Diseño de bases de datos Relacionales

Miguel Sosa Flores
Flor Angel Hernández Pérez

DISEÑO DE BASES DE DATOS RELACIONALES

AUTORES: Lic. Miguel Sosa Flores
M.Sc. Manuel López Vázquez

ÍNDICE

RESUMEN

El objetivo del presente material es el de satisfacer las necesidades apremiantes que existen en la enseñanza del Diseño de Bases de Datos Relacionales tanto en pregrado como postgrado. Los autores para su confección, además de apoyarse en la literatura existente sobre el tema, utilizan sus experiencias como profesores universitarios de impartir estos contenidos, en diversos cursos de pregrado, postgrado y maestrías, así como la obtenida de los trabajos prácticos que realizan a organismos y empresas.

La presente edición tiene un carácter provisional, un trabajo posterior permitirá ofrecer una edición definitiva que reúna todos los requerimientos de un libro de texto, la cual se enriquecerá con un buen número de problemas prácticos adecuados estos desde el punto de vista docente.

Los autores.

Tema 1
Conceptos Básicos

Contenido:

- **Surgimiento histórico de las bases de datos integradas.**
- **Objetivos de los SBD.**
- **Arquitectura de un sistema de base de datos.**
- **Representación de la Información**

Surgimiento histórico de las bases de datos integradas.

Al estudiar el desarrollo del procesamiento automatizado de datos, en lo que se refiere al aseguramiento técnico, se habla de diferentes generaciones.

Desde el punto de vista del aseguramiento matemático y en particular, el aseguramiento de programas, algunos autores reconocen 3 generaciones:

a) Solución de tareas aisladas.

b) Integración de tareas aisladas en sistemas particulares.

c) Integración de sistemas particulares en sistemas automatizados de dirección.

Este proceso de integración ocurre paralelamente, aunque no simultáneamente, en dos esferas:

a) Integración de los programas.

b) Integración de los datos.

a) Ha estado facilitada por el uso de lenguajes de programación cada vez más sofisticados y de redactores que permiten el acoplamiento de módulos escritos en lenguajes diferentes.

b) En la integración de los datos se han producido tres categorías de técnicas para su manipulación:

1- Sistemas orientados a los dispositivos.

Programas y ficheros son diseñados y empleados de acuerdo a las características físicas de la unidad central y los periféricos. Cada programa está altamente interconectado con sus ficheros, por lo que la integración de datos de diferentes sistemas es imposible prácticamente.

2- Sistemas orientados a los ficheros.

La lógica de los programas depende de las técnicas de organización de los ficheros (secuencial, directo, etc.). Cada usuario organiza su fichero de acuerdo con sus necesidades y las relaciones entre los elementos se establecen a través de los programas de aplicación.

Esta forma de trabajo implica redundancia de datos que trae aparejada mayor gasto de memoria y complica las operaciones de actualización (modificar un dato donde quiera que aparezca). Esto aumenta el tiempo de tratamiento y atenta contra la integridad de la información.

Integridad: que en todo momento los datos almacenados estén correctos en correspondencia con la realidad.

Además, en la vida real se establecen relaciones entre los objetos que son muy difíciles de representar u obtener a partir de sistemas tradicionales de ficheros. Por ejemplo, si se tiene información sobre trabajadores y estudiantes de una facultad, las aplicaciones requeridas van a definir la manera de organizar y estructurar los ficheros.

Si se desea obtener datos como:

- Calificaciones promedio de cada estudiante.
- Listado de estudiantes por grupos.
- Categoría científica y docente de cada profesor.
- Salario de cada profesor.

Resulta adecuado establecer dos ficheros: uno de profesores y uno de estudiantes.

¿Qué ocurre si se quiere establecer vínculos entre los profesores y estudiantes? Por ejemplo, si se desea obtener:

- Los estudiantes de un profesor.
- Los profesores de un estudiante.

