Unidad 3: Modelos de Datos.

Es aparente que una representación del mundo es necesaria, la que debe ser suficientemente abstracta para que no sea afectada por la dinámica del mundo (los pequeños cambios), y debe ser suficientemente robusta para poder representar como los datos y el mundo se relacionan. Una herramienta como esta es llamada **Modelo de Datos,** el cual permite representar en forma más o menos razonable alguna realidad. El modelo de datos permite realizar abstracciones del mundo, permitiendo centrarse en los aspectos macros, sin preocuparse de las particularidades; así nuestra preocupación se centra en generar un esquema de representación, y no en los valores de los datos.

Los modelos de datos nos permiten capturar parcialmente el mundo, ya que es improbable generar un modelo que lo capture totalmente.

Sin embargo se puede tener un conocimiento relativamente completo de la parte del mundo que nos interesa. Así un modelo captura la cantidad de conocimiento tal que cumpla con los requerimientos que nos hemos impuesto previamente.

Un Modelo de Datos define las reglas por las cuales los datos son estructurados. Esta estructuración sin embargo, no da a una interpretación completa acerca de los significados de los datos y la forma en que serán usados. Las operaciones que se permiten efectuar a los datos deben ser definidos..

3.1 Niveles de Abstracción de los datos

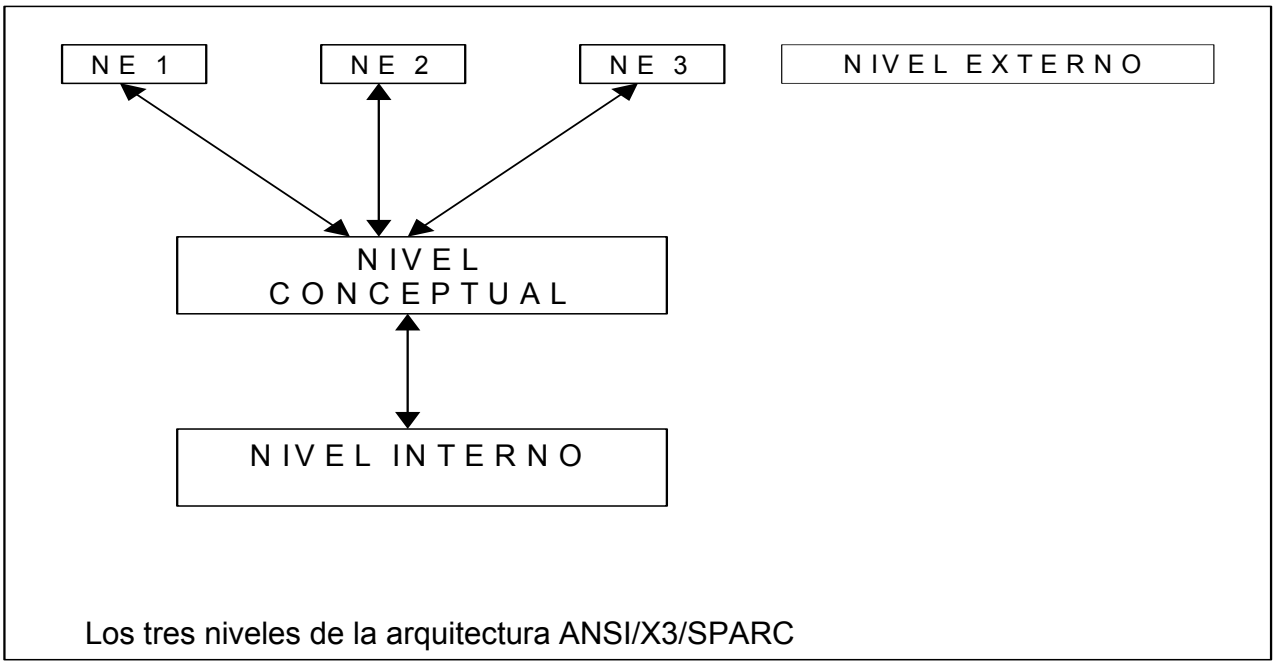
La mayoría de las aplicaciones son dependientes de los datos; la organización del almacenamiento y los modos de acceso dependen de los requerimientos de la aplicación y el conocimiento de la organización física de los datos y las técnicas de acceso forman parte de la lógica de la aplicación. La aplicación es dependiente de los datos, porque no se puede mejorar la estructura de almacenamiento o los modos de acceso sin afectar la aplicación.

En los sistemas de bases de datos se plantean los siguientes objetivos:

1. Independencia de la base de datos de los programas para su utilización.
2. Proporcionar a los usuarios una visión abstracta de los datos. El sistema esconde los detalles de almacenamiento físico (como se almacenan y se mantienen los datos) pero estos deben extraerse eficientemente.

Independencia de los datos:

"La independencia de los datos es la capacidad de un sistema para permitir que las referencias a los datos almacenados, especialmente en los programas y en sus descriptores de los datos, estén aislados de los cambios y de los diferentes usos en el entorno de los datos, como pueden ser la forma de almacenar dichos



Los tres niveles de la arquitectura ANSI/X3/SPARC

- **Nivel Interno:** En el se define la estructura física de la base de datos: dispositivos de almacenamiento físico, direcciones físicas, estrategias de acceso, relaciones, índices, apuntadores, etc. Es responsabilidad de los diseñadores de la base de datos física. Ningún usuario, en calidad de tal, tiene conocimiento de este nivel.

