E,s,p,... perdería la semántica.

Ejemplos de dominios serían:

Colores: Es el conjunto de los colores  $D=\{\text{rojo, verde, azul,}\}$

Números de CI: Es conjunto de números del Cedula de Identidad válidos.

Edad: Edades posibles de los empleados entre 18 y 80 años.

Sexo: Conjunto de valores legales para Sexo: Masculino o Femenino.

Un atributo es el papel que tiene un determinado dominio en una relación.

Es muy usual dar el mismo nombre al atributo y al dominio. En el caso de que sean varios los atributos de una misma tabla definidos sobre el mismo

dominio, habrá que darles nombres distintos, ya que una tabla no puede tener dos atributos con el mismo nombre.

Por ejemplo los atributos *edad\_física* y *edad\_mental* pueden estar definidos sobre el mismo dominio *edad*; o los atributos *precio\_compra* y *precio\_venta* pueden estar definidos sobre el mismo dominio de enteros.

Además de los dominios y atributos simples que acabamos de definir, en los últimos trabajos de algunos autores [Codd (1990), Date (1990)] se introduce el concepto de dominio compuesto.

Un dominio compuesto se puede definir como una combinación de dominios simples que tiene un nombre y a la que se pueden aplicar ciertas restricciones de integridad. Por ejemplo, un usuario puede necesitar manejar, además de los tres dominios *Día*, *Mes* y *Año*, un dominio compuesto denominado *Fecha* que sería la combinación de los tres primeros, y al que podríamos aplicar las adecuadas restricciones de integridad a fin de que no aparecieran valores no válidos para la fecha; algo análogo ocurre con el nombre y los apellidos, que, según las aplicaciones, puede ser conveniente tratarlos en conjunto o por separado.

De la misma forma, es posible definir un atributo compuesto *Fecha* que tomaría sus valores del dominio compuesto de igual nombre.

## Llaves o Claves

**Clave candidata.** Es un conjunto no vacío de atributos que identifican unívoca y mínimamente cada tupla. Por la propia definición de relación, siempre hay al menos una clave candidata, ya que al ser la relación un conjunto no existen tuplas repetidas y por tanto, el conjunto de todos los atributos identificará unívocamente a las tuplas. Una relación puede tener más de una clave candidata, entre las cuales se debe distinguir:

**Clave primaria:** Es una clave candidata que el usuario escogerá, por consideraciones ajenas al modelo relacional, para identificar a las tuplas

de una relación.

La clave primaria de una tabla es aquella que permite identificar de forma única a una fila de la tabla.

**Clave alternativa:** Son aquellas claves candidatas que no han sido elegidas.

**Clave Foránea (Clave Ajena).** Se denomina clave foránea de una relación R2 a un conjunto no vacío de atributos cuyos valores han de coincidir con los valores de la clave primaria de otra relación R1. La clave foránea y la correspondiente clave primaria han de estar definidas sobre los mismos dominios.

**Importante.** La clave foránea de una Tabla A es una columna que no es propia de la Tabla A, sino que se ha adicionado para mantener relación con una Tabla B. La clave foránea de la Tabla A, debe ser clave primaria de la Tabla B.

La regla de integridad de entidad establece que "Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo"; esto es, un valor desconocido o inexistente. Esta restricción debería aplicarse también a las claves alternativas, pero el modelo no lo exige, sino que el diseñador debe especificar esas restricciones.

La restricción de integridad referencial dice que los valores de clave foránea deben coincidir con los de clave primaria asociada a ella.

La integridad referencial es una restricción de comportamiento ya que viene impuesta por el mundo real y es el usuario quien la define al describir el esquema relacional; es también de tipo implícito, ya que se define en el esquema y el modelo la reconoce (o así algunos productos) sin necesidad de que se programe ni de que se tenga que escribir ningún procedimiento para obligar a que se cumpla.

## Pasos de la Metodología de diseño lógico en el modelo relacional

Es el proceso de construir el esquema lógico global a partir del esquema conceptual global, esta implementación lógica es 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. Esta técnica se presenta en el capítulo dedicado al diseño lógico de bases de datos.

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.

La metodología que se va a seguir para obtener el diseño lógico en el modelo relacional consta de dos fases, cada una de ellas compuesta por los pasos que se detallan a continuación.

- Construir y validar los esquemas lógicos locales para cada vista de usuario.
  1. Convertir los esquemas conceptuales locales en esquemas lógicos locales.
  2. Derivar un conjunto de relaciones (tablas) para cada esquema lógico local.
  3. Validar cada esquema mediante la normalización.
  4. Validar cada esquema frente a las transacciones del usuario.
  5. Dibujar el diagrama entidad-relación.
  6. Definir las restricciones de integridad.
  7. Revisar cada esquema lógico con el usuario correspondiente.
  
- Construir y validar el esquema lógico global.
  1. Mezclar los esquemas lógicos locales en un esquema lógico global.
  2. Validar el esquema lógico global.
  3. Estudiar el crecimiento futuro.
  4. Dibujar el diagrama entidad-relación final.
  5. Revisar el esquema lógico global con los usuarios.



## 1. Convertir los esquemas conceptuales locales en esquemas lógicos locales

En este paso se aplican las siguientes reglas:

1. Cada entidad genera una tabla
2. Los atributos de la entidad se convierten en columnas de la tabla.
3. Las ocurrencias de la entidad se convierten en filas (tuples) de la tabla.
4. Las claves primarias se conservan desde el esquema conceptual.
5. Además en este paso, se eliminan de cada esquema conceptual las estructuras de datos que los sistemas relacionales no modelan directamente.

**Esquema Formal.-** Se debe especificar el resultado del modelamiento como un esquema formal, es decir el nombre de la Tabla, y sus correspondientes columnas, así:

Tabla = (Columna1, Columna2, Columna3,...,ColumnaN).

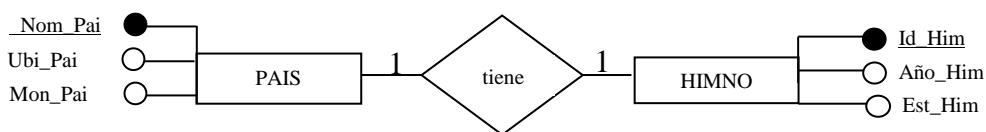
En donde la Clave Primaria se subraya.

Para las relaciones se aplicarán las siguientes Reglas de Conversión de Esquemas.

## Reglas de Conversión de Esquemas Conceptuales a Esquemas Lógicos

**5.1.** Si la relación es de Uno a Uno, se genera una sola tabla, con todos los atributos de las entidades asociadas con la relación de Uno a Uno, siendo la clave primaria de esta tabla, la clave de la entidad más importante.

Por ejemplo en la relación: Un país tiene un solo himno.



Se generará una sola tabla llamada PAIS, con todos los atributos de las 2 entidades, y cuya clave primaria será el Nombre del Pais, así:

### PAIS

<u>Nom_Pai</u>	Ubi_Pai	Mon_Pai	Id_Him	Año_Him	Est_Him
Ecuador	SudAmerica	Dolar	H001	1833	6
España	Europa	Euro	H005	1770	0

Esquema Formal:

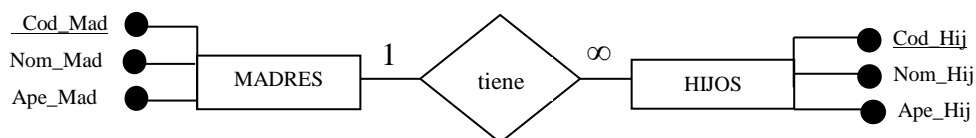
PAIS=(Nom\_Pai, Ubi\_Pai, Mon\_Pai, Id\_Him, Año\_Him, Est\_Him).

### AutoEvaluación. Ejercicios Propuestos

Transforme a esquema de tablas las relaciones planteadas en la sección de Autoevaluación de Relaciones Uno a Uno de la Pág. 26.

**5.2.** Si la relación es de Uno a Muchos, se generan 2 tablas, una tabla para cada entidad asociada, cada una con sus propios atributos y propia clave primaria desde el esquema conceptual, **pero la clave principal del de la tabla del lado 1, se adiciona a la tabla del lado muchos, como una columna extra** (que no es clave), para mantener la asociación.

Por ejemplo en la relación: Una madre tiene muchos hijos.



Se generarán dos tablas, pero la Clave Primaria de la Tabla Madre (lado1), se adicionará en la tabla Hijos(lado varios), para saber cuál Hijo le pertenece a cual Madre.

### MADRES

<u>Cod_Mad</u>	Nom_Mad	Ape_Mad
1501	Ana	López
1902	María	Sanchez



<u>Cod_Hij</u>	Nom_Hij	Ape_Hij	Cod_Mad_Per
1001	Jose	Perez	1501
1002	Ana	Pozo	1902
1008	Carlos	Moya	1902
1009	Alicia	Vasquez	1501

Esquema Formal:

MADRES=(Cod\_Mad,Nom\_Mad,Ape\_Mad).

ESTUDIANTES=(Cod\_Hij,Nom\_Hij,Ape\_Hij,Cod\_Mad\_Per).

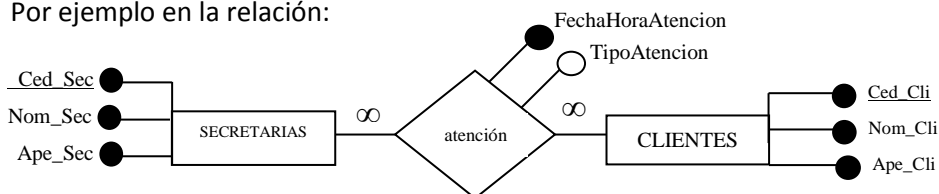
La integridad referencial es la regla que verifica que los valores que se ingresan en la Clave Foránea (CodSemPert), sean valores que existan en la Clave Primaria (CodSem) a la que se hace referencia.

### AutoEvaluación. Ejercicios Propuestos

Transforme a esquema de tablas las relaciones planteadas en la sección de AutoEvaluación de Relaciones Uno a Varios de la Pág. 27.

**5.3.** Si la relación es de Muchos a Muchos, se generan 3 tablas, una tabla para cada entidad asociada, cada una con sus propios atributos y propia clave primaria desde el esquema conceptual, pero además se genera una tercera tabla intermedia entre las 2 anteriores, para guardar la asociación, esta tercera tabla, esta tercera tabla tendrá como columnas, las claves primarias de las entidades asociadas, y los atributos de la relación si existieren.

Por ejemplo en la relación:



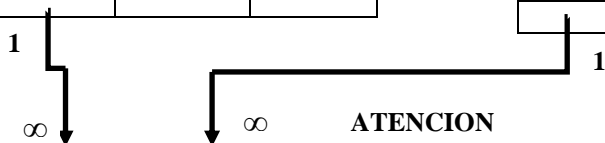
Se generaran las siguientes tablas:

### SECRETARIAS

<u>Ced_Sec</u>	Nom_Sec	Ape_Sec
1800001	Maria	Auz
1800002	Karla	Rios

### CLIENTES

<u>Ced_Cli</u>	Nom_Cli	Ape_Cli
1800006	Ruth	Garces
1800007	Susana	Villa



<u>Ced_Sec_Per</u>	<u>Ced_Cli_Per</u>	<u>Fecha_Hora_Ate</u>	<u>Tip_Ate</u>
1800001	1800006	15/Jul/2019 13:45:12	Consulta
1800002	1800006	15/Jul/2019 13:49:27	Prestamo
1800002	1800007	15/Jul/2019 15:22:08	Abono
1800001	1800006	15/Jul/2019 15:22:10	Consulta
1800001	1800007	15/Jul/2019 16:05:51	Transferencia

Esquema Formal:

SECRETARIAS=(Ced\_Sec,Nom\_Sec ,Ape\_Sec)

CLIENTES=(Ced\_Cli,Nom\_Cli,Ape\_Cli)

ATENCIÓN=(Ced\_Sec\_Per,Ced\_Cli\_Per,Fecha\_Hora\_Ate,Tip\_Ate)

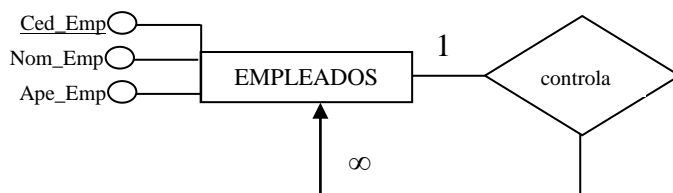
La clave primaria de la tabla intermedia dependerá de la regla del negocio.

Asimismo, la integridad referencial es verifica que los valores que se ingresan en Ced\_Sec\_Per, sean valores que existan en la Clave Primaria Ced\_Sec a la que se hace referencia, y que los valores que se ingresan en Ced\_Cli\_Per, sean valores que existan en la Clave Primaria Ced\_Cli a la que se hace referencia.

## AutoEvaluación

Transforme a esquema de tablas las relaciones planteadas en la sección de Autoevaluación de Relaciones Varios a Varios de la Pág. 29 .

**5.4.** Para las relaciones recursivas, se genera una sola tabla, y con una relación de uno a muchos, pero se aumenta el atributo al que se refiere la relación.



Se obtendrá la tabla:

### EMPLEADOS

<u>Ced_Emp</u>	<u>Nom_Emp</u>	<u>Ape_Emp</u>	<u>Supervisado_Por</u>
1700001	Juan	Mera	1800009
1700002	Marco	Padilla	1800009
1700003	Maria	Solis	1800009
1800009	Andres	López	

**5.5.** Para las relaciones ternarias, o relaciones entre tres o más entidades, generando las tablas en dependencia de su significado.

**5.6.** Tomar en cuenta que se debe eliminar las relaciones redundantes. Una relación es redundante cuando se puede obtener la misma información que ella aporta mediante otras relaciones. El hecho de que haya dos caminos diferentes entre dos entidades no implica que uno de

los caminos corresponda a una relación redundante, eso dependerá del significado de cada relación.

Una vez finalizado este paso, es más correcto referirse a los esquemas conceptuales locales refinados como esquemas lógicos locales, ya que se adaptan al modelo de base de datos que soporta el SGBD escogido.

## 2. Derivar un conjunto de tablas para cada esquema lógico local

En este paso, se obtiene un conjunto de tablas para cada uno de los esquemas lógicos locales en donde se representen las entidades y relaciones entre entidades, que se describen en cada una de las vistas que los usuarios tienen de la empresa. Cada relación de la base de datos tendrá un nombre, y el nombre de sus atributos aparecerá, a continuación, entre paréntesis. El atributo o atributos que forman la clave primaria se subrayan. Las claves ajenas, mecanismo que se utiliza para representar las relaciones entre entidades en el modelo relacional, se especifican aparte indicando la tabla a la que hacen referencia.