Se estructuraría un fichero de profesores y estudiantes que resolvería algunas demandas, pero sería ineficiente para otras.

¿ Será posible representar de manera eficiente, utilizando los medios de cómputo, los fenómenos o procesos de la realidad objetiva, aunque sea por supuesto de forma esquemática, pero en la que se establezcan determinados vínculos entre los elementos u objetos que forman parte de esos procesos o fenómenos?

Veremos que esto es posible hacerlo a través de la utilización de bases de datos (BD) y de los sistemas de gestión de bases de datos (SGBD) que dirigen su manipulación.

3- Sistemas orientados a BD

En ellos hay una débil interdependencia entre los programas de aplicación y la organización física de los datos.

¿Qué es una base de datos?

Definición: Conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una BD puede considerarse una colección de datos variables en el tiempo.

El software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina **sistema de gestión de bases de datos** (SGBD).

Es importante diferenciar los términos BD y SGBD.

El objetivo fundamental de un SGBD consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, o sea, de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado.

Los programas de aplicación operan sobre los datos almacenados en la base utilizando las facilidades que brindan los SGBD, los que, en la mayoría de los casos, poseen lenguajes especiales de manipulación de la información que facilitan el trabajo de los usuarios.

Objetivos de los SBD

Existen muchas formas de organizar las bases de datos, pero hay un conjunto de objetivos generales que deben cumplir todas los SGBD, de modo que faciliten el proceso de diseño de aplicaciones y que los tratamientos sean más eficientes y rápidos, dando la mayor flexibilidad posible a los usuarios. Estos objetivos son:

a. Independencia de los datos y los programas de aplicación.

Ya vimos que con ficheros tradicionales la lógica de la aplicación contempla la organización de los ficheros y el método de acceso. Por ejemplo, si por razones de eficiencia se utiliza un fichero secuencial indexado, el programa de aplicaciones debe considerar la existencia de los índices y la secuencia del fichero. Entonces es imposible modificar la estructura de almacenamiento o la estrategia de acceso sin afectar el programa de aplicación (naturalmente, lo que se afecta en el programa son las partes de éste que tratan los ficheros, lo que es ajeno al problema real que el programa de aplicación necesita resolver. En un SBD sería indeseable la existencia de aplicaciones y datos dependientes entre sí, por dos razones fundamentales:

1. Diferentes aplicaciones necesitarán diferentes aspectos de los mismos datos (decimal o binario).

2. Se debe poder modificar la estructura de almacenamiento o el método de acceso según los cambios en el fenómeno o proceso de la realidad sin necesidad de modificar los programas de aplicación (también para buscar mayor eficiencia).

Independencia de los datos: inmunidad de las aplicaciones a los cambios en la estructura de almacenamiento y en la estrategia de acceso , constituye el objetivo fundamental de los SBD.

b. Minimización de la redundancia

Habíamos visto cómo, con los ficheros tradicionales, se produce redundancia de la información. Uno de los objetivos de los SBD es minimizar la redundancia de los datos. Se dice disminuir la redundancia, no eliminarla, pues, aunque se definen las BD como no redundantes, en realidad existe redundancia en un grado no significativo para disminuir el tiempo de acceso a los datos o para simplificar el método de direccionado. Lo que se trata de lograr la eliminación de la redundancia superflua.

c. Integración y sincronización de las bases de datos

La integración consiste en garantizar una respuesta a los requerimientos de diferentes aspectos de los mismos datos por diferentes usuarios, de forma que, aunque el sistema almacene la información con cierta estructura y cierto tipo de representación, debe garantizar entregar al programa de aplicación los datos que solicita y en la forma en que lo solicita. Esta se encuentra vinculada a la sincronización, que consiste en la necesidad de garantizar el acceso múltiple y simultáneo a la BD, de modo que los datos puedan ser compartidos por diferentes usuarios a la vez. Están relacionadas, ya que lo usual es que diferentes usuarios trabajen con diferentes enfoques y requieran los mismos datos, pero desde diferentes puntos de vista.