- **Nivel Conceptual:** Contiene el nivel conceptual de la base de datos, que implica el análisis de las necesidades de información de los usuarios y las clases de datos necesarias para satisfacer dichas necesidades. El resultado del diseño conceptual contiene la descripción de todos los datos y las interrelaciones entre ellos, así como las restricciones de integridad y de confidencialidad.

- **Nivel Externo:** Visión que de la base de datos tiene un usuario o aplicación en particular. Habrá tantas vistas de la base de datos como exijan las diferentes aplicaciones. La vistas se derivan directamente del esquema conceptual, o de otras vistas, y con tienen una descripción de los elementos de datos y sus interrelaciones orientadas al usuario o aplicación y de las que se compone la vista. Una misma vista puede ser utilizada por varias aplicaciones.

Esta arquitectura de tres niveles nos proporciona la deseada independencia, que definiremos como capacidad para cambiar el esquema en un nivel sin tener que cambiarlo en ningún otro nivel. Distinguimos entre independencia física y lógica:

- *Independencia lógica de los datos:* Cambio del esquema conceptual sin cambiar las vistas externas o las aplicaciones.

- *Independencia física de los datos:* Cambio del esquema interno sin necesidad de cambiar el esquema conceptual o los esquemas externos.

3.2 Semántica de los datos.

La semántica de los datos es el significado asociado al lenguaje (por ejemplo, el significado de las palabras y su interpretación dentro de un contexto dado).

3.3 Cardinalidad

La Cardinalidad de un objeto o entidad es el número de ocurrencias del objeto, entendiéndose por *ocurrencia* de una entidad o instancia de un objeto, al producto de asociar valores a los atributos de la entidad u objeto.

3.4 Grado

Se denomina grado, a la cantidad de atributos que se consideran para una entidad u objeto.

3.5 Dependencia

Igual que para los tipos de entidad, los tipos de interrelación pueden ser regulares o fuertes y débiles, según se asocien dos entidades fuertes o una fuerte y una débil, respectivamente.

En los tipos de interrelación débil no pueden existir si desaparecen en existencia y la dependencia en identificación.

Dependencia en existencia: Cuando la ocurrencia en un tipo de entidad débil no puede existir si desaparece la ocurrencia de la entidad fuerte de la que depende.

Dependencia en Identificación: Cuando, además de ser una dependencia en existencia las ocurrencias de la entidad débil no pueden identificarse únicamente mediante los atributos propios de la misma.

3.8 Clase

Una clase es un objeto que permite instanciar objetos.

3.9 Agregación

Es una correspondencia que se establece entre dos clases.

3.10 Modelos de Datos dependientes de la tecnología.

La forma o vista externa con que se presentan los datos al usuario en la mayoría de los sistemas actuales es idéntica o muy semejante a la vista conceptual.

La estructura lógica, a nivel contextual o externo, es la base para la clasificación de los DBMS en las tres categorías siguientes: **Jerárquica , Red y Relacional.**

Cualquier categoría del DBMS debe permitir un acceso aleatorio a los datos requeridos, utilizando para tal fin una de las siguientes estructuras lógicas para almacenar los datos: redes, árboles, tablas o listas enlazadas.

Cada DBMS esta diseñado para manejar un tipo determinado de estructura lógica. Los programas que se ejecutan bajo un DBMS no se pueden procesar en otro DBMS.

Los DBMS más conocidos, disponibles en el Mercado en función de su categoría, son:

- **Enfoque Jerárquico**: El IMS de IBM y el SYSTEM 2000 de Intel.
- **Enfoque de Red**: Los ejemplos más importantes los proporciona las especificaciones del grupo de trabajo de base de datos (DBTG) de CODASYL.
- **Enfoque Relacional**: System R y QBE de IBM, MAGNUM de Tymshare, ORACLE y otros.

- **Enfoque Jerárquico**

Un DBMS de enfoque jerárquico utiliza ÁRBOLES para la representación lógica de los datos.