A continuación, se describe cómo las tablas del modelo relacional representan las entidades y relaciones que pueden aparecer en los esquemas lógicos.

**Claves Primarias.** Cada uno de los identificadores de la entidad será una clave candidata de la tabla. De entre las claves candidatas hay que escoger la clave primaria; el resto serán claves alternativas. Para escoger la clave primaria entre las claves candidatas se pueden seguir estas indicaciones:

- Escoger la clave candidata que tenga menos atributos.
- Escoger la clave candidata cuyos valores no tengan probabilidad de cambiar en el futuro.
- Escoger la clave candidata cuyos valores no tengan probabilidad de perder la unicidad en el futuro.
- Escoger la clave candidata con el mínimo número de caracteres (si es de tipo texto).

- Escoger la clave candidata más fácil de utilizar desde el punto de vista de los usuarios.

Es importante tener en cuenta que para cada relación binaria se incluyen los atributos de la clave primaria de la entidad padre en la tabla que representa a la entidad hijo, para actuar como una clave ajena. La entidad hijo es la que participa de forma total (obligatoria) en la relación, mientras que la entidad padre es la que participa de forma parcial (opcional). Si las dos entidades participan de forma total o parcial en la relación, la elección de padre e hijo es arbitraria. Además, en caso de que ambas entidades participen de forma total en la relación, se tiene la opción de integrar las dos entidades en una sola relación tabla. Esto se suele hacer si una de las

En las Jerarquías de generalización, se denomina entidad padre a la entidad genérica y entidades hijo a las subentidades. Hay tres opciones distintas para representar las jerarquías. La elección de la más adecuada se hará en función de su tipo (total/parcial, exclusiva/superpuesta).

1. Crear una tabla por cada entidad. Las relaciones de las entidades hijo heredan como clave primaria la de la entidad padre. Por lo tanto, la clave primaria de las entidades hijo es también una clave ajena al padre. Esta opción sirve para cualquier tipo de jerarquía, total o parcial y exclusiva o superpuesta.
2. Crear una relación por cada entidad hijo, heredando los atributos de la entidad padre. Esta opción sólo sirve para jerarquías totales y exclusivas.
3. Integrar todas las entidades en una tabla, incluyendo en ella los atributos de la entidad padre, los atributos de todos los hijos y un atributo discriminativo para indicar el caso al cual pertenece la entidad en consideración. Esta opción sirve para cualquier tipo de jerarquía. Si la jerarquía es superpuesta, el atributo discriminativo será multievaluado.



Una vez obtenidas las relaciones con sus atributos, claves primarias y claves ajenas, sólo queda actualizar el diccionario de datos con los nuevos atributos que se hayan identificado en este paso.

### **3. Validar cada esquema mediante la normalización**

La normalización se utiliza para mejorar el esquema lógico, de modo que satisfaga ciertas restricciones que eviten la duplicidad de datos. La normalización garantiza que el esquema resultante se encuentra más próximo al modelo de la empresa, que es consistente y que tiene la mínima redundancia y la máxima estabilidad.

La normalización es un proceso que permite decidir a qué entidad pertenece cada atributo. Uno de los conceptos básicos del modelo relacional es que los atributos se agrupan en relaciones (tablas) porque están relacionados a nivel lógico. En la mayoría de las ocasiones, una base de datos normalizada no proporciona la máxima eficiencia, sin embargo, el objetivo ahora es conseguir una base de datos normalizada por las siguientes razones:

- Un esquema normalizado organiza los datos de acuerdo a sus dependencias funcionales, es decir, de acuerdo a sus relaciones lógicas.
- El esquema lógico no tiene porqué ser el esquema final. Debe representar lo que el diseñador entiende sobre la naturaleza y el significado de los datos de la empresa. Si se establecen unos objetivos en cuanto a prestaciones, el diseño físico cambiará el esquema lógico de modo adecuado. Una posibilidad es que algunas relaciones normalizadas se desnormalicen. Pero la desnormalización no implica que se haya malgastado tiempo normalizando, ya que mediante este proceso el diseñador aprende más sobre el significado de los datos. De hecho, la normalización obliga a entender completamente cada uno de los atributos que se han de representar en la base de datos.

- Un esquema normalizado es robusto y carece de redundancias, por lo que está libre de ciertas anomalías que éstas pueden provocar cuando se actualiza la base de datos.
- Los equipos informáticos de hoy en día son mucho más potentes, por lo que en ocasiones es más razonable implementar bases de datos fáciles de manejar (las normalizadas), a costa de un tiempo adicional de proceso.
- La normalización produce bases de datos con esquemas flexibles que pueden extenderse con facilidad.

El objetivo de este paso es obtener un conjunto de relaciones que se encuentren en la Tercer Forma Normal. Para ello, hay que pasar por la primera, segunda y tercera formas normales. El proceso de normalización se describe en el capítulo III.

#### **4. Validar cada esquema frente a las transacciones del usuario**

El objetivo de este paso es validar cada esquema lógico local para garantizar que puede soportar las transacciones requeridas por los correspondientes usuarios. Estas transacciones se encontrarán en las especificaciones de requisitos de usuario. Lo que se debe hacer es tratar de realizar las transacciones de forma manual utilizando el diagrama entidad-relación, el diccionario de datos y las conexiones que establecen las claves ajenas de las tablas. Si todas las transacciones se pueden realizar, el esquema queda validado. Pero si alguna transacción no se puede realizar, seguramente será porque alguna entidad, relación o atributo no se ha incluido en el esquema.

#### **5. Dibujar el diagrama relacional**

En este momento, se puede dibujar el diagrama relacional final para cada vista de usuario que recoja la representación lógica de los datos desde su punto de vista. Este diagrama habrá sido validado mediante la normalización y frente a las transacciones de los usuarios.

## 6. Definir las restricciones de integridad

Las restricciones de integridad son reglas que imponen para proteger la base de datos, de modo que no pueda llegar a un estado inconsistente. Se debe tener en cuenta que la base de datos debe pasar siempre de un estado consistente a otro estado consistente; nunca debe pasar de un estado consistente a un estado inconsistente.

Hay cinco tipos de restricciones de integridad.

(a)

*Datos requeridos.* Algunos atributos deben contener valores en todo momento, es decir, no admiten nulos.

(b)

*Restricciones de dominios.* Todos los atributos tienen un dominio asociado, es decir un conjunto de los valores que cada atributo puede tomar.

(c)

*Integridad de entidades.* El identificador de una entidad no puede ser nulo, por lo tanto, las claves primarias de las tablas no deben admitir nulos.

(d)

*Integridad referencial.* Una clave foránea enlaza cada tupla de la relación hijo con la tupla de la relación padre que tiene el mismo valor en su clave primaria. La integridad referencial dice que si una clave foránea tiene valor (osea si no es nula), ese valor debe ser uno de los valores de la clave primaria a la que referencia. Hay varios aspectos a tener en cuenta sobre las claves foráneas para lograr que se cumpla la integridad referencial.

1. ¿Debe admitir nulos la clave foránea? Cada clave ajena expresa una relación. Si la participación de la entidad hijo

en la relación es total, entonces la clave ajena no admite nulos; si es parcial, la clave ajena debe aceptar nulos.

2. ¿Qué hacer cuando se quiere borrar una ocurrencia de la entidad padre que tiene algún hijo? O lo que es lo mismo, ¿qué hacer cuando se quiere borrar una tupla padre que está siendo referenciada por otra tupla a través de una clave ajena? Hay varias respuestas posibles:
  - *Restringir*: no se pueden borrar tuplas que están siendo referenciadas por otras tuplas.
  - *Propagar*: se borra la tupla deseada y se propaga el borrado a todas las tuplas que le hacen referencia.
  - *Anular*: se borra la tupla deseada y todas las referencias que tenía se ponen, automáticamente, a nulo (esta respuesta sólo es válida si la clave ajena acepta nulos).
  - *Valor por defecto*: se borra la tupla deseada y todas las referencias toman, automáticamente, el valor por defecto (esta respuesta sólo es válida si se ha especificado un valor por defecto para la clave ajena).
  - *No comprobar*: se borra la tupla deseada y no se hace nada para garantizar que se sigue cumpliendo la integridad referencial.
3. ¿Qué hacer cuando se quiere modificar la clave primaria de una tupla que está siendo referenciada por otra tupla a través de una clave ajena? Las respuestas posibles son las mismas que en el caso anterior. Cuando se escoge propagar, se actualiza la clave primaria en la tupla deseada y se propaga el cambio a los valores de clave ajena que le hacían referencia.

(e)

*Reglas de negocio.* Cualquier operación que se realice sobre los datos debe cumplir las restricciones que impone el funcionamiento de la empresa.

Todas las restricciones de integridad establecidas en este paso se deben reflejar en el diccionario de datos para que puedan ser tenidas en cuenta durante la fase del diseño físico.

## **7. Revisar cada esquema lógico local con el usuario correspondiente**

Para garantizar que cada esquema lógico local es una fiel representación de la vista del usuario lo que se debe hacer es comprobar con él que lo reflejado en el esquema y en la documentación es correcto y está completo.

El esquema lógico refleja la estructura de los datos a almacenar que maneja la empresa. Un diagrama de flujo de datos muestra cómo se mueven los datos en la empresa y los almacenes en donde se guardan. Si se han utilizado diagramas de flujo de datos para modelar las especificaciones de requisitos de usuario, se pueden utilizar para comprobar la consistencia y completitud del esquema lógico desarrollado. Para ello:

- Cada almacén de datos debe corresponder con una o varias entidades completas.
- Los atributos en los flujos de datos deben corresponder a alguna entidad.

Los esquemas lógicos locales obtenidos hasta este momento se integrarán en un solo esquema lógico global en la siguiente fase para modelar los datos de toda la empresa.

## 8. Mezclar los esquemas lógicos locales en un esquema lógico global

En este paso, se deben integrar todos los esquemas locales en un solo esquema global. En un sistema pequeño, con dos o tres vistas de usuario y unas pocas entidades y relaciones, es relativamente sencillo comparar los esquemas locales, mezclarlos y resolver cualquier tipo de diferencia que pueda existir. Pero en los sistemas grandes, se debe seguir un proceso más sistemático para llevar a cabo este paso con éxito:

1. Revisar los nombres de las entidades y sus claves primarias.
2. Revisar los nombres de las relaciones.
3. Mezclar las entidades de las vistas locales.
4. Incluir (sin mezclar) las entidades que pertenecen a una sola vista de usuario.
5. Mezclar las relaciones de las vistas locales.
6. Incluir (sin mezclar) las relaciones que pertenecen a una sola vista de usuario.
7. Comprobar que no se ha omitido ninguna entidad ni relación.
8. Comprobar las claves ajenas.
9. Comprobar las restricciones de integridad.
10. Dibujar el esquema lógico global.
11. Actualizar la documentación.

## 9. Validar el esquema lógico global

Este proceso de validación se realiza, de nuevo, mediante la normalización y mediante la prueba frente a las transacciones de los usuarios. Pero ahora sólo hay que normalizar las relaciones que hayan cambiado al mezclar los esquemas lógicos locales y sólo hay que probar las transacciones que requieran acceso a áreas que hayan sufrido algún cambio.

## **10. Estudiar el crecimiento futuro**

En este paso, se trata de comprobar que el esquema obtenido puede acomodar los futuros cambios en los requisitos con un impacto mínimo. Si el esquema lógico se puede extender fácilmente, cualquiera de los cambios previstos se podrá incorporar al mismo con un efecto mínimo sobre los usuarios existentes.

## **11. Dibujar el diagrama relacional final**

Una vez validado el esquema lógico global, ya se puede dibujar el diagrama entidad-relación que representa el modelo de los datos de la empresa que son de interés. La documentación que describe este modelo (incluyendo el esquema relacional y el diccionario de datos) se debe actualizar y completar.

## **12. Revisar el esquema lógico global con los usuarios**

Una vez más, se debe revisar con los usuarios el esquema global y la documentación obtenida para asegurarse de que son una fiel representación de la empresa.

## **AutoEvaluación. Ejercicios Propuestos**

Sobre los esquemas conceptuales obtenidos en la sección de evaluación de la Pág. 44. Aplique la metodología de Diseño Lógico y Obtenga el Esquema Lógico.

## Reglas de Integridad y Reglas de Negocio

### Reglas de integridad

Una vez definida la estructura de datos del modelo relacional, se procederá a profundizar las reglas de integridad que los datos almacenados en dicha estructura deben cumplir para garantizar que son correctos.

Al definir cada atributo sobre un dominio se impone una restricción sobre el conjunto de valores permitidos para cada atributo. A este tipo de restricciones se les denomina *restricciones de dominios*. Hay además dos reglas de integridad muy importantes que son restricciones que se deben cumplir en todas las bases de datos relacionales y en todos sus estados o instancias (las reglas se deben cumplir todo el tiempo). Estas reglas son la *regla de integridad de entidades* y la *regla de integridad referencial*. Antes de definir las, es preciso conocer el concepto de *nulo*.

### 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 entidades

La primera regla de integridad se aplica a las claves primarias de las relaciones base: *ninguno de los atributos que componen la clave primaria puede ser nulo*.

Por definición, una clave primaria es un identificador irreducible que se utiliza para identificar de modo único las tuplas. Que es irreducible significa que ningún subconjunto de la clave primaria sirve para identificar las tuplas de modo único. Si se permite que parte de la clave primaria sea nula, se está diciendo que no todos sus atributos son necesarios para distinguir las tuplas, con lo que se contradice la irreducibilidad.

Nótese que esta regla sólo se aplica a las relaciones base y a las claves primarias, no a las claves alternativas.

## Regla de integridad referencial

La segunda regla de integridad se aplica a las claves foráneas: *si en una relación hay alguna clave foránea, sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o bien, deben ser completamente nulos*.

La regla de integridad referencial se enmarca en términos de estados de la base de datos: indica lo que es un estado ilegal, pero no dice cómo puede evitarse. La cuestión es ¿qué hacer si estando en un estado legal, llega una petición para realizar una operación que conduce a un estado ilegal? Existen dos opciones: *rechazar* la operación, o bien *aceptar* la operación y realizar operaciones adicionales compensatorias que conduzcan a un estado legal.

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

## Reglas de negocio

Además de las dos reglas de integridad anteriores, los usuarios o los administradores de la base de datos pueden imponer ciertas restricciones específicas sobre los datos, denominadas *reglas de negocio*.