d. Integridad de los datos

Consiste en garantizar la no contradicción entre los datos almacenados de modo que, en cualquier momento del tiempo, los datos almacenados sean correctos, es decir, que no se detecte inconsistencia entre los datos. Está relacionada con la minimización de redundancia, ya que es más fácil garantizar la integridad si se elimina la redundancia.

e. Seguridad y protección de los datos

Protección: garantizar el acceso autorizado a los datos, de forma de interrumpir cualquier intento de acceso no autorizado, ya sea por error del usuario o por mala intención.

Seguridad: que el sistema de bases de datos disponga de métodos que garanticen la restauración de las BD al producirse alguna falla técnica, interrupción de la energía eléctrica, etc.

f. Facilidad de manipulación de la información

Los usuarios de una BD pueden referirse a ella con las solicitudes para resolver muchos problemas diferentes. El SBD debe contar con la capacidad de una búsqueda rápida por diferentes criterios, permitir que los usuarios planteen sus demandas de una forma simple, aislándolo de las complejidades del tratamiento de los ficheros y del direccionado de los datos. Los SBD actuales brindan lenguajes de alto nivel con diferentes grados de facilidad para el usuario no programador que garantizan este objetivo, los llamados sublenguajes de datos.

g. Control centralizado

Uno de los objetivos más importantes de los SBD es garantizar el control centralizado de la información, es decir, controlar de manera sistemática y única los datos que se almacenan en la BD, así como el acceso a ella.

Lo anterior implica que debe existir una persona o conjunto de personas que tenga la responsabilidad de los datos operacionales: el administrador de la BD, que puede considerarse parte integrante del SBD. Entre las tareas del administrador de la BD está:

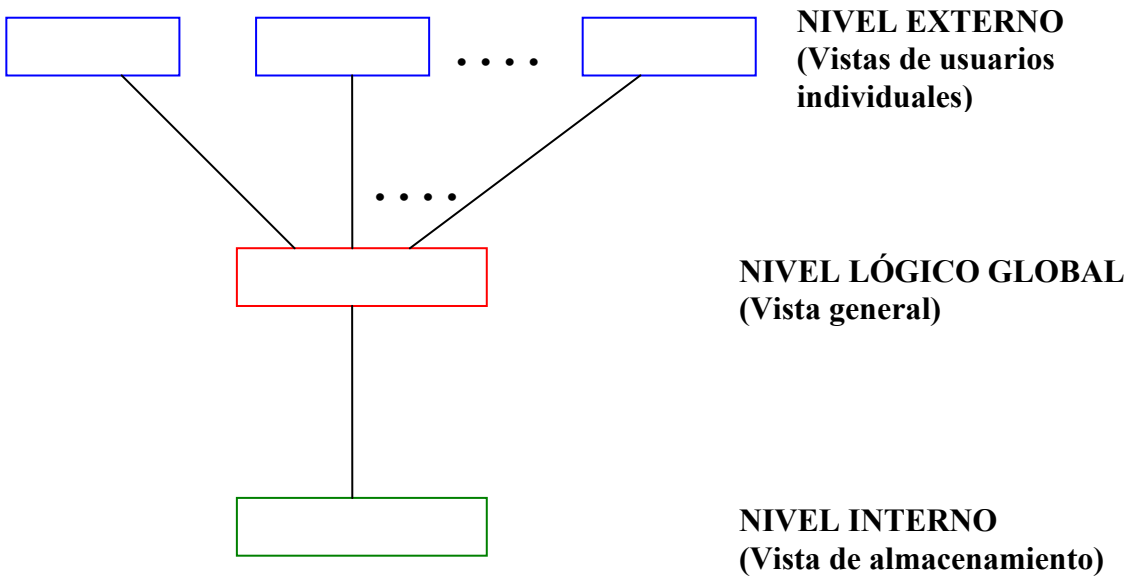
- decidir el contenido informativo de la BD
- decidir la estructura de almacenamiento y la estrategia de acceso
- garantizar el enlace con los usuarios
- definir los chequeos de autorización y procedimientos de validación
- definir la estrategia de reorganización de las BD para aumentar la eficiencia del sistema

Existen otros objetivos que deben cumplir los SBD que en muchos casos dependen de las condiciones o requerimientos específicos de utilización del sistema.