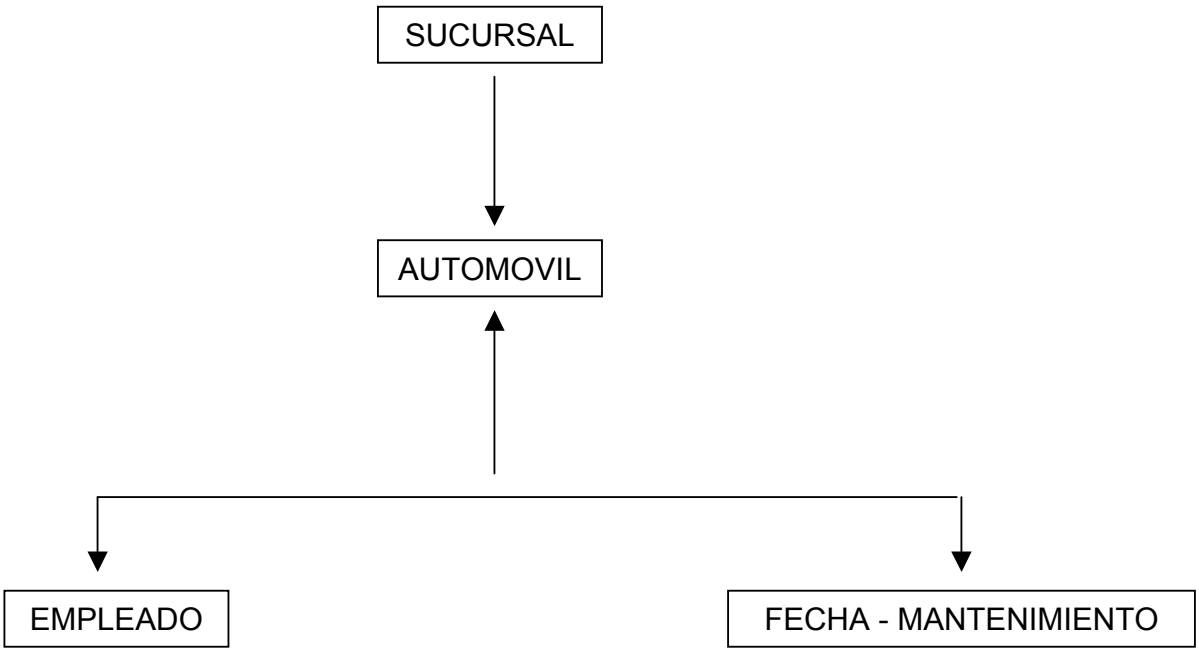
A los archivos que entre sus registros guardan una relación tipo árbol se les llama Archivos Jerárquicos.

La figura siguiente muestra una estructura en árbol con 4 tipos de registros:

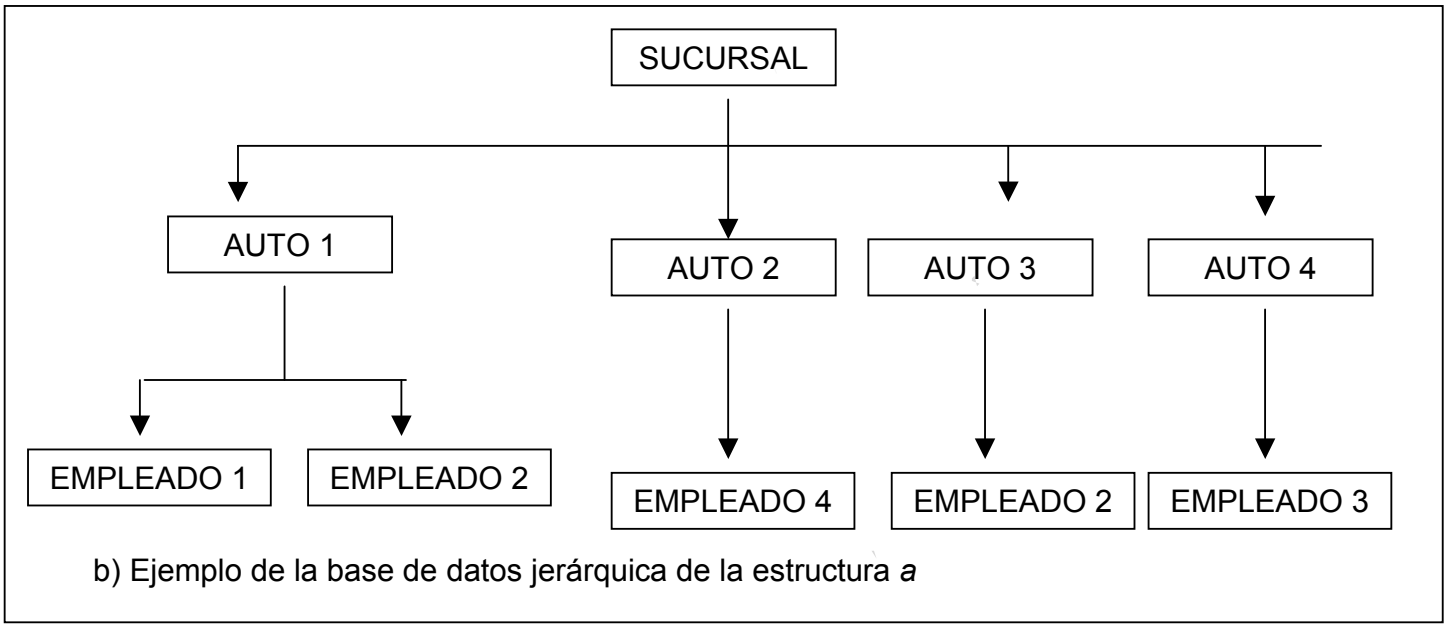
SUCURSAL
AUTOMOVIL
EMPLEADOS
FECHA-MANTENIMIENTO

Que representan las sucursales filiales de una empresa, los automóviles asignados a cada una de ellas, los empleados que deben conducir un determinado coche y la fecha de mantenimiento. El registro SUCURSAL contiene los campos NUMERO-SUCURSAL, NOMBRE-SUCURSAL, NOMBRE-CIUDAD, etc.; el registro AUTOMOVIL incluye los datos de los coches; el registro EMPLEADO, los datos personales del mismo: NUMERO, NONBRE, etc. y, por último el registro FECHA-MANTENIMIENTO contiene los campos FECHA, OPERACIÓN.