Por ejemplo, si en una oficina de la empresa inmobiliaria sólo puede haber hasta veinte empleados, el SGBD debe dar la posibilidad al usuario de definir una regla al respecto y debe hacerla respetar. En este caso, no debería permitir dar de alta un empleado en una oficina que ya tiene los veinte permitidos.

## **Metodología de 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. El diseño físico, parte desde el esquema lógico y da como resultado un esquema físico. Un esquema físico depende del SGBD concreto escogido según la aplicación que va a tener la base de datos, y el esquema físico se expresa mediante su lenguaje de definición de datos.

### **Diseño de aplicaciones**

Se debe analizar dos aspectos del diseño de las aplicaciones: el diseño de las transacciones y el diseño de las interfaces de usuario.

### **Diseño de transacciones**

Una transacción es un conjunto de acciones llevadas a cabo por un usuario o un programa de aplicación, que acceden o cambian el contenido de la base de datos. Las transacciones representan eventos del mundo real, como registrar un inmueble para ponerlo en alquiler, concertar una visita con un cliente a un inmueble, dar de alta un nuevo empleado o registrar

un nuevo cliente. Estas transacciones se deben realizar sobre la base de datos para que ésta siga siendo un fiel reflejo de la realidad.

Una transacción puede estar compuesta por varias operaciones, como la transferencia de dinero de una cuenta bancaria a otra. Sin embargo, desde el punto de vista del usuario, estas operaciones conforman una sola tarea.

Desde el punto de vista del SGBD, una transacción lleva a la base de datos de un estado consistente a otro estado consistente. El SGBD garantiza la consistencia de la base de datos incluso si se produce algún fallo, y también garantiza que una vez se ha finalizado una transacción, los cambios realizados por ésta quedan permanentemente en la base de datos, no se pueden perder ni deshacer (a menos que se realice otra transacción que compense el efecto de la primera).

Si la transacción no se puede finalizar por cualquier motivo, el SGBD garantiza que los cambios realizados por esta transacción son deshechos. En el ejemplo de la transferencia de fondos entre dos cuentas bancarias, si el dinero se extrae de una cuenta y la transacción falla antes de que el dinero se ingrese en la otra cuenta, el SGBD deshacerá la extracción de fondos.

El objetivo del diseño de las transacciones es definir y documentar las características de alto nivel de las transacciones que requiere el sistema. Esta tarea se debe llevar a cabo al principio del proceso de diseño para garantizar que el esquema lógico es capaz de soportar todas las transacciones necesarias. Las características que se deben recoger de cada transacción son las siguientes:

- Datos que utiliza la transacción.
- Características funcionales de la transacción.
- Salida de la transacción.
- Importancia para los usuarios.
- Frecuencia de utilización.

Hay tres tipos de transacciones:

**Transacciones de recuperación.** Se accede a los datos para visualizarlos en la pantalla a modo de informe.

**Transacciones de actualización.** Se insertan, borran o actualizan datos de la base de datos.

**Transacciones mixtas.** Se mezclan operaciones de recuperación de datos y de actualización.

El diseño de las transacciones utiliza la información dada en las especificaciones de requisitos de usuario

### **Diseño de interfaces de usuario**

Antes de implementar los formularios y los informes, hay que diseñar su aspecto. Es conveniente tener en cuenta las siguientes recomendaciones:

- Utilizar títulos que sean significativos, que identifiquen sin ambigüedad el propósito del informe o formulario.
- Dar instrucciones breves y fáciles de comprender.
- Agrupar y secuenciar los campos de forma lógica.
- Hacer que el aspecto del informe o formulario sea atractivo a la vista.
- Utilizar nombres familiares para etiquetar los campos.
- Utilizar terminología y abreviaturas consistentes.
- Hacer un uso razonable y consistente de los colores.
- Dejar un espacio visible para los datos de entrada y delimitarlos.
- Permitir un uso sencillo y adecuado del cursor.
- Permitir la corrección carácter a carácter y de campos completos.
- Dar mensajes de error para los valores "ilegales".
- Marcar los campos que sean opcionales.
- Dar mensajes a nivel de campo para explicar su significado.

- Dar una señal que indique cuándo el informe o formulario está completo.

## **Pasos de la Metodología de Diseño Físico**

El objetivo de esta etapa es producir una descripción de la implementación de la base de datos en memoria secundaria. Esta descripción incluye las estructuras de almacenamiento y los métodos de acceso que se utilizarán para conseguir un acceso eficiente a los datos.

El diseño físico se divide de cuatro fases, cada una de ellas compuesta por una serie de pasos:

- Traducir el esquema lógico global para el SGBD específico.
  1. Diseñar las relaciones base para el SGBD específico.
  2. Diseñar las reglas de negocio para el SGBD específico.
- Diseñar la representación física.
  1. Analizar las transacciones.
  2. Escoger las organizaciones de ficheros.
  3. Escoger los índices secundarios.
  4. Considerar la introducción de redundancias controladas.
  5. Estimar la necesidad de espacio en disco.
- Diseñar los mecanismos de seguridad.
  1. Diseñar las vistas de los usuarios.
  2. Diseñar las reglas de acceso.
- Monitorizar y afinar el sistema.

## **Traducir el esquema lógico global**

La primera fase del diseño lógico consiste en traducir el esquema lógico global en un esquema que se pueda implementar en el SGBD escogido. Para ello, es necesario conocer toda la funcionalidad que éste ofrece. Por ejemplo, el diseñador deberá saber:

- Si el sistema soporta la definición de claves primarias, claves ajenas y claves alternativas.
- Si el sistema soporta la definición de datos requeridos (es decir, si se pueden definir atributos como no nulos).
- Si el sistema soporta la definición de dominios.
- Si el sistema soporta la definición de reglas de negocio.
- Cómo se crean las relaciones base.

## **1. Diseñar las relaciones base para el SGBD específico**

Las relaciones base se definen mediante el lenguaje de definición de datos (DDL) del SGBD. Para ello, se utiliza la información producida durante el diseño lógico: el esquema lógico global y el diccionario de datos. El esquema lógico consta de un conjunto de relaciones y, para cada una de ellas, se tiene:

- El nombre de la relación.
- La lista de atributos entre paréntesis.
- La clave primaria y las claves ajenas, si las tiene.
- Las reglas de integridad de las claves ajenas.

En el diccionario de datos se describen los atributos y, para cada uno de ellos, se tiene:

- Su dominio: tipo de datos, longitud y restricciones de dominio.
- El valor por defecto, que es opcional.
- Si admite nulos.
- Si es derivado y, en caso de serlo, cómo se calcula su valor.

A continuación, se muestra un ejemplo de la definición de la relación entre UN SEMESTRE TIENE MUCHOS ESTUDIANTES, implementada en SQL ANSI.

```
CREATE TABLE SEMESTRES
```

```
(  
Cod_Sem CHAR(5) PRIMARY KEY,  
Nom_Sem CHAR(15) NOT NULL,  
Des_Sem CHAR(20)  
);
```

```
CREATE TABLE ESTUDIANTES
```

```
(  
Ced_Est CHAR(10) PRIMARY KEY,  
Nom_Est CHAR(10) NOT NULL,  
Ape_Est CHAR(10) NOT NULL,  
Dir_Est CHAR(15) NOT NULL,  
Tel_Est CHAR(9),  
Pro_Est NUMBER NOT NULL,  
Cod_Sem_Per CHAR(5) REFERENCES SEMESTRES(CodSem)  
);
```

## 2. Diseñar las reglas de negocio para el SGBD específico

Las actualizaciones que se realizan sobre las relaciones de la base de datos deben observar ciertas restricciones que imponen las reglas de negocio de la empresa. Algunos SGBD proporcionan mecanismos que permiten definir estas restricciones y vigilan que no se violen.

Por ejemplo, si no se quiere que un SEMESTRE tenga más de treinta estudiantes asignados, se puede definir una restricción en la sentencia CREATE TABLE de la relación ESTUDIANTE:

```
CONSTRAINT estudiantes_por_semestre  
CHECK (NOT EXISTS (SELECT Cod_Sem_Per  
FROM ESTUDIANTES  
GROUP BY CodSemPert  
HAVING COUNT(*)>30))
```



Hay algunas restricciones que no las pueden manejar los SGBD, como por ejemplo “A las 20:30 del último día laborable de cada año archivar los artículos vendidos y borrarlos”. Para estas restricciones habrá que escribir programas de aplicación específicos. Por otro lado, hay SGBD que no permiten la definición de restricciones, por lo que éstas deberán incluirse en los programas de aplicación.

Todas las restricciones que se definan deben estar documentadas. Si hay varias opciones posibles para implementarlas, hay que explicar porqué se ha escogido la opción implementada.

### **Diseñar la representación física**

Uno de los objetivos principales del diseño físico es almacenar los datos de modo eficiente. Para medir la eficiencia hay varios factores que se deben tener en cuenta:

- *Productividad de transacciones.* Es el número de transacciones que se quiere procesar en un intervalo de tiempo.
- *Tiempo de respuesta.* Es el tiempo que tarda en ejecutarse una transacción. Desde el punto de vista del usuario, este tiempo debería ser el mínimo posible.
- *Espacio en disco.* Es la cantidad de espacio en disco que hace falta para los ficheros de la base de datos. Normalmente, el diseñador querrá minimizar este espacio.

Un problema que suele suceder, es que todos estos factores no se pueden satisfacer a la vez. Por ejemplo, para conseguir un tiempo de respuesta mínimo, puede ser necesario aumentar la cantidad de datos almacenados, ocupando más espacio en disco. Por lo tanto, el diseñador deberá ir ajustando estos factores para conseguir un equilibrio razonable. El diseño físico inicial no será el definitivo, sino que habrá que ir monitorizándolo para observar sus prestaciones e ir ajustándolo como sea

oportuno. Muchos SGBD proporcionan herramientas para monitorizar y afinar el sistema.

Hay algunas estructuras de almacenamiento que son muy eficientes para cargar grandes cantidades de datos en la base de datos, pero no son eficientes para el resto de operaciones, por lo que se puede escoger dicha estructura de almacenamiento para inicializar la base de datos y cambiarla, a continuación, para su posterior operación. Los tipos de organizaciones de ficheros disponibles varían en cada SGBD. Algunos sistemas proporcionan más estructuras de almacenamiento que otros. Es muy importante que el diseñador del esquema físico sepa qué estructuras de almacenamiento le proporciona el SGBD y cómo las utiliza.

Para mejorar las prestaciones, el diseñador del esquema físico debe saber cómo interactúan los dispositivos involucrados y cómo esto afecta a las prestaciones:

- *Memoria principal.* Los accesos a memoria principal son mucho más rápidos que los accesos a memoria secundaria (decenas o centenas de miles de veces más rápidos). Generalmente, cuanto más memoria principal se tenga, más rápidas serán las aplicaciones. Sin embargo, es aconsejable tener al menos un 5% de la memoria disponible, pero no más de un 10%. Si no hay bastante memoria disponible para todos los procesos, el sistema operativo debe transferir páginas a disco para liberar memoria ( *paging*). Cuando estas páginas se vuelven a necesitar, hay que volver a traerlas desde el disco ( *faltas de página*). A veces, es necesario llevar procesos enteros a disco ( *swapping*) para liberar memoria. El hacer estas transferencias con demasiada frecuencia empeora las prestaciones.
- *CPU.* La CPU controla los recursos del sistema y ejecuta los procesos de usuario. El principal objetivo con este dispositivo es lograr que no haya bloqueos de procesos para conseguirla. Si el sistema operativo, o los procesos de los usuarios, hacen muchas

demandas de CPU, ésta se convierte en un cuello de botella. Esto suele ocurrir cuando hay muchas faltas de página o se realiza mucho swapping.

- *Entrada/salida a disco.* Los discos tienen una velocidad de entrada/salida. Cuando se requieren datos a una velocidad mayor que ésta, el disco se convierte en un cuello de botella. Dependiendo de cómo se organicen los datos en el disco, se conseguirá reducir la probabilidad de empeorar las prestaciones. Los principios básicos que se deberían seguir para repartir los datos en los discos son los siguientes:
  - Los ficheros del sistema operativo deben estar separados de los ficheros de la base de datos.
  - Los ficheros de datos deben estar separados de los ficheros de índices
  - Los ficheros con los diarios de operaciones deben estar separados del resto de los ficheros de la base de datos.
- *Red.* La red se convierte en un cuello de botella cuando tiene mucho tráfico y cuando hay muchas colisiones.

Cada uno de estos recursos afecta a los demás, de modo que una mejora en alguno de ellos puede provocar mejoras en otros.

### 3. Analizar las transacciones

Para realizar un buen diseño físico es necesario conocer las consultas y las transacciones que se van a ejecutar sobre la base de datos. Esto incluye tanto información cualitativa, como cuantitativa. Para cada transacción, hay que especificar:

- La frecuencia con que se va a ejecutar.
- Las relaciones y los atributos a los que accede la transacción, y el tipo de acceso: consulta, inserción, modificación o eliminación. Los atributos que se modifican no son buenos candidatos para construir estructuras de acceso.

- Los atributos que se utilizan en los predicados del WHERE de las sentencias SQL. Estos atributos pueden ser candidatos para construir estructuras de acceso dependiendo del tipo de predicado que se utilice.
- Si es una consulta, los atributos involucrados en el join de dos o más relaciones. Estos atributos pueden ser candidatos para construir estructuras de acceso.
- Las restricciones temporales impuestas sobre la transacción. Los atributos utilizados en los predicados de la transacción pueden ser candidatos para construir estructuras de acceso.

#### **4. Escoger las organizaciones de ficheros**

El objetivo de este paso es escoger la organización de ficheros óptima para cada relación. Por ejemplo, un fichero desordenado es una buena estructura cuando se va a cargar gran cantidad de datos en una relación al inicializarla, cuando la relación tiene pocas tuplas, también cuando en cada acceso se deben obtener todas las tuplas de la relación, o cuando la relación tiene una estructura de acceso adicional, como puede ser un índice. Por otra parte, los ficheros dispersos (hashing) son apropiados cuando se accede a las tuplas a través de los valores exactos de alguno de sus campos (condición de igualdad en el WHERE). Si la condición de búsqueda es distinta de la igualdad (búsqueda por rango, por patrón, etc.), la dispersión no es una buena opción. Hay otras organizaciones, como la ISAM o los árboles B+.

Las organizaciones de ficheros elegidas deben documentarse, justificando en cada caso la opción escogida.