Arquitectura de un SBD

Presentaremos a continuación la arquitectura de un SBD, aunque no podemos asegurar que cualquier SBD se corresponda exactamente con ella. Sin embargo, esta arquitectura se corresponde suficientemente bien con un gran número se sistemas. Además, está de acuerdo con la arquitectura propuesta por el grupo ANSI/SPARC.

La arquitectura se divide en tres niveles generales: interno, lógico global y externo.



El nivel interno es el más cercano al almacenamiento físico, o sea, es el relacionado con la forma en que los datos están realmente almacenados.

El nivel externo es el más cercano a los usuarios, o sea, es el relacionado con la forma en que los datos son vistos por cada usuario individualmente.

El nivel lógico global es un nivel intermedio entre los dos anteriores.

Existirán varias "**vistas externas**" , siendo cada una representación más o menos abstracta de alguna porción de la BD total y existiendo una única "**vista general**", consistente en una representación también abstracta de la BD en su totalidad. Igualmente, existirá una única "vista interna" que representa a la BD completa, tal y como está realmente almacenada.

A continuación estudiaremos con mayor detalle cada uno de los niveles de la arquitectura vista anteriormente y la forma en que ellos interactúan.

A. El nivel externo

Es el nivel del usuario individual, donde un usuario puede ser bien un programador de aplicación o un usuario final con cualquier grado de sofisticación. Cada usuario tiene un lenguaje a su disposición:

- Para el programador, ese lenguaje será bien un lenguaje de programación convencional, tal como Pascal o C, o bien un lenguaje de programación específico de un sistema, tal como el FoxPro.

- Para el usuario final, el lenguaje será bien un lenguaje de consulta (interrogaciones, query) o un lenguaje de propósito especial, quizás basado en sistemas de menúes y construido para satisfacer los requerimientos de un usuario, encontrándose soportado por algún programa de aplicación en línea.

Es importante señalar que todo lenguaje incluirá un sublenguaje de datos, o sea, un subconjunto del lenguaje que trata específicamente con los objetos de la base de datos y sus operaciones. Se dice que el sublenguaje de datos (DSL) está embebido dentro del correspondiente lenguaje huésped. Este lenguaje huésped es el encargado de asegurar otras facilidades ajenas a la base de datos, tales como variables locales, operaciones de cálculo, lógica if-then-else, etc. Un sistema dado, puede soportar múltiples lenguajes huésped y múltiples sublenguajes de datos.

En principio, cualquier sublenguaje de datos es realmente una combinación de, al menos, dos lenguajes subordinados: un lenguaje de definición de datos (DDL), el cual garantiza la definición o descripción de los objetos de la base de datos, y un lenguaje de manipulación de datos (DML), el que garantiza la manipulación o procesamiento de esos objetos.

Ya se ha indicado que un usuario individual estará generalmente interesado sólo en cierta porción de la BD completa. Aún más, la vista de esa porción será generalmente abstracta cuando se compara con la forma en que los datos están físicamente almacenados. El término definido por el comité ANSI/SPARC para una vista de un usuario es vista externa, la cual es el contenido de la BD como es vista por un usuario en particular. O sea, para ese usuario, la vista externa es la BD.

En general, una vista externa consiste en múltiples ocurrencias de múltiples tipos de artículos externos. Un artículo externo no es necesariamente igual a un artículo almacenado.