Ver figura Anexa.



a) Estructura lógica con cuatro tipos de registros



b) Ejemplo de la base de datos jerárquica de la estructura *a*

Archivo jerárquico con cuatro tipos de registros

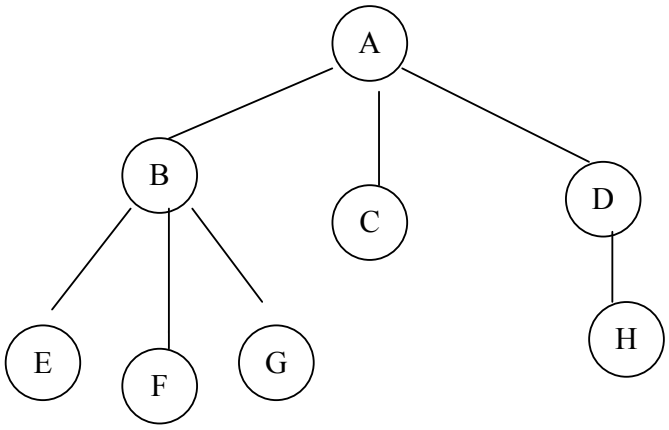
Terminología para describir los árboles

El anexo siguiente muestra un árbol compuesto por una jerarquía de elementos llamados **nodos**. Los árboles se dibujan con la raíz arriba y las hojas abajo

Nivel 1: Raíz

Nivel 2

Nivel 3



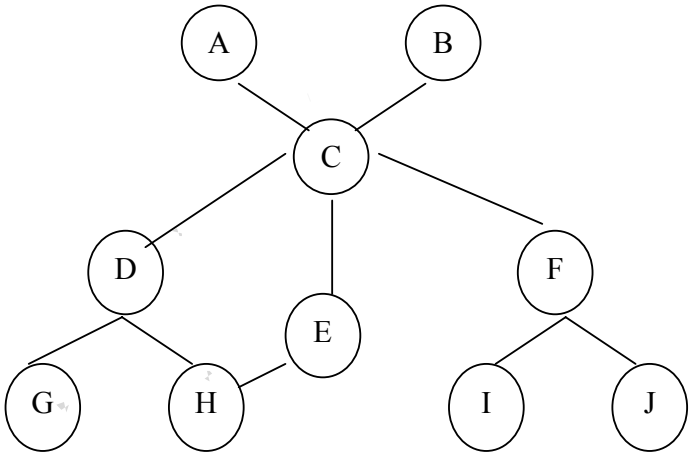
Estructura de un árbol

La terminología para describir los nodos de un árbol es la siguiente:

- Raíz : es el nodo más alto de la jerarquía (nodo A)
- Padre : Es el nodo al que se haya vinculado otros de nivel inferior. EL padre de B es A.
- Gemelos : Nodos con el mismo padre Ej: B, C y D.
- Hijos : Son los nodos vinculados con otros del nivel superior los hijos de B son E, F, G.
- Hojas : Reciben este nombre los nodos que no tienen hijos. (C, H)

El Enfoque de Red

Una estructura de datos en RED, también llamada estructura PLEX, se caracteriza porque cada nodo hijo puede tener más de un padre, a diferencia de la estructura en árbol en la que un hijo sólo podía tener un padre.



Estructura de red

El nodo C tiene dos padres A y B. Lo mismo sucede con en nodo H cuyos padres son D Y E

3.11 Modelos de Datos Independientes de la tecnología.

Objetivos del diseño

En los temas anteriores se han estudiado las arquitecturas de los distintos DBMS, así como los lenguajes para el manejo de los datos. Pero todavía no se ha considerado un aspecto fundamental de las bases de datos, como es su diseño. Por diseño se entiende el generar un conjunto de esquemas de relaciones que permitan almacenar la información con un mínimo de redundancia pero al mismo tiempo faciliten su recuperación.

Entre los distintos objetivos en el diseño de una base de datos se pueden considerar:

1. La base de datos resultante tiene que ser capaz de almacenar toda la información necesaria. El primer paso será determinar los atributos que van a formar parte de la base de datos y reunirlos en una **relación universal**. Hasta que se hayan concretado los campos necesarios no podrá el diseñador establecer las relaciones entre ellos.
2. Eliminación de la información redundante siempre que sea posible.
3. Mantener el número de relaciones al mínimo entre los componentes de la base de datos con el fin de facilitar su programación o uso por parte del usuario.
4. Las relaciones obtenidas deben estar normalizadas con el fin de minimizar los problemas de actualización y borrado.

Orientado a Objeto

El Modelo Orientado a Objetos se basa en el paradigma de programación orientada a objetos. Este paradigma ha tenido gran aceptación debido a que es de gran naturalidad buscar objetos en la realidad a modelar.

Estructura de objetos.
El Modelo orientado a objetos se basa en encapsular código y datos en una única unidad llamada objeto. La Interfaz entre un objeto y el resto del sistema se define mediante un conjunto de mensajes.

El motivo de este enfoque puede ilustrase considerando una base de datos de documentos en la que los documentos se preparan usando uno entre varios paquetes software con formateador de texto. Para imprimir un documento debe ejecutarse el formateador correcto en el documento. Bajo un enfoque orientado a objetos cada documento es un objeto que contiene el texto de un documento y el código que opera sobre el objeto.