#### **5. Escoger los índices secundarios**

Los índices secundarios permiten especificar caminos de acceso adicionales para las relaciones base. Por ejemplo, la relación INMUEBLE se puede haber almacenado en un fichero disperso a través del atributo

inum. Si se accede a menudo a esta relación a través del atributo alquiler, se puede plantear la creación de un índice sobre dicho atributo para favorecer estos accesos. Pero hay que tener en cuenta que estos índices conllevan un coste de mantenimiento que hay que sopesar frente a la ganancia en prestaciones. A la hora de seleccionar los índices, se pueden seguir las siguientes indicaciones:

- Construir un índice sobre la clave primaria de cada relación base.
- No crear índices sobre relaciones pequeñas.
- Añadir un índice sobre los atributos que se utilizan para acceder con mucha frecuencia.
- Añadir un índice sobre las claves ajenas que se utilicen con frecuencia para hacer joins.
- Evitar los índices sobre atributos que se modifican a menudo.
- Evitar los índices sobre atributos poco selectivos (aquellos en los que la consulta selecciona una porción significativa de la relación).
- Evitar los índices sobre atributos formados por tiras de caracteres largas.

Los índices creados se deben documentar, explicando las razones de su elección.

## **6. Considerar la introducción de redundancias controladas**

En ocasiones puede ser conveniente relajar las reglas de normalización introduciendo redundancias de forma controlada, con objeto de mejorar las prestaciones del sistema. En la etapa del diseño lógico se recomienda llegar, al menos, hasta la tercera forma normal para obtener un esquema con una estructura consistente y sin redundancias. Pero, a menudo, sucede que las bases de datos así normalizadas no proporcionan la máxima eficiencia, con lo que es necesario volver atrás y desnormalizar algunas relaciones, sacrificando los beneficios de la normalización para mejorar las prestaciones. Es importante hacer notar que la desnormalización sólo debe realizarse cuando se estime que el sistema no

puede alcanzar las prestaciones deseadas. Y, desde luego, la necesidad de desnormalizar en ocasiones no implica eliminar la normalización del diseño lógico: la normalización obliga al diseñador a entender completamente cada uno de los atributos que se han de representar en la base de datos. Por lo tanto, hay que tener en cuenta los siguientes factores:

- La desnormalización hace que la implementación sea más compleja.
- La desnormalización hace que se sacrifique la flexibilidad.
- La desnormalización puede hacer que los accesos a datos sean más rápidos, pero ralentiza las actualizaciones.

Por regla general, la desnormalización de una relación puede ser una opción viable cuando las prestaciones que se obtienen no son las deseadas y la relación se actualiza con poca frecuencia, pero se consulta muy a menudo. Las redundancias que se pueden incluir al desnormalizar son de varios tipos: se pueden introducir datos derivados (calculados a partir de otros datos), se pueden duplicar atributos o se pueden hacer joins de relaciones.

El incluir un atributo derivado dependerá del coste adicional de almacenarlo y mantenerlo consistente con los datos de los que se deriva, frente al coste de calcularlo cada vez que se necesita.

No se pueden establecer una serie de reglas que determinen cuándo desnormalizar relaciones, pero hay algunas situaciones muy comunes en donde puede considerarse esta posibilidad:

- *Combinar relaciones de uno a uno.* Cuando hay relaciones (tablas) involucradas en relaciones de uno a uno, se accede a ellas de manera conjunta con frecuencia y casi no se les accede separadamente, se pueden combinar en una sola tabla.

- *Duplicar atributos no clave en relaciones de uno a muchos para reducir los joins.* Para evitar operaciones de join, se pueden incluir atributos de la relación (tabla) padre en la relación (tabla) hijo de las relaciones de uno a muchos.
- *Tablas de referencia.* Las tablas de referencia ( *lookup*) son listas de valores, cada uno de los cuales tiene un código. Por ejemplo puede haber una tabla de referencia para los tipos de inmueble, con las descripciones de estos tipos y un código asociado. Este tipo de tablas son un caso de relación de uno a muchos. En la relación INMUEBLE habrá una clave ajena a esta tabla para indicar el tipo de inmueble. De este modo, es muy fácil validar los datos, además de que se ahorra espacio escribiendo sólo el código y no la descripción para cada inmueble, además de ahorrar tiempo cuando se actualizan las descripciones. Si las tablas de referencia se utilizan a menudo en consultas críticas, se puede considerar la introducción de la descripción junto con el código en la relación (tabla) hijo, manteniendo la tabla de referencia para validación de datos.
- *Duplicar atributos en relaciones de muchos a muchos para reducir los joins.* Durante el diseño lógico se eliminan las relaciones de muchos a muchos introduciendo dos relaciones de uno a muchos. Esto hace que aparezca una nueva relación (tabla) intermedia, de modo que si se quiere obtener la información de la relación de muchos a muchos, se tiene que realizar el join de tres relaciones (tablas). Para evitar algunos de estos joins se pueden incluir algunos de los atributos de las relaciones (tablas) originales en la relación (tabla) intermedia.
- *Introducir grupos repetitivos.* Los grupos repetitivos se eliminan en el primer paso de la normalización para conseguir la primera forma normal. Estos grupos se eliminan introduciendo una nueva relación (tabla), generando una relación de uno a muchos. A veces, puede ser conveniente reintroducir los grupos repetitivos para mejorar las prestaciones.

Todas las redundancias que se introduzcan en este paso se deben documentar y razonar. El esquema lógico se debe actualizar para reflejar los cambios introducidos.

## **7. Estimar la necesidad de espacio en disco**

En caso de que se tenga que adquirir nuevo equipamiento informático, el diseñador debe estimar el espacio necesario en disco para la base de datos. Esta estimación depende del SGBD que se vaya a utilizar y del hardware. En general, se debe estimar el número de tuplas de cada relación y su tamaño. También se debe estimar el factor de crecimiento de cada relación.

## **8. Diseñar los mecanismos de seguridad**

Los datos constituyen un recurso esencial para la empresa, por lo tanto su seguridad es de vital importancia. Durante el diseño lógico se habrán especificado los requerimientos en cuanto a seguridad que en esta fase se deben implementar. Para llevar a cabo esta implementación, el diseñador debe conocer las posibilidades que ofrece el SGBD que se vaya a utilizar.

## **9. Diseñar las vistas de los usuarios**

El objetivo de este paso es diseñar las vistas de los usuarios correspondientes a los esquemas lógicos locales. Las vistas, además de preservar la seguridad, mejoran la independencia de datos, reducen la complejidad y permiten que los usuarios vean los datos en el formato deseado.

## **10. Diseñar las reglas de acceso**

El administrador de la base de datos asigna a cada usuario un identificador que tendrá una palabra secreta asociada por motivos de seguridad. Para cada usuario o grupo de usuarios se otorgarán permisos para realizar



determinadas acciones sobre determinados objetos de la base de datos. Por ejemplo, los usuarios de un determinado grupo pueden tener permiso para consultar los datos de una relación base concreta y no tener permiso para actualizarlos.

## **11. Monitorizar y afinar el sistema**

Una vez implementado el esquema físico de la base de datos, se debe poner en marcha para observar sus prestaciones. Si éstas no son las deseadas, el esquema deberá cambiar para intentar satisfacerlas. Una vez afinado el esquema, no permanecerá estático, ya que tendrá que ir cambiando conforme lo requieran los nuevos requisitos de los usuarios. Los SGBD proporcionan herramientas para monitorizar el sistema mientras está en funcionamiento.

### **AutoEvaluación. Ejercicios Propuestos**

Sobre los esquemas lógicos obtenidos en la sección de evaluación de la Pág. 72. Aplique la metodología de Diseño Físico y Obtenga el Esquema Físico. Implementado en SQL ANSI.

## Resumen del Capítulo

El diseño de una base de datos es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en subproblemas y se resuelve cada uno de estos subproblemas independientemente, utilizando técnicas específicas. Así, el diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico.

El diseño conceptual parte de las especificaciones de requisitos de usuario y su resultado es el esquema conceptual de la base de datos. Un *esquema conceptual* es una descripción de alto nivel de la estructura de la base de datos, independientemente del SGBD que se vaya a utilizar para manipularla. Un *modelo conceptual* es un lenguaje que se utiliza para describir esquemas conceptuales. El objetivo del diseño conceptual es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información.

El diseño lógico parte del esquema conceptual y da como resultado un esquema lógico. Un *esquema lógico* es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD. Un *modelo lógico* es un lenguaje usado para especificar esquemas lógicos (modelo relacional, modelo de red, etc.). El diseño lógico depende del tipo de SGBD que se vaya a utilizar, no depende del producto concreto.

El diseño físico parte del esquema lógico y da como resultado un esquema físico. Un *esquema físico* es una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Por ello, el diseño físico depende del SGBD concreto y el esquema físico se expresa mediante su lenguaje de definición de datos.

**Autoevaluación. Ejercicios Varios****EJERCICIO 1:**

En una institución de enseñanza existen varios laboratorios, en los cuales se dictan diferentes seminarios de capacitación en diferentes momentos. En un mismo laboratorio pueden dictarse diferentes seminarios, pero un determinado seminario se dicta en un mismo laboratorio siempre por cuestión de software instalado. Se da el caso de que un determinado profesor siempre dictará un mismo seminario. Un profesor dicta un mismo seminario a varios grupos de alumnos, y el mismo grupo de alumnos recibe varios seminarios. El hecho de que un profesor dicte un seminario a un grupo de alumnos se conoce como una tutoría presencial que esta identificada por el código, el día, la hora, y el tema dictado, dentro del seminario pueden existir muchas tutorías presenciales.

De cada seminario se conoce el código, el nombre y el contenido

De cada laboratorio se conoce el código, el número de computadores y el SW instalado

De cada profesor se conoce el código, el nombre y el título académico.

De cada grupo se conoce el Código y el número de estudiantes.

- Crear el MER ( EER ) , el MR.

**EJERCICIO2:**

**Dado el siguiente Modelo Relacional, reconstruir el Diagrama Entidad Relación.**

EMPLEADOS = (CodEmple,Nombre,Dirección,Telf,Cargo,CodDpto).

DEPARTAMENTO=(Coddpto,Nombre,Ubicación,Piso,Codedif,CodEmpresa).

EDIFICIO=(CodEdif,Nombre,Dirección,Numero de pisos)

EMPRESA=(CodEmpresa,Nombre,Tipo)

EDIF\_EMPRE=(CodEmpresa,Codedif,CostoArrendamiento).

OFICINAS=(CodOfi,Nombre,Ubicación,CodEdif,CodEmpresa)

EMPLE\_OFIC=(CodEmple,CodOfi,Horario).

CLIENTE=(CodCli,Nombre,Dirección,Telf)

EMPLE\_CLI=(CodEmp,CodCli,Tiposervicio)

### EJERCICIO 3

En un supermercado se necesita llevar el control de las compras a proveedores y las ventas a clientes de los diferentes productos.

De cada cliente se conoce el RUC, Nombre y Dirección

De cada distribuidor se conoce el Código, Nombre y Tipo de Distribuidor

De cada producto se conoce el Código, Nombre, Descripción, Unidad de medida y Precio Unitario. Además es conocido que un mismo producto tiene diferentes presentaciones, por ejemplo: el detergente Deja tiene presentaciones de 500g, 200g, 100g, o por ejemplo el aceite tiene presentaciones de 5 litros, 2 litros, 1 litro, ½ litro, en diferentes marcas y de diferentes distribuidores y se vende a diferentes clientes. Establecer el escenario para las condiciones de diseño y crear la Base de Datos.

## CAPITULO III

### NORMALIZACION

#### **Definición.**

La normalización es una técnica que se utiliza para crear relaciones lógicas apropiadas entre tablas de una base de datos. La normalización se adoptó porque el viejo estilo de poner todos los datos en un solo lugar, como un archivo o una tabla de la base de datos, era ineficiente y conducía a errores de lógica cuando se trataba de manipular los datos.

El proceso de normalización parte de las formas normales definidas por Edgar Frank Codd (1970) creador de las bases de datos relacionales. Primeramente, Codd formuló las tres primeras formas normales (1FN, 2FN, 3FN); posteriormente, unas anomalías detectadas forzaron a crear una forma normal más completa que la 3FN, es la FNBC (forma normal de Boyce y Codd), después Fagin definió la 4FN y 5FN.

La normalización es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener.

Los seres humanos tenemos la tendencia de simplificar las cosas al máximo. Lo hacemos con casi todo, desde los animales hasta con los automóviles. Vemos una imagen de gran tamaño y la hacemos más simple agrupando cosas similares juntas. Las guías que la normalización provee crean el marco de referencia para simplificar una estructura de datos compleja.

La normalización se utiliza para mejorar el esquema lógico, de modo que satisfaga ciertas restricciones que eviten la duplicidad de datos.

## Objetivos de la Normalización

Minimizar la redundancia

Minimizar el mantenimiento de datos

Minimizar el impacto de futuros cambios (anomalías de actualización y anomalías de borrado) de de datos, e ingreso de información (anomalías de inserción) .

## Ventajas de la Normalización

Evita anomalías en inserciones, modificaciones y borrados.

Mejora la independencia de datos.

No establece restricciones artificiales en la estructura de los datos.

Facilidad de uso.

Flexibilidad.

Precisión.

Seguridad.

Facilidad de implementación.

Independencia de datos.

Claridad.

Facilidad de gestión.

Mínima redundancia.

Máximo rendimiento de las aplicaciones.

Existen 5 Formas Normales:

- Primera Forma Normal (1FN)
- Segunda Forma Normal (2FN)
- Tercera Forma Normal (3FN)
- Forma Normal de Boyce Codd(FNBC)
- Cuarta Forma Normal (4FN)
- Quinta Forma Normal (5FN)

Cada una de estas formas tiene sus propias reglas.

En general, las primeras tres formas normales son suficientes para cubrir las necesidades de la mayoría de las bases de datos.

Antes de iniciar con la teoría referente a normalización es necesario conocer los conceptos de Dependencias Funcionales, los mismos que se utilizarán en la teoría de Normalización.

## Dependencias Funcionales

Codd introdujo el concepto de dependencia funcional como:

***"Dados dos atributos A y B de una relación R, se dice que B es funcionalmente dependiente de A, si para cada valor de A existe un valor de B, y sólo uno, asociado con él".***

En otros términos, se puede decir que si dos tuplas de una relación R tienen el mismo valor en el atributo A deben tener el mismo valor en el atributo B. O dicho de otro modo, si conocemos el valor de A podemos conocer el valor de B.