El sublenguaje de datos del usuario se define en términos de artículos externos; por ejemplo, una operación del DML que sea recuperar artículos, recuperará una ocurrencia de artículos externos y no una ocurrencia de artículos almacenados.

Cada vista externa se define mediante un esquema externo, consistente, básicamente, en definiciones de cada uno de los diferentes tipos de artículos externos en esa vista. El esquema externo se escribe usando la porción del DDL del sublenguaje de datos del usuario, además tiene que existir una definición de la correspondencia entre el esquema externo y el esquema lógico global.

B. El nivel lógico global

La vista lógica es una representación del contenido informativo total de la BD. Es una forma abstracta en comparación con la forma en que los datos están almacenados físicamente. Esta vista puede ser bien diferente de la forma en la que los datos son vistos por un usuario en particular. La vista lógica pretende ser una vista de los datos tal como son, en lugar de cómo los usuarios están forzados a verlos por las restricciones, digamos, de un lenguaje particular o de un determinado hardware que utilicen.

La vista lógica consiste en múltiples ocurrencias de múltiples tipos de artículos lógicos. Por ejemplo, puede ser una colección de ocurrencias de artículos de departamentos, más una colección de ocurrencia de artículos de empleados, etc. Un artículo lógico no es necesariamente igual a un artículo externo ni a un artículo almacenado.

La vista lógica se define mediante el esquema lógico que incluye las definiciones de cada uno de los diferentes tipos de artículos lógicos. El esquema lógico se describe usando otro lenguaje de definición de datos: el DDL lógico. Si se desea lograr la independencia de los datos, entonces las definiciones del DDL lógico no deben comprender ninguna consideración sobre la estructura de almacenamiento ni la estrategia de acceso. Ellas tienen que ser definiciones sólo referentes al contenido informativo.

Si el esquema lógico logra verdaderamente la independencia de los datos, entonces los esquemas externos que se definen sobre el esquema lógico lograrán también, necesariamente, la independencia de los datos.

La vista lógica es entonces una vista del contenido total de la BD y el esquema lógico es una definición de esa vista. Sin embargo, el esquema lógico no es simplemente un conjunto de definiciones como las que se encuentran, por ejemplo, en un programa Pascal. Las definiciones en el esquema lógico deben incluir una gran cantidad de aspectos adicionales, tales como los chequeos de protección y los chequeos de integridad.

En la mayoría de los sistemas actuales, el esquema lógico es realmente sólo un poco más que la simple unión de todos los esquemas externos individuales, posiblemente con la adición de algunos chequeos simples de protección e integridad. Sin embargo, está claro que los sistemas del futuro soportarán un nivel lógico mucho más sofisticado, que permita también describir la forma en que se usan los datos, cómo fluyen de un punto a otro, para qué se usan en cada punto, a qué controles son sometidos, etc.

C. El nivel interno

La vista interna es una representación de bajo nivel de la BD completa, que consiste en múltiples ocurrencias de múltiples tipos de artículos internos.

"**Artículo interno**" es el término definido por ANSI/SPARC para la construcción que hasta ahora hemos llamado artículo almacenado. La vista interna está entonces aún a un paso del nivel físico, ya que ella no opera en término de artículos físicos (también llamados páginas o bloques) ni con consideraciones específicas de los equipos, tales como tamaños de sectores o pistas. Básicamente, la vista interna asume un espacio de dirección lineal infinita. Los detalles de cómo se hace corresponder ese espacio con el almacenamiento físico son muy específicos de un sistema y deliberadamente se omitieron de la arquitectura.

La vista interna se describe mediante el esquema interno, el cual no sólo define los diferentes tipos de artículos almacenados, sino que también especifica los índices que existen, la representación de las

campos almacenados, la secuencia física en que están los artículos almacenados, etc. El esquema interno se describe usando otro lenguaje de definición de datos: el DDL interno.