Todos los objetos del tipo documento responden al mensaje imprimir, pero lo hacen de forma diferente. Cada documento responde ejecutando el código formateador adecuado. Encapsulando dentro del objeto documento la información acerca de cómo imprimirlo, podemos tener todos los documentos con la misma interfaz externa al usuario (aplicación).

En General un objeto tiene asociado:

- Un conjunto de atributos que contienen datos acerca del objeto. A su vez, cada valor de un atributo es un objeto.
- Un conjunto de mensajes a los que responde.
- Un conjunto (puede ser unitario) de métodos, que es un procedimiento o trozo de código para implementar la respuesta a cada mensaje. Un método devuelve el valor (otro objeto) como respuesta al mensaje.

Puesto que la única interfaz externa de un objeto es el conjunto de mensajes al que responde, es posible modificar la definición de métodos y atributos sin afectar a otros objetos.

También es posible sustituir un atributo por un método que calcule un valor.

Ejemplo: Un objeto documento puede contener un atributo de tamaño que contenga el número bytes de texto en el documento, o bien un método de tamaño que calcule el tamaño del documento leyéndolo y contando el número de bytes.

La capacidad de modificar la definición de un objeto sin afectar al resto del sistema está considerada como una de las mayores ventajas del modelo de programación orientada a objetos.

Entidad - Relación

En 1976, Peter Chen publicó el modelo entidad – relación, el cual tuvo gran aceptación principalmente por su expresividad gráfica. Sobre esta primera versión han trabajado numerosos autores, generando distintas extensiones de mayor a menor utilidad y de aceptación variable en el medio académico y profesional. Muchas de estas extensiones son muy útiles, pero poco difundidas debido principalmente ala ausencia de herramientas automatizadas que apoyen su uso.

¿Cómo modelar en MER (Modelo Entidad Relación)?

Para modelar en MER se sigue generalmente el siguiente orden:

1. Identificar los tipos de entidades
2. Identificar los tipos de Interrelaciones
3. Encontrar las cardinalidades

4. Identificar los atributos de cada entidad
5. Identificar las claves de cada tipo de entidad

La regla básica es distinguir tipos de entidades e interrelaciones de atributos. Así, los atributos deben ser atómicos y característicos del tipo entidad o interrelación que describan.

También los atributos deben pertenecer al tipo de entidad o interrelación que describen y no a otro tipo.

Otra diferencia entre tipo entidad y atributo es que, por ejemplo, se puede tener el tipo de entidad empleado, que tiene como atributo el departamento al que pertenece. En Forma alternativa se pueden tener los tipos de entidades Empleado y Departamento, y el tipo de interrelación trabaja_en, que relaciona a un empleado con el departamento en donde trabaja.

Esta segunda alternativa es mejor desde el punto de vista del modelamiento conceptual y presenta una clara diferencia entre atributo y tipos de entidad.

Reglas para elegir identificadores

1. No deben existir dos entidades con el mismo valor del identificador (en los tipos de entidad).
2. En los tipos de interrelación, la clave es la composición de las claves de los tipos de entidad involucrados, en caso que no se pueda utilizar la clave de un subconjunto de ellos.

Ejercicios Propuestos:

1. Construir un esquema MER para una secretaria de universidad. La secretaria mantiene datos sobre cada asignatura, incluyendo el profesor, lista de alumnos y la hora y el lugar de las clases. Para cada par estudiante – asignatura se registra su nota.

2. Construir un esquema MER para una compañía de seguros de autos con un conjunto de clientes, cada uno de los cuales es propietario de un número de autos. Cada auto tiene asociado el número de accidentes asociados.

3. Construir un esquema MER para modelar la documentación requerida para un esquema conceptual E – R.

3.12 Normalización

Se entiende por normalización la descomposición o subdivisión de una relación en dos o más relaciones para evitar la redundancia; en definitiva, que "cada hecho esté en su lugar".

El proceso de normalización generalmente se utiliza en el enfoque relacional; sin embargo, un modelo relacional se puede modificar para su implantación en un DBMS jerárquico o de red.

La relación universal

Supongamos que se desea implantar en una base de datos las ventas de una determinada empresa a sus clientes por la relación ORDENES-VENTA (NCLI, NOMBRE, LOCALIDAD, CT, NART, ARTICULO, CANT, PVP, FECHA), donde NCLI es el número del cliente, CT es el costo de transporte y NART el número de artículo. La implantación, tal como indica la Figura 1, no se puede realizar debido a la gran cantidad de información redundante y los problemas que surgen a la hora de las actualizaciones.

Relación ORDENES-VENTA

NCLI	NOMBRE	LOCALIDAD	CT	NART	ARTICULO	CANT	PVP	FECHA
11	Luis	Málaga	0.8	A1	Papel	100	5	3/5
11	Luis	Málaga	0.8	A3	Cinta	50	500	5/5
11	Luis	Málaga	0.8	A9	Disco	25	200	7/5
44	Ana	Gijón	1.1	A1	Papel	100	5	10/5
55	José	Valencia	1.4	A4	Grapas	30	50	3/5

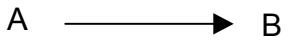
Figura 1. Información deseada para las ORDENES-VENTA

DEPENDENCIA FUNCIONAL: DF

La normalización se basa en la dependencia funcional.