**Definición.** Un atributo B de un esquema de relación R depende funcionalmente de un atributo A de R, si y sólo si, cada valor de A está asociado con un único valor de B. Es decir, dado un valor de A queda unívocamente determinado el valor de B. Se dice que B depende funcionalmente de A, y que A determina funcionalmente a B. Tanto A como B pueden ser atributos simples o compuestos

Suponga que tiene  $R = \{A_1, A_2, A_3, \dots, A_n\}$ , R es una relación y A es un conjunto de atributos. Sea X, Y subconjuntos de A.

Notación DF:  $X \rightarrow Y$

Se lee: X determina o implica Y

Y depende funcionalmente de X

Si y sólo si cada valor de X tiene asociado en todo momento un único valor de Y

Donde: X es el determinante

Y es el implicado

X e Y se les dice Descriptores.

Un determinante es un conjunto del que depende funcionalmente otro conjunto de atributos llamado implicado.

**Ejemplo:**

CI\_estudiante  $\rightarrow$  Nombre\_estudiante

Las CI del estudiante determina el nombre del estudiante.

Dos descriptores son equivalentes cuando  $X \rightarrow Y \wedge Y \rightarrow X$ .

**Ejemplo:**

CI\_estudiante  $\rightarrow$  Num\_Carnet\_estudiante y

Num\_Carnet\_estudiante  $\rightarrow$  CI\_estudiante

Notación:

$X \rightarrow Y$  ó bien,  $X_1, \dots, X_n \rightarrow Y_1, \dots, Y_m$   $n, m \geq 1$

### **Axiomas de Armstrong**

Conjunto de dependencias, que permite encontrar otras dependencias implicadas por ellas, las cuales son consistentes y completas.

Los axiomas de Armstrong son consistentes y completos

Los Axiomas de Armstrong son más bien reglas de inferencia.

Estas reglas permiten deducir todas las dependencias funcionales que tienen lugar a un conjunto dado de atributos que se dan a partir del conocimiento del problema.

### **Reflexividad**

Los valores de X y Y que son un conjunto de atributos Y están incluidos o son iguales a los conjuntos de atributos X, entonces se cumplen que Y depende funcionalmente de X.



Es decir que matematicamente es:

$$(Y \rightarrow X) \rightarrow (X \rightarrow Y)$$

Por ejemplo

$$(\text{Cod\_Pais} \rightarrow \text{Cod\_Cap}) \rightarrow (\text{Cod\_Cap} \rightarrow \text{Cod\_Pais})$$

Siendo Cod\_Pais el código del Pais y Cod\_Cap el código de la Capital.

### Aumentatividad

Si el conjunto de atributos Y depende funcionalmente de X, entonces dicha dependencia se mantiene aunque se añada un atributo a ambos conjuntos. Es decir, si:

$$(X \rightarrow Y) \rightarrow (X.Z \rightarrow Y.Z)$$

Por ejemplo si:

$$ci \rightarrow \text{nombre}$$

$$ci, \text{dirección} \rightarrow \text{nombre}, \text{dirección}$$

Si con ci se determina el nombre de una persona, entonces con ci más la dirección también se determina el nombre o su dirección.

### Transitividad

Si Y depende funcionalmente de X y Z depende funcionalmente de X, entonces se verifica que Z depende funcionalmente de X. por lo tanto, si:

$$(X \rightarrow Y) \text{ y } (Y \rightarrow Z) \rightarrow (X \rightarrow Z)$$

Por ejemplo:

$$\text{FechaDeNacimiento} \rightarrow \text{Edad}$$

$$\text{Edad} \rightarrow \text{Puede\_Conducir}$$

$$\text{FechaDeNacimiento} \rightarrow \text{Edad} \rightarrow \text{Puede\_Conducir}$$

Entonces tenemos que FechaDeNacimiento determina a Edad y la Edad determina a Conducir, indirectamente. Por tanto podemos saber a través de FechaDeNacimiento si Puede\_Conducir.

## **Aditividad**

Si Y depende funcionalmente de X y también se cumple que Z depende funcionalmente de X, eso implica que Y y Z dependen funcionalmente de X. por tanto:

**$(X \rightarrow Y) \text{ y } (X \rightarrow Z) \text{ entonces } (X \rightarrow YZ)$**

## **PseudoTransitividad**

Si se cumple que:

**$(X \rightarrow Y) \text{ y } (WY \rightarrow Z) \text{ entonces } (WX \rightarrow Z)$**

La demostración de esta regla se la obtiene aplicando:

**$(X \rightarrow Y)$ , por aumentación  $(WX \rightarrow WY) (WY \rightarrow Z)$  y ahora tras aplicar la transitividad  $(WX \rightarrow Z)$**

Por ejemplo:

**$(\text{Nombre} \rightarrow \text{CI}) \text{ y } (\text{CI.Nom\_Empresa} \rightarrow \text{Sueldo}) \rightarrow$   
 **$(\text{Nombre.Nom\_Empresa} \rightarrow \text{Sueldo})$****

## **Descomposición**

Si el conjunto de atributos Y depende funcionalmente de X y también se cumple que los valores del conjunto de atributos Z están incluidos en los valores de Y, entonces se tiene que cumplir que Z depende funcionalmente de X. por lo tanto:

**$(X \rightarrow Y) \text{ y } (Z \subseteq Y) \text{ entonces } (X \rightarrow Z).$**

Se puede demostrar aplicando:

**$(X \rightarrow Y)$**

**$(Z \subseteq Y)$ , por reflexividad  $(Y \rightarrow Z)$**

Este conjunto de reglas nos permite abordar y resolver una serie de problemas fundamentales que luego nos conducirán al establecimiento de algoritmos de diseño que sean sencillos y fiables.

Estos problemas fundamentales, son :

- Cierre de un conjunto de dependencias.
- Equivalencia lógica de esquemas .
- Deducción de dependencias .
- Cierre de un descriptor respecto de un conjunto de dependencias.
- Cálculo de las claves de un esquema

## **Tipos de Dependencias Funcionales**

### **Dependencia Funcional Completa**

Un atributo B de R tiene dependencia funcional completa de un atributo A de una relación R, si tiene dependencia funcional de A pero no tiene dependencia funcional de ningún subconjunto de A.

**Formato:**

**$A \twoheadrightarrow B$**

**“A determina funcionalmente a B”**

### **Ejemplo**

#### **CI.Empleado $\rightarrow$ Sueldo**

La cedula del empleado determina el sueldo.

El sueldo no depende de ningún subconjunto de la cédula que es la clave primaria.

### **Dependencia Funcional Multivaluada**

“Sea A,B,C tres subconjuntos distintos de atributos de una tabla T, se dice que A tiene una dependencia Multivaluada con B, que A multidetermina a B, o que B depende multivaluadamente de A.”. Existen casos de relaciones en los que un atributo puede determinar a otro restringiendo su rango de

valores validos. A este tipo de dependencias se les conoce como dependencias multivaluadas.

Formato:

$$A \twoheadrightarrow B$$

**“A multidetermina a B”**

### **Dependencia Funcional Transitiva**

Sean A, B y C atributos de un esquema de relación R; si C tiene dependencia funcional de B y B tiene dependencia funcional de A, entonces C tiene dependencia funcional transitiva de A.

Formato:

$$A \rightarrow B, B \rightarrow C$$

**“A determina B y si B determina C entonces A determina a C”**

### **Ejemplo:**

La tabla

Ciudades(ciudad, población, superficie, renta, país, continente)

ciudad  $\rightarrow$  país, país  $\rightarrow$  continente.

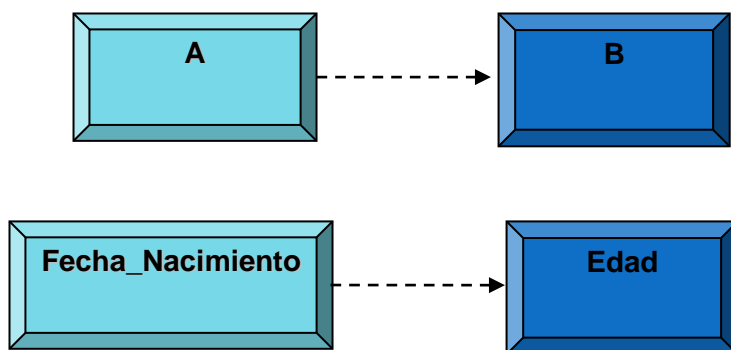
ciudad  $\rightarrow$  continente.

Además, país  $\rightarrow$  |ciudad. Es decir, cada ciudad pertenece a un país y cada país a un continente, pero en cada país puede haber muchas ciudades. En este caso continente tiene una dependencia funcional transitiva con respecto a ciudad, a través de país. Es decir, cada ciudad está en un país, pero también en un continente.

## Diagrama de Dependencias Funcionales

Los diagramas de dependencia son gráficos que representan el contexto semántico observado en un determinado universo, donde los nodos son atributos y los arcos representan dependencias entre nodos. Normalmente se representan dependencias que van de un nodo o a un solo atributo.

Es la forma grafica de representar las dependencias funcionales, así:



Comprender la Dependencia Funcional es parte importante para entender la semántica de los datos.

### Ejemplo

Para la entidad Proveedor conformada por:

**Proveedor (Numero\_Prov, Nombre\_Prov, Tipo\_Prov, Ciudad).**



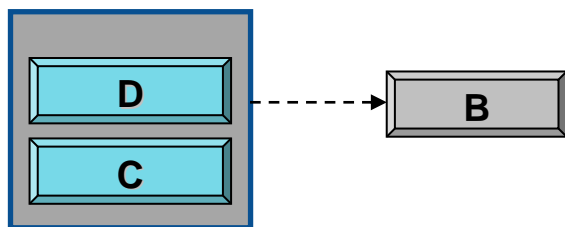
## Diagrama de Dependencia Funcional Completa

Dado los atributos A, B y C de una relación R. Si, C depende de ambos atributos A y B y lo denotamos así:  $[A, B] \twoheadrightarrow C$ , y si además: C, no depende de A ni tampoco depende de B exclusivamente.

C, tiene una dependencia completa de  $[A, B]$  y lo denotamos:

$[A, B] \twoheadrightarrow C$  (también se dice: C tiene dependencia total de  $[A, B]$ ).

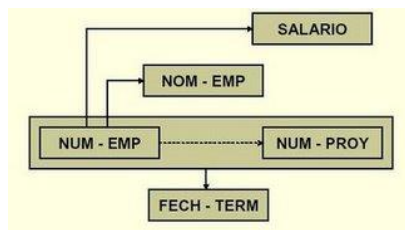
**NOTA:** Si existiera una dependencia funcional completa en una relación R, todos los demás atributos de R que no son llave primaria, deberán tener la misma dependencia completa de los mismos atributos, de lo contrario se presentarían ineficiencias y anomalías en R, así



En el ejemplo B, tiene una dependencia completa de  $[C,D] \twoheadrightarrow B$

Se dice: B tiene dependencia total de  $[C,D]$

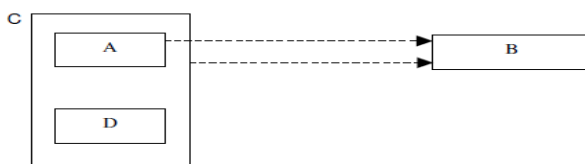
### Segundo Ejemplo.



En el ejemplo Fecha de Terminación del Proyecto(Fech-Term), Tienet una Dependencia Funcional Completa de la Clave(Num\_Emp + Num\_Proj), y no depende de un subconjunto de ella.

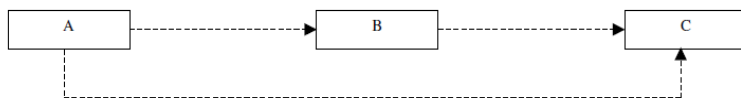
### Diagrama de Dependencia Funcional Parcial

Un atributo B de R tiene dependencia funcional parcial de un atributo C de R, si tiene dependencia funcional de C y además tiene dependencia funcional de un subconjunto propio A de C.



### Diagrama de Dependencia Funcional Transitiva

Sean A, B y C atributos de un esquema de relación R; si C tiene dependencia funcional de B y B tiene dependencia funcional de A, entonces C tiene dependencia funcional transitiva de A.



**NOTA:** La dependencia transitiva no es buena en una relación o tabla de base de datos, porque evidencia la existencia de atributos que no dependen únicamente de la llave primaria sino de otros atributos, ocasionando lo que se llama: anomalía en los procesos de actualización, inserción o eliminación. Una vez definidos claramente los conceptos de Dependencia Funcional, se puede analizar la Normalización.

## NORMALIZACION

### Formas Normales

#### Primera Forma Normal (1FN)

La regla de la Primera Forma Normal establece que las columnas repetidas deben eliminarse y colocarse en tablas separadas.

Poner la base de datos en la Primera Forma Normal resuelve el problema de los encabezados de columna múltiples. Muy a menudo, los diseñadores de bases de datos inexpertos harán algo similar a la tabla no normalizada. Una y otra vez, crearán columnas que representen los mismos datos. La normalización ayuda a clarificar la base de datos y a organizarla en partes más pequeñas y más fáciles de entender. En lugar de tener que entender una tabla gigantesca y monolítica que tiene muchos diferentes aspectos, sólo tenemos que entender los objetos pequeños y más tangibles, así como las relaciones que guardan con otros objetos también pequeños.

Si una relación no está en 1FN, hay que eliminar de ella los grupos repetitivos. Un grupo repetitivo será el atributo o grupo de atributos que tiene múltiples valores para cada tupla de la relación.

Por Ejemplo: En la siguiente tabla se encuentran los datos del alquiler de diferentes películas. Esta tabla se encuentra en Forma Normal 0. (Totalmente Desnormalizada).



PELICULAS : Tabla

Cod_Pel	Tit_Pel	Pro_Pel	Tip_Pel	Idio_Pel	Ci_Cli	Nom_Cli	Fech_Dev	Cod_R	Nom_Ren	Fech_Ren
P001	TITANIC	UNIVERSAL	ROMANTICA	INGLES	1801	DIANA MARTINEZ	30/06/2009	1501	JUAN LOZADA	28/06/2009
P002	ROMEO Y JULIETA	METRO GOLDEN MAYER	ROMANTICA	INGLES	1801	DIANA MARTINEZ	29/06/2009	1501	JUAN LOZADA	26/06/2009
P003	DEPREDADOR	UNIVERSAL	FICCION	ESPAÑOL	1802	JUAN PEREZ	30/06/2009	1502	ANA LOPEZ	29/06/2009
P004	TRASFORMES	CENTURY	FICCION	SUBIO A INGLES	1803	MARIA JIMENEZ	28/06/2009	1501	JUAN LOZADA	25/06/2009
P005	CARS	WALT DISNEY	INFANTIL	ESPAÑOL	1803	MARIA JIMENEZ	28/06/2009	1502	ANA LOPEZ	27/06/2009
P006	EL TODOPODEROSO	DREAMS WORKS	COMEDIA	SUBIO A ESPAÑOL	1804	RICARDO ALVAREZ	29/06/2009	1502	ANA LOPEZ	27/06/2009

Registro: 7 de 7

Para que esta tabla pase a 1FN se debe eliminar los grupos repetitivos de las tablas individuales. Es decir la columna que contiene múltiples valores, que es Nom\_Cli contiene el nombre y el apellido del cliente, esta columna se debe dividir en columnas individuales que guarden valores indivisibles, como Nom\_Cli, y Ape\_Cli. Se puede hacer lo mismo con la columna Nom\_Ren.