En el esquema presentado de la arquitectura de un SBD, se observan los niveles de correspondencias , una entre los niveles externo y lógico global y otra entre los niveles lógico global e interno.

La correspondencia lógica/interna especifica la forma en que los artículos y campos lógicos se representan en el nivel interno. Si se cambia la estructura de la vista interna, o sea, si se hace un cambio en el esquema interno, entonces la correspondencia lógica/interna tiene también que cambiar en consecuencia, de modo que el esquema lógico permanezca invariable. En otras palabras, los efectos de estos cambios deben ser aislados por debajo del nivel lógico para que se mantenga la independencia datos.

Existe también una correspondencia externo/lógica entre cada vista externa particular y la vista lógica. Las diferencias que pueden existir entre estos dos niveles son similares a las que pueden existir entre las vistas lógicas y la interna. Por ejemplo, los campos pueden tener diferente tipos de datos, se pueden cambiar los nombres de artículos y campos, múltiples campos lógicos pueden ser combinados en un único campo externo, etc. Puede existir al mismo tiempo cualquier cantidad de vistas externas; cualquier cantidad de usuarios puede compartir una vista externa dada; las diferentes vistas externas se pueden solapar. Algunos sistemas permiten la definición de una vista externa a partir de otra (mediante una correspondencia externa/externa); esta característica es útil cuando varias vista externas están estrechamente relacionadas entre si.

Es importante señalar que en la arquitectura de un sistema de bases de datos también es usual que se considere, como parte de ella, al administrador de la base de datos, quien es la persona o grupo de personas responsable del control total de todo el sistema.

El SGBD interactúa con cada uno de los niveles y las correspondencias entre ellos.

Representación de la información

En el proceso y construcción de todo sistema informativo automatizado, el diseño de la BD ocupa un lugar importante, a tal punto que ésta puede verse como un proceso relativamente independiente dentro del diseño del sistema y compuesto por una serie de etapas. Es por ello que resulta de interés el estudio de los problemas relacionados con el diseño de las bases de datos y la modelación de la información.

Cuando se habla de información, se hace referencia, de forma general, a tres niveles diferentes, tendiéndose a saltar de uno a otro sin establecer una advertencia previa.

1. El primero de estos niveles es el del **MUNDO REAL**, en el que existen entidades u objetos, que no son más que cosas o elementos que existen y están bien diferenciados entre si, que poseen propiedades y entre los cuales se establecen relaciones. Por ejemplo, una silla es una entidad u objeto, un automóvil, un empleado, un profesor, un estudiante, que son cosas concretas; pero también puede ser algo no tangible, como un suceso cualquiera, una cuenta de ahorro, o un concepto abstracto.

   Entre las propiedades que caracterizan a una entidad u objeto pudieran encontrarse el color, el valor monetario, el nombre, etc.

   De las relaciones entre las entidades u objetos hablaremos más adelante.

   La determinación de cierta entidad u objeto correspondiente a un fenómeno o proceso, está muy relacionada con el nivel de abstracción en que se esté realizando el análisis. Así, por ejemplo, si se estudia el comportamiento de un insecto especifico en determinadas condiciones climáticas, las propiedades y relaciones que interesan son de un cierto tipo; sin embargo, si se estuviera realizando un estudio de las diferentes especies de insectos, entonces serían otros los objetos a definir, así como las propiedades que los caracterizarían y las relaciones que se establecerían. Si se estuviera analizando todo el reino animal, serían también otros los objetos a definir, con sus características y propiedades.

2. El segundo nivel es el **dominio de las ideas** (Sistema Objeto) y es en el que se decide la información que debe existir en la BD sobre un fenómeno o proceso del mundo real, o sea, qué información debe almacenarse. En este nivel es donde realmente se define el contenido informativo que representará al fenómeno, proceso o ente de la realidad objetiva que se está analizando. De modo que, en este nivel, se definen cuáles objetos y qué propiedades de éstos son representativas y sobre los cuales es necesario almacenar información.