El concepto de dependencia funcional se tomó de las matemáticas. [Y = f(X)], Y es función de X si el valor de Y está siempre determinado por el valor de X.

La **dependencia funcional (DF)** se define: Dados dos atributos A y B de una relación R se dice que B es funcionalmente dependiente del atributo A si para cada valor de A existe un valor de B, y sólo uno, asociado con él. En otros términos: si en cualquier instante, conocido el valor de A, podemos conocer el valor de B. Se simboliza por:



Tanto A como B pueden ser un conjunto de atributos en lugar de atributos simples.

La dependencia funcional establece condiciones entre atributos pertenecientes a la misma relación. No permite establecer condiciones entre atributos pertenecientes a la misma relación. No permite establecer condiciones entre atributos de diferentes relaciones.

Las DF se determinan al estudiar las propiedades de todos los atributos de la relación y deducir cómo están relacionados los atributos entre sí.

La dependencia funcional está íntimamente ligada con el concepto de **clave**. Para el diseño, las claves aparecen subrayadas.
Se pueden distinguir los siguientes tipos de claves:

- **Clave candidata:** Conjunto de uno o más atributos que podría ser utilizado como clave principal de una relación.
- **Superclave:** Conjunto de uno o más atributos que, juntos, permiten identificar de forma única a una entidad dentro de una relación.
- **Clave principal:** Es una clave candidata en la que ningún componente puede tomar el valor nulo.

Para encontrar la clave candidata es preciso estudiar las dependencias funcionales y, a partir de ellas, obtener el mínimo conjunto posible de atributos tales que, una vez conocidos sus valores en la tupla, los demás queden definidos

Consideremos la relación CLIENTES: NCLI, NOMBRE, LOCALIDAD, donde NCLI es el número del cliente. Los campos NOMBRE y LOCALIDAD son funcionalmente dependientes de NCLI: Para un valor de NCLI existe un único valor de NOMBRE y LOCALIDAD. Se expresa:

NCLI $\longrightarrow$ NOMBRE
NCLI $\longrightarrow$ LOCALIDAD
En forma abreviada:
NCLI $\longrightarrow$ (NOMBRE, LOCALIDAD)

La proposición NCLI $\longrightarrow$ NOMBRE se lee: “el atributo NOMBRE es funcionalmente dependiente del atributo NCLI”, o también: “el atributo NCLI determina funcionalmente al atributo NOMBRE”.

La proposición NCLI $\longrightarrow$ (NOMBRE, LOCALIDAD) se puede leer: “el atributo compuesto formado por NOMBRE y LOCALIDAD es funcionalmente dependiente de NCLI”.

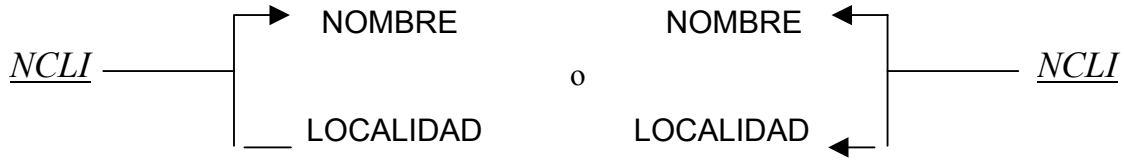
El atributo NCLI es un **determinante** de los atributos NOMBRE y LOCALIDAD.
Dicho de otra forma: por cada NCLI sólo puede haber un NOMBRE y una

LOCALIDAD asociados a él. NCLI es una superclave. Sin embargo, el NOMBRE no es un determinante de la LOCALIDAD, ya que puede haber varias personas con igual nombre en ciudades diferentes o en la misma ciudad

Determinante: Si A B es una DF y B *no* es funcionalmente dependiente de A se dice que A es el determinante de B.

Un determinante son todos los atributos situados en el lado izquierdo de una DF.

Es conveniente representar las DF de una relación en un **diagrama de dependencia funcional**; en el ejemplo anterior:



El diagrama de dependencia funcional para la relación ORDENES-VENTA se muestra en la Figura 2. Se aprecia que el atributo CANT es totalmente dependiente de los atributos NCLI, NART y FECHA, lo que da lugar a la aparición de un nuevo concepto: **dependencia funcional total.**

Dependencia funcional total: En una relación R, un atributo o colección de atributos B tiene una dependencia funcional total de otra colección de atributos A de la relación R, si B es funcionalmente dependiente de todos los atributos de A pero no de un subconjunto de A.