La tabla en 1FN será:

PELICULAS\_1F : Tabla

Cod_P	Tit_Pel	Pro_Pel	Tip_Pel	Idio_Pel	Ci_Cli	Nom_Cli	Ape_Cli	Fech_Dev	Cod_R	Nom	Ape_Ren	Fech_Ren
P001	TITANIC	UNIVERSAL	ROMANTICA	INGLES	1801	DIANA	MARTINEZ	30/06/2009	1501	JUAN	LOZADA	28/06/2009
P002	ROMENO Y JULIETA	METRO GOLDEN MAYER	ROMANTICA	INGLES	1801	DIANA	MARTINEZ	29/06/2009	1501	JUAN	LOZADA	26/06/2009
P003	DEPREDADOR	UNIVERSAL	FICCION	ESPAÑOL	1802	JUAN	PEREZ	30/06/2009	1502	ANA	LOPEZ	29/06/2009
P004	TRANSFORMES	CENTURY	FICCION	SUBIDO A ESPAÑOL	1803	MARIA	JIMENEZ	28/06/2009	1501	JUAN	LOZADA	25/06/2009
P005	CARS	WALT DISNEY	INFANTIL	ESPAÑOL	1803	MARIA	JIMENEZ	28/06/2009	1502	ANA	LOPEZ	27/06/2009
P006	EL TODO PODEROSO	DREAM WORKS	COMEDIA	SUBIDO A ESPAÑOL	1804	RICARDO	ALVAREZ	29/06/2009	1502	ANA	LOPEZ	27/06/2009

Registro: 7 de 7

## Segunda Forma Normal (2FN)

Una tabla se dice que está en **2FN** si y sólo si cumple dos condiciones:

- 1) Se encuentra en 1 FN

- 2) Todo atributo secundario (aquellos que no pertenecen a la clave principal, los que se encuentran fuera de la caja) depende totalmente (tiene una dependencia funcional total) de la clave completa y, por tanto, no de una parte de ella.

Esta forma normal sólo se considera si la clave principal es compuesta y, por tanto, está formada por varios atributos. Si la clave principal de la tabla está formada por un único atributo, entonces la tabla ya se encuentra en 2FN.

Si una tabla T tiene como atributos A, B, C, D y la clave es A.B cumpliéndose las dependencias:

A.B —> C

B —> D

Se observa que la tabla no se encuentra en 2FN puesto que el atributo D no tiene una dependencia funcional total con la clave entera A.B, sino con una parte de la clave (B).

Para convertir una tabla que no está en segunda forma normal a 2FN, se realiza una proyección y se crea:

- 1) Una tabla con la clave y todas sus dependencias totales con los atributos secundarios afectados
- 2) Otra tabla con la parte de la clave que tiene dependencias, junto con los atributos secundarios implicados

La clave de la nueva tabla será la antigua parte de la clave.

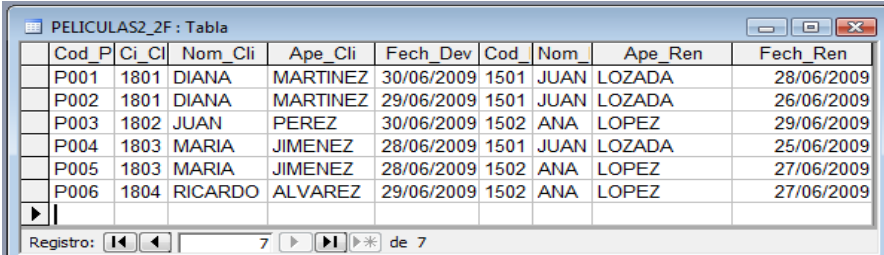
En nuestro ejemplo, tendremos que se va a crear una tabla con los datos de las Películas:

## PELICULAS

Cod_Pel	Tit_Pel	Pro_Pel	Tip_Pel	Idio_Pel
P001	TITANIC	UNIVERSAL	ROMANTICA	INGLES
P002	ROMEO Y JULIETA	METRO GOLDEN MAYER	ROMANTICA	INGLES
P003	DEPREDADOR	UNIVERSAL	FICCION	ESPAÑOL
P004	TRASFORMES	CENTURY	FICCION	SUBIO A INGLES
P005	CARS	WALT DISNEY	INFANTIL	ESPAÑOL
P006	EL TODOPODEROSO	DREAMS WORKS	COMEDIA	SUBIO A ESPAÑOL
*				

Registro: 1 de 6

Otra tabla con los datos de los préstamos de esas películas.



Cod_P	Ci_Ci	Nom_Cli	Ape_Cli	Fech_Dev	Cod_	Nom_	Ape_Ren	Fech_Ren
P001	1801	DIANA	MARTINEZ	30/06/2009	1501	JUAN	LOZADA	28/06/2009
P002	1801	DIANA	MARTINEZ	29/06/2009	1501	JUAN	LOZADA	26/06/2009
P003	1802	JUAN	PEREZ	30/06/2009	1502	ANA	LOPEZ	29/06/2009
P004	1803	MARIA	JIMENEZ	28/06/2009	1501	JUAN	LOZADA	25/06/2009
P005	1803	MARIA	JIMENEZ	28/06/2009	1502	ANA	LOPEZ	27/06/2009
P006	1804	RICARDO	ALVAREZ	29/06/2009	1502	ANA	LOPEZ	27/06/2009

Registro: 7 de 7

### Tercera Forma Normal (3FN)

Una tabla se dice que está en tercera forma normal si y sólo si se cumplen dos condiciones:

- 1) Se encuentra en 2FN
- 2) No existen atributos no primarios que son transitivamente dependientes de cada posible clave de la tabla.

Esto quiere decir que un atributo secundario sólo se debe conocer a través de la clave principal o claves secundarias de la tabla y no por medio de otro atributo no primario.

Por tanto, para pasar una tabla que no cumple la tercera forma normal a 3FN, se realiza una proyección y se genera:

- 1) Una tabla con la clave y todos los atributos no primarios que no son transitivos
- 2) Otra tabla con los atributos transitivos y el atributo no primario (que será la clave de la nueva tabla) por medio del cual mantienen la transitividad

Lógicamente, en la primera tabla T1, el atributo C es clave ajena con respecto de T2 y de ese modo todos los atributos quedan relacionados entre sí. Es lo que se denomina interrelación entre la tabla T1 y T2.

En nuestro ejemplo tendremos que se van a generar las siguientes tablas:

Tabla Clientes para contener los datos de la persona que rento la película agregando en su cedula de identidad como clave.

### Clientes

Ci_Cli	Nom_Cli	Ape_Cli
1801	DIANA	MARTINEZ
1802	JUAN	PEREZ
1803	MARIA	JIMENEZ
1804	RICARDO	ALVAREZ

Tabla Rentadores para contener los datos de la persona que renta la película agregando un su cedula de identidad como clave.

### Rentadores

Ci_Ren	Non_Ren	Ape_Ren
1501	JUAN	LOZADA
1502	ANA	LOPEZ

Tabla Películas con los datos de las películas, con el código de la película como clave.

### Películas

PELICULAS_2F : Tabla				
Cod_Pel	Tit_Pel	Pro_Pel	Tip_Pel	Idio_Pel
P001	TITANIC	UNIVERSAL	ROMANTICA	INGLES
P002	ROMEO Y JULIETA	METRO GOLDEN MAYER	ROMANTICA	INGLES
P003	DEPREDADOR	UNIVERSAL	FICCION	ESPAÑOL
P004	TRASFORMES	CENTURY	FICCION	SUBIO A INGLES
P005	CARS	WALT DISNEY	INFANTIL	ESPAÑOL
P006	EL TODOPODEROSO	DREAMS WORKS	COMEDIA	SUBIO A ESPAÑOL
*				
Registro: 1 de 6				

Tabla Prestamos con el código de la película, cedula del cliente y la cedula de la persona que rento, la fecha devolución y fecha de rento

### Prestamos

Cod_Pel ▾	Ci_Cli ▾	Ci_Ren ▾	Fech_Dev ▾	Fech_Ren ▾
P001	1801	1501	30/06/2009	28/06/2009
P002	1801	1501	29/06/2009	26/06/2009
P003	1802	1502	30/06/2009	29/06/2009
P004	1803	1501	28/06/2009	25/06/2009
P005	1803	1502	28/06/2009	27/06/2009
P006	1804	1502	29/06/2009	27/06/2009

Hasta esta 3FN se considera un esquema de base de datos normalizado.

### Forma Normal de Boyce Codd (FNBC)

Tras la creación de la 3FN se observó posteriormente que se encontraban algunas anomalías que no eran abordadas. Son casos de tablas que estando en 3FN mantienen una dependencia de un atributo secundario con parte de la clave. Para poder manejar esa dependencia en las aplicaciones es imprescindible manejar una gran cantidad de registros innecesarios (aquellos donde se mantiene fija la parte de la clave que depende y va variando el resto de la clave)

La nueva definición se debe a Boyce y Codd:

"Una tabla T está en FNBC si y sólo si las únicas DF elementales son aquellas en las que la clave principal (y claves secundarias) determinan un atributo".

La definición engloba la 3FN puesto que las dependencias transitivas existen por medio de atributos secundarios que no eran clave.

Pero esta definición realmente se creó para evitar los casos anómalos que no se evitaban con la 3FN y que aparecen cuando a partir de un atributo no primario se conoce una parte de la clave.

Si la clave está formada por un sólo atributo, la tabla está en FNBC (si ya estaba en 3FN) como sucedía con la 2FN.

Por lo tanto, para que una tabla que está en 3FN y no cumple la norma de Boyce-Codd se encuentre en FNBC se realiza una proyección procediendo de la siguiente manera:

1. Se crea una tabla con la parte de la clave que es independiente (A) y todos los atributos no primarios (C)
2. Se crea otra tabla con la parte de la clave restante y el atributo secundario del que depende, y será éste último la clave de la nueva tabla

### **Cuarta Forma Normal (4FN)**

La 4FN la generó Fagin tras el teorema que demostró y que dice:

"Una tabla T con atributos A, B y C se puede descomponer sin pérdidas en sus dos proyecciones T1(A,B) y T2(A,C) si y solo si la Dependencia Multivaluada  $A \twoheadrightarrow B \mid C$  se cumple en T"

De ese modo, se define la 4FN de la siguiente forma:

Una tabla T se dice que está en 4FN si cumple dos condiciones:

- 1) Está en FNBC.
- 2) Las únicas Dependencias Multivaluadas existentes son las Dependencias Funcionales de la clave con los atributos secundarios.

### **Quinta Forma Normal (5FN)**

Para que una tabla se encuentra en 5FN se deben cumplir dos condiciones:

- 1) Se encuentra en 4FN
- 2) Toda Dependencia de Join viene implicada por las claves (principal o secundarias) de la tabla.

Es decir, que la tabla estará en 5FN si es posible generar unas proyecciones y al realizar su join, los atributos comunes que realizan la operación (atributos de join) están formados por claves (principal o secundarias) de la tabla.

## Ejercicios Guía Resueltos

Dado R (H, I, K, L, M, O) y el conjunto de dependencias funcionales  $F = \{H \rightarrow IO, O \rightarrow HO, KM \rightarrow L, L \rightarrow MK, M \rightarrow K, HK \rightarrow M\}$ , hallar todas las claves candidatas

### Respuesta:

$$(H)^+ = \{H, I, O\}$$

$$(I)^+ = \{I\}$$

$$(K)^+ = \{K\}$$

$$(L)^+ = \{L, M, K\}$$

$$(M)^+ = \{M, K, L\}$$

$$(O)^+ = \{O, H, I\}$$

Como en  $(H)^+ \cup (I)^+ \cup (O)^+$  no están ni K, ni L, ni M y en  $(K)^+ \cup (L)^+ \cup (M)^+$  no están ni H, ni I, ni O, la clave **no** va a estar formada por la combinación de los atributos H, I, O, ni por la combinación de los atributos K, L, M.

$$(HK)^+ = \{H, K, I, O, M, L\} \Rightarrow HK \text{ es Clave Candidata}$$

$$(HL)^+ = \{H, L, I, O, M, L\} \Rightarrow HL \text{ es Clave Candidata}$$

$$(HM)^+ = \{H, M, I, O, K, L\} \Rightarrow HM \text{ es Clave Candidata}$$

$$(IK)^+ = \{I, K\}$$

$$(IL)^+ = \{I, L, M, K\}$$

$$(IM)^+ = \{I, M, K, L\}$$

$$(OK)^+ = \{O, K, H, I, M, L\} \Rightarrow OK \text{ es Clave Candidata}$$

$$(OL)^+ = \{O, L, H, I, M, K\} \Rightarrow OL \text{ es Clave Candidata}$$

$$(OM)^+ = \{O, M, H, I, K, L\} \Rightarrow OM \text{ es Clave Candidata}$$

## Ejercicio 2

Dado R (A, B, C, D, E) y el conjunto de dependencias funcionales  $F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$ , probar, utilizando los axiomas, que las siguientes dependencias pertenecen a  $F^+$ .

a)  $E \rightarrow B$

b)  $EA \rightarrow CA$

c)  $AD \rightarrow E$

d)  $AC \rightarrow E$

### Respuestas:

(a)

Si  $E \rightarrow A$  y  $A \rightarrow BC$ , por transitividad  $E \rightarrow BC$

Por descomposición,  **$E \rightarrow B$**  y  $E \rightarrow C$

(b)

Si  $E \rightarrow A$  y  $A \rightarrow BC$ , por transitividad  $E \rightarrow BC$

Por descomposición,  **$E \rightarrow B$**  y  $E \rightarrow C$

Por aumentatividad,  **$EA \rightarrow CA$**

(c)

Si  $A \rightarrow BC$ , por aumentatividad,  $AD \rightarrow BCD$

Por descomposición,  $AD \rightarrow CD$  y  $AD \rightarrow B$

Si  $AD \rightarrow CD$  y  $CD \rightarrow E$ , por transitividad  **$AD \rightarrow E$**

(d)

Si  $A \rightarrow BC$ , por aumentatividad,  $AC \rightarrow BC$

Por descomposición,  $AC \rightarrow B$  y  $AC \rightarrow C$

Si  $AC \rightarrow B$  y  $B \rightarrow D$ , por transitividad  $AC \rightarrow D$

Si  $AC \rightarrow D$  y  $AC \rightarrow C$ , por unión  $AC \rightarrow DC$

Si  $AC \rightarrow DC$  y  $CD \rightarrow E$ , por transitividad,  **$AC \rightarrow E$**

### Ejercicio 3

Probar, utilizando los axiomas para df's, la siguiente regla:

e) De  $X \rightarrow Y$ ,  $X \text{ÍV}$ ,  $W \text{ÍY}$ ,  $W \rightarrow Z$  inferir  $V \rightarrow Z$ .