En este nivel es donde se trabaja con los conceptos más importantes del modelo de datos, que establecen la relación entre el mundo real y la información almacenada físicamente en la base de datos:

Campo o atributo: es la unidad menor de información sobre un objeto (almacenada en la base) y representa una propiedad de un objeto (por ejemplo, el color). Sin embargo, hay que distinguir entre el **nombre o tipo del atributo** y el **valor del atributo**, ya que un nombre de atributo puede tomar diferentes valores sobre un cierto conjunto que se denomina **dominio**. A un valor de un atributo determinado o definido en el dominio dado, en un cierto momento del tiempo, se denomina **ocurrencia** del atributo.

Ejemplo:

Atributo	Color	Cat_Doc
Dominio	{Azul,Rojo,Verde,...}	{PT, PA, A, I}
Ocurrencia	Rojo	A

Ahora bien, una colección identificable de campos asociados es un **artículo o registro** y representa un objeto con sus propiedades. Una vez más, es imprescindible distinguir entre **nombre o tipo de artículo y ocurrencia de artículo**. Una **ocurrencia de artículo o tuplo o uplo** consiste en un grupo de ocurrencias de campos relacionados, representando una asociación entre ellos. Por ejemplo, tenemos un artículo correspondiente al objeto profesor, en un fenómeno o proceso de la realidad que pretenda representar el comportamiento de una Facultad. El nombre o tipo de artículo puede ser PROFESOR, que esté formado por los siguientes tipos de campos o atributos:

NUM_IDENT: número de identidad del profesor
NOM_PROF: nombre del profesor
CAT_DOC : categoría docente del profesor
DPTO : departamento docente al que pertenece el profesor

Una ocurrencia de este artículo puede ser:

45112801731 Hernandez Juan PA Computación.

Un fichero o archivo o conjunto de datos puede ser definido como un conjunto de ocurrencias de un mismo tipo de artículo.

En la práctica, a menudo interesan las colecciones o conjuntos de objetos similares, necesitándose almacenar la información de las mismas propiedades para cada uno de ellos, por ejemplo, el conjunto de profesores de la Facultad.

Entonces, una **base de datos** contendrá muchas ocurrencias de cada uno de los tipos de artículos, lo que implica que la base de datos, por supuesto, también contendrán muchas ocurrencias de los distintos tipos de atributos.

Uno de los momentos cruciales en el diseño de un fenómeno de la realidad objetiva que se concreta en una base de datos es, precisamente, la selección de los conjuntos de objetos y sus propiedades.

Además, existe otro concepto muy importante en este nivel, que es el concepto de **llave o clave**: un atributo o conjunto de atributos de un artículo que define que cada ocurrencia de artículo de la base de datos sea único. En principio, cada artículo tiene una llave, ya que se tiene como hipótesis que cada elemento u ocurrencia del artículo es diferente de las demás. Por ejemplo, número de identidad del trabajador.

3. El tercer nivel es de los datos propiamente dichos, representados mediante cadenas de caracteres o de bits.

En este nivel es necesario tener en cuenta la diferencia entre tipo de dato y valor del dato. El tipo de dato corresponde a un atributo o tipo de atributo, que está asociado a un tipo de artículo

correspondiente, mientras que el valor corresponde a una ocurrencia del atributo. Sin embargo, una colección de bits o caracteres que representa un único valor de datos y que puede existir independientemente de cualquier información que se almacena, adquiere significado sólo cuando se le asocia a un tipo de atributo. Se puede, por ejemplo, almacenar permanentemente los valores ROJO, AZUL, VERDE, etc. y asociarlo en un momento determinado a un tipo de atributo a través de los valores que toma, representando una ocurrencia en un tuplo.

Es importante notar que, en general, habrá asociaciones o relaciones enlazando las entidades básicas.