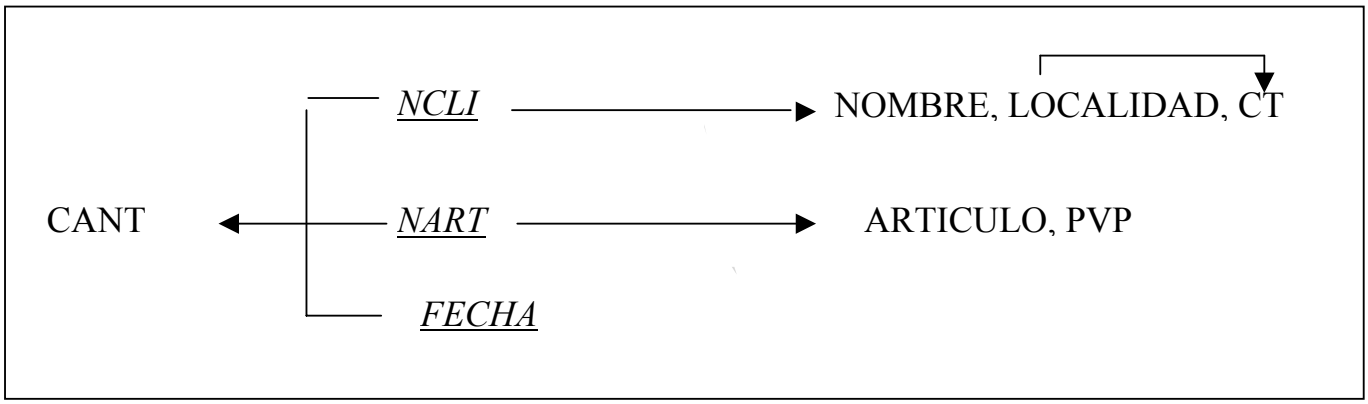


Figura 2 Diagrama de dependencia funcional

PRIMERA FORMA: 1FN

Una relación está en primera forma normal si todo atributo contiene un valor indivisible, atómico.

Esta forma normal está justificada por la sencillez y la estética en la representación de los registros (Fig. 1).

NCLI	NOMBRE	LOCALIDAD	CT	NART	ARTICULO	CANT	PVP	FECHA
11	Luis	Málaga	0.8	A1	Papel	100 50	5 5	3/5 5/5

se puede normalizar con la creación de un registro nuevo por cada uno de los distintos valores de un campo, tal que permita expresar la relación como una tabla

NCLI	NOMBRE	LOCALIDAD	CT	NART	ARTICULO	CANT	PVP	FECHA
11 11	Luis Luis	Málaga Málaga	0.8 0.8	A1 A3	Papel Cinta	100 50	5 500	3/5 5/5

Una relación en 1FN contiene una serie de **anomalías de almacenamiento** a la hora de realizar las actualizaciones por la información redundante, como se puede apreciar en la Figura 1.

NORMALIZACIÓN DE LA RELACIÓN 1FN

Las anomalías de almacenamiento, que se deben a la presencia de campos no clave en la relación, se pueden subsanar de la siguiente forma:

a) Dividiendo la relación universal en nuevas relaciones.
b) Cada relación tiene la propiedad de que su clave, en su totalidad, es necesaria para definir cada uno de los campos no clave.

Al proceso de dividir cualquier relación en dos o más relaciones se llama **proceso de normalización**. Consiste en reemplazar las relaciones por proyecciones adecuadas, de tal forma que la reunión natural de las proyecciones genere la relación original, es decir, que no se produzca pérdida de la información. Incluso las nuevas relaciones pueden contener información que no se podía representar originalmente (un nuevo registro en alguna de las nuevas relaciones), pero siempre conservando las dependencias funcionales.

Descomposición sin pérdida. Descomposición de una relación R en R1, R2, ... RN, tal que:

$$R = R1 * R2 * ... * RN.$$

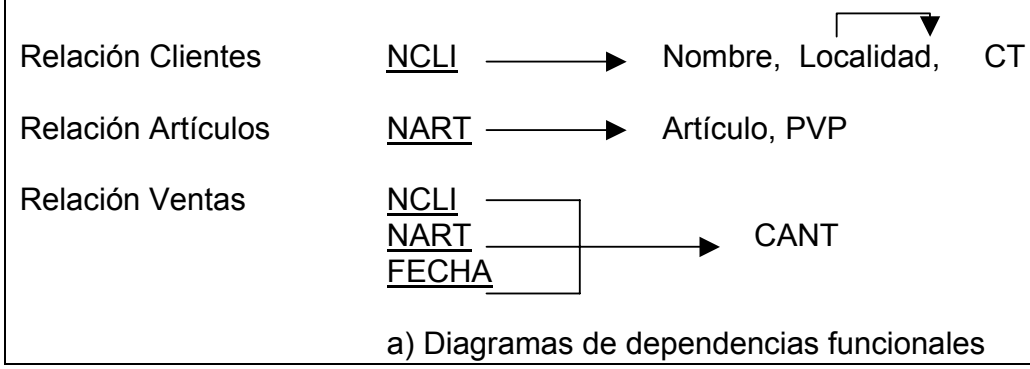
Cuando se actualiza la base de datos, el sistema debe poder comprobar que la actualización no va a generar una relación ilegal, es decir, una que no satisfaga todas las DF establecidas.

Para llevar a cabo el proceso de normalización es aconsejable dar los siguientes pasos:

1. Elegir una clave primaria que puede representar de forma única a cada registro de la relación.
2. Construir un diagrama de dependencia en función de esas claves.
3. Construir las nuevas relaciones basándose en dichas claves.

Por el paso 1, en la relación ORDENES-VENTA, los atributos que forman la clave primaria son: NCLI, NART, FECHA.