### Respuesta:

Por reflexividad, si  $X \text{ÍV}$ , entonces,  $V \rightarrow X$

Por reflexividad, si  $W \text{ÍY}$ , entonces,  $Y \rightarrow W$

Si  $V \rightarrow X$  y  $X \rightarrow Y$ , por transitividad,  $V \rightarrow Y$

Si  $V \rightarrow Y$  y  $Y \rightarrow W$ , por transitividad,  $V \rightarrow W$

Si  $V \rightarrow W$  y  $W \rightarrow Z$ , por transitividad,  **$V \rightarrow Z$**



## Ejercicio 4

Sea el esquema de relación  $R(A, B, C, D, E, F, G, H)$  y el siguiente conjunto de df's  $F = \{ABC \rightarrow E, FD \rightarrow A, AG \rightarrow E, D \rightarrow C, BC \rightarrow F, A \rightarrow H, F \rightarrow D, H \rightarrow G\}$ .

Encontrar un cubrimiento minimal para  $F$ .

### Respuesta:

Primer paso: todas las dependencias de  $F$  deben ser de la forma  $X \rightarrow A$ .

Segundo paso: Eliminar atributos redundantes de los lados izquierdos de cada df.

$$(A)^+ = \{A, H, G, E\}$$

$$(BC)^+ = \{B, C, F, D, A\} \Rightarrow A \text{ es redundante en la df } ABC \rightarrow E$$

$$(F)^+ = \{F, D\} \Rightarrow D \text{ es redundante en la df } FD \rightarrow A$$

$$(A)^+ = \{A, H, G\} \Rightarrow G \text{ es redundante en la df } AG \rightarrow E$$

$$(B)^+ = \{B\}$$

$$(C)^+ = \{C\}$$

Tercer paso: No deben existir dfs redundantes.

$$(BC)^+ = \{B, C, F, A, E\} \Rightarrow BC \rightarrow E \text{ es redundante}$$

## Ejercicio 5

Dado  $R(A, B, C, D, E, F)$  y  $F = \{AB \rightarrow C, C \rightarrow E, E \rightarrow F, F \rightarrow B, C \rightarrow B\}$

1 - Descomponer en 3FN SPI y SPD

2 - Eliminar **F → B**, descomponer en 3FN y decir si la descomposición resultante está en FNBC

### Respuesta:

1 - Las claves candidatas son **ADB, ADC, ADE y ADF**, por lo que el esquema ya está en 3FN.

2 - Si eliminamos **F → B**, **ADE** y **ADF** dejarán de ser claves candidatas.

Al descomponer en 3FN:

$$R_1(ABC); F_1 = \{AB \rightarrow C, C \rightarrow B\}$$

$$R_2(CE); F_2 = \{C \rightarrow E\}$$

$$R_3(EF); F_3 = \{E \rightarrow F\}$$

$R_4(CB); F_4=\{C \rightarrow B\}$

La descomposición estaría en FNBC si cada uno de los lados izquierdos de las DFs de  $F_1$  fueran clave candidata, cosa que no se cumple; por lo que descomponemos  $R_1(ABC)$  para que se cumpla la FNBC.

Al descomponer  $R_1(ABC)$ :

$R_{11}(CB); F_{11}=\{C \rightarrow B\}$

$R_{12}(AC); F_{12}=\{\}$

Con ello, el esquema está en FNBC.

## AutoEvaluación. Ejercicios Propuestos

### Ejercicio 1

Una BD debe contener información concerniente a las ventas de los productos de una cierta compañía (Agentes, Áreas y Productos). Cada Agente es responsable de las ventas en una o más Áreas, cada Área tiene uno o más Agentes como responsables de las ventas en ella. Del mismo modo, cada Agente es responsable de la venta de uno o más Productos y cada Producto tiene uno o más Agentes responsables de su venta. Todos los Productos se venden en todas las Áreas, pero no hay dos Agentes que vendan el mismo producto en la misma Área. Cada Agente vende el mismo conjunto de Productos en todas las Áreas en las que opera, y con independencia del Área tiene establecido un precio mínimo de venta para cada producto.

Se pide determinar las dependencias funcionales y representarlas mediante el correspondiente diagrama.

### Ejercicio 2

Dada la relación R, con la extensión  $r(R)$  que se muestra en la figura,

A B C D

a1 b1 c1 d1

a2 b1 c2 d1

a3 b2 c3 d1

a4 b2 c3 d1

Definir si las siguientes afirmaciones relativas a la variable de relación (no al contenido en este momento) son ciertas, falsas o desconocidas:

a)

$A \rightarrow BCD$

b)

$B \rightarrow A$

c)

$C \text{ no} \rightarrow A$

d)

$BC \text{ no} \rightarrow A$

e)

$BC \rightarrow D$

f)

$D \rightarrow A$

### Ejercicio 3

La Seguridad Social desea conocer los pacientes (CI) que han sido atendidos en sus hospitales (CodHosp) y el doctor (CIDoc) que los atiende. Se supone que un doctor sólo puede atender en un hospital y que, aunque un paciente puede ser atendido en varios hospitales, en cada uno de ellos sólo le atiende un doctor.

Determinar las dependencias funcionales de este supuesto.

### Ejercicio 4

Supongamos que al diseñar una BD se obtienen las cuatro relaciones siguientes:

$R1(\text{nombre\_emp}, \text{direcc\_emp}, \text{edad}, \text{sexo}, \text{nombre\_superv})$

$R2(\text{nombre\_superv}, \text{departamento})$

$R3(\text{nombre\_empl}, \text{departamento})$

$R4(\text{departamento}, \text{num\_tel\_depart}, \text{direcc-depart})$

Una de las relaciones es redundante. Identificarla y explicar las razones de dicha redundancia.

**Ejercicio 5**

Cada despacho de una oficina es identificado por un #despacho y tiene precisamente un teléfono. Cada teléfono tiene su propio #extensión. Hay dos tipos de teléfonos, sólo para llamadas internas (tipo I), y para llamadas externas/internas (tipo E). Los costes de alquiler de extensión dependen únicamente del tipo, teléfonos de tipo I son cargados con la tarifa T1, y los del tipo E con la tarifa T2. La información sobre despachos y teléfonos será almacenada en la relación:

Oficina(#despacho, número\_ocupantes, #extensión, tipo\_teléfono, tarifa)

Haciendo cualquier asunción plausible necesaria se pide:

- Identificar las dependencias funcionales.
- Identificar las dependencias funcionales, pero con la adición de los atributos #empleado y nombre\_emp. Los valores de #empleado identifican empleados individuales. Cada empleado tiene un único nombre y ocupa sólo un despacho.
- Identificar las dependencias funcionales, pero permitiendo varios teléfonos por despacho. Todos los empleados de un despacho comparten todos los teléfonos de dicho despacho.

**Ejercicio 6**

¿Cuáles son las claves de la relación  $R\{X, Y, Z, W, U\}$ , siendo DF el conjunto de dependencias funcionales de la figura.

W

U

X

Y

Z

**Ejercicio 7**

Dado el siguiente esquema de relación  $R\{O, R, U, V, W, X, Y, Z\}$ ; DF con  $DF=\{XY \rightarrow Z, Z \rightarrow U, XYZ \rightarrow V, R \rightarrow X, X \rightarrow R, W \rightarrow O, O \rightarrow W\}$

Determinar las claves de R.

## CAPITULO IV

### ALGEBRA RELACIONAL

#### Definición:

El Algebra relacional es un lenguaje de consulta procedural, partiendo de una o más tablas base se puede realizar sobre ellas una operación de algebra y obtener como resultado otra tabla llamada vista

Una Tabla base es aquella que está almacenada físicamente la BDD.

#### Estudiantes

Nom_Est	Ape_Est	Dir_Est	Prom_Est



Selección  
Proyección

Dir_Est

## Operaciones del Álgebra Relacional

Las operaciones que se aplican sobre la tabla base se conocen como operaciones del álgebra relacional; los mismos que son soportados en algunos casos por los SGBD's.

En general el álgebra relacional es el conjunto de operaciones que se ejecutan sobre una o varias tablas base y dan como resultado visiones de los datos o tablas vistas.

El álgebra relacional es un lenguaje formal con una serie de operadores que trabajan sobre una o varias relaciones para obtener otra relación resultado, sin que cambien las relaciones originales. Tanto los operandos como los resultados son relaciones, por lo que la salida de una operación puede ser la entrada de otra operación. Esto permite anidar expresiones del álgebra, del mismo modo que se pueden anidar las expresiones aritméticas. A esta propiedad se le denomina *clausura*: las relaciones son cerradas bajo el álgebra, del mismo modo que los números son cerrados bajo las operaciones aritméticas.

En esta clase se presentan los operadores del álgebra relacional de un modo informal.

Las definiciones formales serán vistas en las siguientes clases, de acuerdo a la bibliografía. Primero se describen los nueve operadores originalmente propuestos por Codd y después se estudian algunos operadores adicionales que añaden potencia al lenguaje.

De los nueve operadores, sólo hay cinco que son fundamentales: *restricción*, *proyección*, *producto cartesiano*, *unión* y *diferencia*, que permiten realizar la mayoría de las operaciones de obtención de datos. Los operadores no fundamentales son la *juntura* (*join*), la *intersección* y la *división*, que se pueden expresar a partir de los cinco operadores fundamentales.

Las Operaciones del Álgebra Relacional son:

#### OPERADORES PRIMITIVOS

- PROYECCIÓN ( $\pi$ )
- SELECCIÓN ( $\sigma$ )
- PRODUCTO CARTESIANO ( $\times$ )
- UNIÓN ( $\cup$ )
- DIFERENCIA ( $-$ )

#### OPERADORES DERIVADOS

- INTERSECCIÓN ( $\cap$ )
- JOIN ( $\theta$ )
- DIVISIÓN ( $:$ )
- ASIGNACION

### Operadores Primitivos

#### Proyección

La Proyección constituye especificar un subconjunto vertical de columnas de una tabla, es decir, la proyección muestra los atributos específicos que se desean visualizar de una tabla

La proyección se señala con la letra griega pi( $\pi$ ). Como subíndice de pi se coloca una lista de todos los atributos que se desean que aparezcan en el resultado, seguida del nombre de la tabla de la cuál se obtendrán los datos

Matemáticamente la proyección se representa así:

$$\pi_{(columna1, columna2, ..., columna n)} Tabla$$

**Ejemplo:**

Proyectar las columnas **ci\_est**, **nom\_est**, **prom\_est** sobre la tabla **Estudiantes**.

$$\pi_{(ci\_est, nom\_est, prom\_est)}^{Estudiantes}$$

**Estudiantes**

Nom_Est	Ape_Est	Dir_Est	Prom_Est



Proyección

ci_est	nom_est	prom_est

En SQL la proyección se indica desde la sentencia **select** hasta la sentencia **from**.

**SELECT ci\_est, nom\_est, prom\_est**  
**FROM Estudiantes;**



## Selección (RESTRICCIÓN)

Seleccionar o restringir constituye especificar una condición que devuelva como resultado un conjunto de las filas o tuples que cumplan con la condición.

Es decir la selección crea un filtro de filas horizontales formado por las filas que cumplen una condición.

Matemáticamente la selección se especifica así:

$$\sigma_{(condición)} Tabla$$

### Ejemplo:

**Seleccionar de la tabla Estudiantes, las tuplas que cumplan con tener un promedio mayor o igual a 7.**

$$\sigma_{(prom\_est \geq 7)} Estudiantes$$

En SQL la restricción o selección se especifica en la cláusula where de la sentencia SELECT así:

**SELECT \***

**FROM Estudiantes**

**WHERE prom\_est >= 7;**

En Aplicaciones prácticas la sentencia SQL especifica tanto columnas como filas que cumplan una determinada condición, es decir combina la proyección con la selección, así:

**SELECT ci\_est, nom\_est, ape\_est, prom\_est**

**FROM Estudiantes**

**WHERE prom\_est >= 7;**

## Producto Cartesiano

El producto de una TablaA X TablaB.

Constituyen el conjunto de todas las tuplas concatenadas con todas las tuplas de la tabla b, en donde no es necesario que a y b tengan el mismo número de columnas.

### Ejemplo:

Realizar un producto cartesiano entre la tabla Estudiantes con sus columnas cedula, nombre y apellido y la tabla Semestre con la columna nombre del semestre

#### Estudiantes

	Nom_est	Apell_Est	Cod_Sem_Pert
	DANIEL	YANEZ	S01
	JAZMINA	SILVA	S01
	DIEGO	RUIZ	S03
	FREDY	ARROBA	S03

#### Semestres

	Cod_sem	Nom_sem
+	S01	CUARTO
+	S02	QUINTO
+	S03	SEXTO

En SQL la sentencia correspondiente será:

```
SELECT ESTUDIANTES.CI_Est,  
       ESTUDIANTES.Nom_Est,  
       ESTUDIANTES.Apell_Est,  
       ESTUDIANTES.Cod_Sem_Pert  
       SEMESTRES.Cod_Sem,  
       SEMESTRES.Nom_Sem,  
FROM ESTUDIANTES,SEMESTRES;
```

La tabla resultante tendrá la cantidad de columnas igual a la sumatoria de las columnas de la tabla Estudiantes, mas las columnas de la tabla Semestres, y la cantidad de filas igual a la multiplicación del número de filas de la tabla Estudiante por el número de filas de la tabla Semestres, así:

CI_EST	NOM_EST	APELL_EST	COD_SEM
1801	DANIEL	YANEZ	S01
1801	DANIEL	YANEZ	S01
1801	DANIEL	YANEZ	S01
1802	JAZMINA	SILVA	S01
1802	JAZMINA	SILVA	S01
1802	JAZMINA	SILVA	S01
1803	DIEGO	RUIZ	S03
1803	DIEGO	RUIZ	S03
1803	DIEGO	RUIZ	S03
1804	FREDY	ARROBA	S03
1804	FREDY	ARROBA	S03
1804	FREDY	ARROBA	S03

El producto cartesiano por si solo no es útil sino que se vuelve útil cuando se combina con una condición de JOIN es decir enlazando la clave foránea con su respectiva clave primaria, así:

```
SELECT ESTUDIANTES.CI_Est,
       ESTUDIANTES.Nom_Est,
       ESTUDIANTES.Apell_Est,
       ESTUDIANTES.Cod_Sem_Pert
       SEMESTRES.Cod_Sem,
       SEMESTRES.Nom_Sem,
FROM ESTUDIANTES,SEMESTRES;
WHERE SEMESTRES.Cod_Sem=ESTUDIANTES.Cod_Sem_Pert;
```

El resultado será una tabla con las filas que cumplan la condición de JOIN, especificada en el WHERE de la sentencia SQL anterior, así:

	CI_EST	NOM_EST	APELL_EST	COD_SEM_PE
	1801	DANIEL	YANEZ	S01
	1802	JAZMINA	SILVA	S01
	1803	DIEGO	RUIZ	S03
▶	1804	FREDY	ARROBA	S03
▶	1804	FREDY	ARROBA	S03

## Unión (U)

La Operación Tabla(A) UNION Tabla(B) da como resultado una tercera tabla con las filas de la Tabla A y las filas de la Tabla B.

El requerimiento para la Operación UNION es que las tablas tengan el mismo número de columnas y que estas estén definidas sobre el mismo dominio (los tipos de datos coincidan).

El operador para esta operación es UNION.

## Ejemplo:

Dadas las siguientes tablas:

### Empleados2008

	CED_EMP	NOM_EMP	APE_EMP
	1801	JAZMINA	SILVA
✎	1802	DANIEL	YANEZ
	1803	GABRIELA	CORTEZ
	1804	PAUL	SILVA
*			

### Empleados2009

	CED_EMP	NOM_EMP	APE_EMP	SUELDO_EMP
	1701	FREDY	ARROBA	450
	1706	DIEGO	RUIZ	500
	1801	JAZMINA	SILVA	600
	1803	JONATHAN	SANMARTIN	500
	1804	DANIEL	YANEZ	700
▶	1804	DANIEL	YANEZ	700

La operación UNION se codificará de la siguiente manera en SQL.

```
SELECT Ced_Emp,Nom_Emp, Ape_Emp  
FROM EMPLEADOS2008  
UNION  
SELECT Ced_Emp,Nom_Emp, Ape_Emp  
FROM EMPLEADOS2009;
```

Como se puede apreciar la UNION debe tener como requisito que las tablas a unirse tengan el mismo número de columnas y definidas sobre el mismo dominio.

El resultado será una tercera tabla con las filas de la Tabla Empleados2008 y las filas de la Tabla Empleados2009.

El resultado será:

CED_EMP	NOM_EMP	APE_EMP
1701	FREDY	ARROBA
1706	DIEGO	RUIZ
1801	JAZMINA	SILVA
1802	DANIEL	YANEZ
1803	GABRIELA	CORTEZ
1803	JONATHAN	SANMARTIN
1804	DANIEL	YANEZ
1804	PAUL	SILVA

Si se desea ver los datos repetidos usamos el operador UNION ALL., Así:

```
SELECT Ced_Emp,Nom_Emp, Ape_Emp  
FROM EMPLEADOS2008  
UNION ALL  
SELECT Ced_Emp,Nom_Emp, Ape_Emp  
FROM EMPLEADOS2009;
```

El resultado será:

	CED_EMP	NOM_EMP	APE_EMP
▶	1801	JAZMINA	SILVA
	1802	DANIEL	YANEZ
	1803	GABRIELA	CORTEZ
	1804	PAUL	SILVA
	1701	FREDY	ARROBA
	1801	JAZMINA	SILVA
	1803	JONATHAN	SANMARTIN
	1706	DIEGO	RUIZ
	1804	DANIEL	YANEZ

## Diferencia

Es la operación entre la tabla A y la tabla B que da como resultado otra tabla cuyas filas serán las filas de la Tabla A que no estén en la Tabla B.

El requerimiento para la Operación Diferencia es que las tablas tengan el mismo número de columnas y que estas estén definidas sobre el mismo dominio (los tipos de datos coincidan).

El operador para esta operación en SQL es MINUS.

La diferencia de tablas no es conmutativa, es decir:

Tabla A – la Tabla B NO ES IGUAL A Tabla B – la Tabla A.

## Ejemplo.

Hallar la diferencia entre la Tabla Empleados2008 menos la Tabla Empleados2009.

```
SELECT Ced_Emp,Nom_Emp, Ape_Emp
FROM EMPLEADOS2008
MINUS
SELECT Ced_Emp,Nom_Emp, Ape_Emp
FROM EMPLEADOS2009;
```

El resultado serán los empleados del 2008 que no fueron contratados para el 2009.

	Ced_Emp	Nom_Emp	Ape_Emp
	1803	GABRIELA	CORTEZ
	1804	PAUL	SILVA
►			

Como se mencionó anteriormente la diferencia no es conmutativa, para demostrarlo realizaremos la operación diferencia entre la Tabla Empleados2009 menos la Tabla Empleados2008.

El resultado serán los empleados del 2009 que no estuvieron en el 2008.

	Ced_Emp	Nom_Emp	Ape_Emp
►	1701	FREDDY	ARROBA
	1706	DIEGO	RUIZ
	1803	JONATHAN	SANMARTIN
*			

## Operadores Derivados

### Intersección ( $\cap$ )

La intersección entre la tabla A y la tabla B da como resultado otra tabla con las filas comunes entre la tabla A y la tabla B.

El requerimiento para la Operación Diferencia es que las tablas tengan el mismo número de columnas y que estas estén definidas sobre el mismo dominio (los tipos de datos coincidan).

El operador para esta operación en SQL es INTERSECT

### Ejemplo.

Hallar la intersección entre la Tabla Empleados2008 con la Tabla Empleados2009.

```
SELECT Ced_Emp,Nom_Emp, Ape_Emp
FROM EMPLEADOS2008
INTERSECT
SELECT Ced_Emp,Nom_Emp, Ape_Emp
FROM EMPLEADOS2009;
```

El resultado serán los empleados del 2008 que fueron contratados también para el 2009.(empleados comunes 2008 y 2009)

	Ced_Emp	Nom_Emp	Ape_Emp
	1801	JAZMINA	SILVA
	1802	DANIEL	YANEZ
►			

## Join

El Join se obtiene al aplicar un producto cartesiano (Tuplas Concatenadas) entre uno o varias tablas, y sobre la tabla resultante se aplica la denominada condición de Join.

La tabla resultante estará formada por los tuples que cumplan la condición de join.

Los Join son de gran ayuda para la obtención de información desde varias tablas de de la base de datos.

El Join ya trabaja con mas de una tablas, por lo que es necesario especificar las columnas en el formato: **Tabla.Columna**, para evitar ambigüedades.

Matemáticamente el JOIN se representa con:

- $\Theta_{\text{JOINS}}(\text{tabla1 x tabla2})$



Ejemplo:

Filtrar el nombre, apellido, código del semestre y nombre del semestre al que pertenecen los estudiantes.

### Estudiantes

	Nom_est	Apell_Est	Cod_Sem_Pert
	DANIEL	YANEZ	S01
	JAZMINA	SILVA	S01
	DIEGO	RUIZ	S03
	FREDY	ARROBA	S03
▶			
▶			

### Semestres

	Cod_sem	Nom_sem
+	S01	CUARTO
+	S02	QUINTO
+	S03	SEXTO
+	S03	SEXTO

En SQL la sentencia correspondiente será:

```
SELECT ESTUDIANTES.CI_Est,  
       ESTUDIANTES.Nom_Est,  
       ESTUDIANTES.Apell_Est,  
       ESTUDIANTES.Cod_Sem_Pert  
       SEMESTRES.Cod_Sem,  
       SEMESTRES.Nom_Sem,  
FROM ESTUDIANTES,SEMESTRES  
WHERE SEMESTRES.Cod_Sem=ESTUDIANTES.Cod_Sem_Pert;
```

En donde

**SEMESTRES.Cod\_Sem=ESTUDIANTES.Cod\_Sem\_Pert** es la condición de JOIN.

El resultado será una tabla con las filas que cumplan la condición de JOIN, especificada en el WHERE de la senetencia SQL anterior, así:

	CI_EST	NOM_EST	APELL_EST	COD_SEM_PE
	1801	DANIEL	YANEZ	S01
	1802	JAZMINA	SILVA	S01
	1803	DIEGO	RUIZ	S03
▶	1804	FREDY	ARROBA	S03
▶	1804	FREDY	ARROBA	S03

La condición de JOIN además puede ir acompañada de otros filtros que permitan complementar la consulta.

Por ejemplo:

```
SELECT ESTUDIANTES.CI_Est,
       ESTUDIANTES.Nom_Est,
       ESTUDIANTES.Apell_Est,
       ESTUDIANTES.Cod_Sem_Pert
       SEMESTRES.Cod_Sem,
       SEMESTRES.Nom_Sem,
FROM ESTUDIANTES,SEMESTRES
WHERE SEMESTRES.Cod_Sem=ESTUDIANTES.Cod_Sem_Pert;
AND Apell_Est='RUIZ';
```

El resultado serán los datos del estudiante que cumpla la condición de JOIN y que cumpla con tener el apellido RUIZ.

	CI_EST	NOM_EST	APELL_EST	COD_SEM_PERT
▶	1803	DIEGO	RUIZ	S03
*				

## División

La operación división es la contraria a la operación producto.

En la división debe cumplirse el criterio de totalidad.

Ejemplo.

Una determinada empresa posee una tabla de sucursales, otra tabla de productos y otra tabla con las ventas de los productos en una determinada sucursal.

El primer paso es filtrar en una tabla todos los códigos de todos los productos, a esta tabla la llamaremos A.

Tabla A

Cod_Producto
1035
2249
5818
2241
2518
*

En una segunda tabla llamada tabla B, filtraremos desde la tabla de ventas, el código del producto vendido, y el comercial que vendió, lo hacemos con una proyección y evitamos traer valores duplicados.

El resultado será el siguiente:

Tabla B

Cod_Comercial	Cod_Producto
10	2241
23	5818
23	1035
39	2518
37	2518
10	2249
23	2249
23	2241
23	2518

Si dividimos la tabla B para la tabla A (Tabla A / Tabla B), obtendremos como resultado una tercera tabla en la que:

Los campos que contiene son aquellos de la tabla B que no existen en la tabla A. En este caso el campo Código Comercial es el único de la tabla B que no existen en la tabla A.

Un registro se encuentra en la tabla resultado si y sólo si está asociado en la tabla B con cada fila de la tabla A.

La respuesta será:

	Cod_Comercial
	23

El código comercial es el campo que no está en la Tabla A y 23 es el dato de la tabla B que su código de producto se encuentra en cada tupla de la tabla A.

Es decir el Comercial 23 ha vendido todos los productos

## Asignación

La operación de Asignación, implica asignar un nombre a la tabla vista resultante de las operaciones del álgebra.

### Ejemplo:

```
SQL> Save Nombre_Consulta;      //Salva
SQL> Start Nombre_Consulta;     //Ejecuta
SQL> Get Nombre_Consulta;       // Muestra El Código sin Ejecutar
```

**AutoEvaluación. Ejercicios Propuestos**

Definir la operación Proyección.

Definir la operación Selección.

Definir la operación Unión.

Definir la operación Diferencia.

Definir la operación Intersección.

Definir la operación Producto cartesiano.

Definir la operación Join.

Definir la operación División.

Definir la operación Asignación.

Dadas las siguientes tablas:

Estudiantes\_Sistemas(CI\_Est,Nom\_Est ,Ape\_Est,Nota\_Est)

Estudiantes\_Sistemas(CI\_Est,Nom\_Est ,Ape\_Est,Nota\_Est)

Encontrar los estudiantes de Sistemas que no siguen Ingles.

Encontrar los estudiantes de Ingles que siguen Sistemas.

Encontrar la lista completa de todos los estudiantes(Tanto de  
Sistemas, como de Ingles).

## **BIBLIOGRAFÍA**

1. Korth, H. y Silberchatz A. (1.998). "Fundamentos de Bases de Datos", Tercera Edición. Mc Graw Hill, Madrid.
2. Kroenke, D. (2.003). Procesamiento de Bases de Datos. Octava Edición. Prentice Hall. México.
3. Pressman, R. (1998). Ingeniería del Software. Un Enfoque Práctico. Cuarta Edición. McGraw Hill/Interamericana de España.
4. A. de Miguel y otros; Diseño de Bases de Datos: Problemas Resueltos. Ed Ra-Ma, 2001.
5. Blanco, R., Mendieta, F. (1.994). Diseño de Bases de Datos Metodologías e Implantación. Ingeniería en Sistemas. Universidad de los Andes. Bogota, Colombia.
6. A. de Miguel y otros; Diseño de Bases de Datos Relacionales. Ed Ra-Ma, 1999 (apéndice A).
7. Metodología de diseño de bases datos relacionales  
<http://www3.uji.es/~mmarques/f47/apun/node1.html>
8. Estrategias de Diseño de Bases de Datos Distribuidas.  
[http://base-de-datos0.tripod.com/unidad\\_4.htm](http://base-de-datos0.tripod.com/unidad_4.htm)

- 9.** Date, C.J. (1.993). Introducción a los Sistemas de Bases de Datos. Quinta Edición. Addison-Wesley Iberoamericana. México.
- 10.** Gómez, A. (1993). Diseño y Gestión de Sistemas de Bases de Datos. Primera Edición. Paraninfo. Madrid.
- 11.** Tanenbaum A. (1.997). Redes de Computadoras. Tercera Edición. Prentice Hall. México.
- 12.** Huguet, G. (2.000). Guía útil de Redes Locales. Segunda Edición. Prensa Técnica. Madrid.
- 13.** IBM Business System Planning. (August 1995). Information Systems Planning Guide, Application Manual GE 20-0527-1. White Plains, N.Y. IBM Corporationn.
- 14.** Johnson, J. (2.000). BASES DE DATOS. Modelos, Lenguaje, Diseño. Primera Edición, Oxford University Press. México.