Los diagramas y las nuevas relaciones aparecen descritas en la figura siguiente.



a) Diagramas de dependencias funcionales

Relación CLIENTES

NCLI	NOMBRE	LOCALIDAD	CT
11	Luis	Málaga	0.8
44	Ana	Gijón	1.1
55	José	Valencia	1.4

Relación ARTICULOS

NART	ARTICULO	PVP
A1	Papel	5
A3	Cinta	500
A4	Grapas	50
A9	Disco	200

Relación VENTAS

NCLI	NART	CANT	FECHA
11	A1	100	3/5
11	A3	50	5/5
11	A9	25	7/5
44	A1	130	10/5
55	A4	30	3/5

b)Registros de las relaciones

RELACION 1FN NORMALIZADA

SEGUNDA FORMA NORMAL: 2FN

Una relación está en segunda forma normal sí, y sólo sí:

1. Está en 1FN
2. Todo atributo que no pertenezca a la clave debe depender de la clave en su totalidad y no sólo de una parte; debe tener una dependencia funcional total.

Las relaciones mostradas en la Figura siguiente pertenecen ya a la 2FN. Sin embargo, la relación CLIENTES presenta anomalías de almacenamiento debido a que el atributo CT es funcionalmente dependiente de LOCALIDAD, que a su vez depende de NCLI; es decir, hay una **dependencia transitiva** que ocasiona problemas a la hora de las actualizaciones.

Por ejemplo, no se puede insertar un CT para una localidad determinada hasta que haya un cliente para dicha localidad.

Dependencia transitiva

Supongamos la relación R(*A*,B,C). Si A $\longrightarrow$ B, B $\longrightarrow$ C y B $\not\longrightarrow$ A ; entonces se dice que C depende transitivamente de A y se puede formar la **cadena**

$\longrightarrow$ $\longrightarrow$

A B C.

En un diagrama de dependencia funcional, C es transitivamente dependiente de A si se tiene la siguiente situación:

Se puede descomponer en dos relaciones por la proyección del último eslabón de la forma;

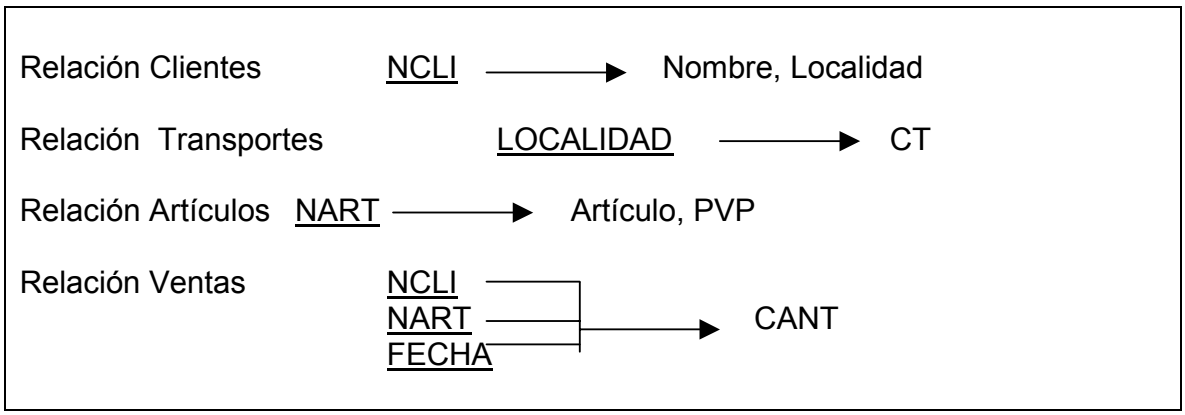
Normalización de la relación 2FN

Las anomalías de almacenamiento, originadas por la dependencia transitiva en una relación 2FN, se puede normalizar mediante los siguientes pasos:

1. En una relación, determinar el atributo que es funcionalmente dependiente de un atributo no clave y dibujar el diagrama de dependencia funcional.

2. Crear una nueva relación para almacenar el atributo no clave y su determinante

El diagrama de dependencia funcional y las relaciones CLIENTES y TRANSPORTE se muestran en la figura 13.4. Han desaparecido las anomalías surgidas por la dependencia transitiva, como se puede comprobar al dar de alta un nuevo registro en la relación TRANSPORTE, aunque no haya ningún cliente de esa ciudad.



a) Diagramas de dependencias funcionales

Relación CLIENTES

NCLI	NOMBRE	LOCALIDAD
44	Ana	Gijón
11	Luis	Málaga
55	José	Valencia

Relación TRANSPORTES

LOCALIDAD	CT
Málaga	0.8
Gijón	1.1
Valencia	1.4

b) Registros de las relaciones CLIENTES Y TRANSPORTES

RELACION 2FN NORMALIZADA

TERCERA FORMA NORMAL: 3FN

Una relación está en 3FN sí, y sólo sí:

1. Está en 2FN
2. Todo atributo que no pertenezca a la clave no depende de un atributo no clave.

La 3FN elimina las redundancias ocasionadas por las dependencias transitivas.

Las relaciones mostradas en la figura 13.4 pertenecen ya a la 3FN.

En la 3FN se puede decir que en cada relación no existe u atributo no clave que defina a otro atributo. Existe una excepción: Cuando en una relación hay dos atributos que podría ser la clave, como el DNI y l número de la seguridad social.