Estos enlaces se pueden establecer entre diferentes objetos o tipos de artículos o entre un mismo tipo de artículo. Por ejemplo, cuando se tiene una base de datos formada por dos tipos de objetos: SUMINISTRADOR y PRODUCTO, se puede tener la relación "CANTIDAD", que establece la cantidad de cada producto que abastece un suministrador dado. Otro ejemplo pudiera ser con el artículo PERSONA, sobre el que se pudiera representar la relación "SER MADRE DE", que no es más que una relación que se establece entre elementos de un mismo tipo de artículo.

Es necesario profundizar acerca de los diferentes tipos de relaciones que pueden ocurrir en la práctica.

Relaciones de correspondencia:

Es necesario establecer la correspondencia que existe entre los datos; esta relación puede ser simple o compleja.

Por relación simple se entiende una correspondencia biunívoca (de uno a uno) entre las ocurrencias de los objetos, o sea, entre los artículos. Si, por ejemplo, los atributos son Nro_Identidad y nombre del profesor la correspondencia entre ellos es simple: a cada nombre corresponde un número de identidad y viceversa.

Relación de uno a uno
(1: 1)

Nombre **<--------->** Nro_Identidad

Si los atributos son Nro_identidad y departamento, la relación es más complicada, porque a cada departamento corresponden varios empleados. La terminología corriente expresa que la correspondencia de empleado a departamento es simple (cada empleado es miembro de un único departamento), mientras que la correspondencia de departamento a empleado es compleja, pues cada departamento tiene, por lo general, muchos empleados.

Relación de uno a muchos
(1: M)

Nro_Dpto. **<------>>** Nro_Identidad

Hay cuatro tipos de relaciones posibles entre dos tipos de artículos A y B: La correspondencia de A a B puede ser simple y la recíproca compleja. La correspondencia de A a B puede ser compleja y la recíproca simple. Ambas correspondencias pueden ser complejas o ambas pueden ser simples, es decir.

A <<-----> B **A <----->> B**

A <<----->> B **A <-----> B**

Un ejemplo donde ambas correspondencias son complejas, lo es la relación que se establece entre PROFESOR y ESTUDIANTE por la impartición de clases, ya que un profesor puede impartir clases a varios estudiantes, pero, a su vez, un estudiante puede recibir clases de varios profesores:

Relación de muchos a muchos
(N : M)
PROFESOR <<----->> ESTUDIANTE

Las relaciones pueden tener diferentes características:

- Aunque la mayoría de las relaciones asocian dos tipos de entidades, éste no es siempre el caso. Por ejemplo, PROFESOR_HORARIO_ESTUDIANTE. Esto podría representar el hecho de que un profesor imparte clases a una cierta hora a un cierto estudiante. Esto no es lo mismo que la

combinación PROFESOR_HORARIO y HORARIO_ESTUDIANTE, ya que la información de que "el profesor P5 imparte clases en el horario H1 al estudiante E4" dice más que la combinación "el profesor P5 imparte clases en el horario H1" y "el estudiante E4 recibe clases en el horario H1"

- Las relaciones pueden establecerse entre un mismo tipo de entidad. Por ejemplo, una asociación entre un profesor y otro puede venir dada por el hecho de que un profesor sea el jefe de otros profesores.

- Es importante señalar que una asociación entre entidades puede ser considerada en sí como una entidad, ya que una relación se puede ver como un objeto bien diferenciado sobre el cual se desea almacenar información.

Entonces, un modelo de datos no es más que la representación de un fenómeno de la realidad objetiva a través de los objetos, sus propiedades y las relaciones que se establecen entre ellos.

Tema 2

El Modelo Relacional

Contenido:

- **Características del Modelo Conceptual.**
- **Modelo Entidad-Relación y su representación gráfica.**
- **Operaciones sobre los objetos básicos del Modelo Entidad-Relación**
- **El Modelo Relacional y sus componentes**

Características del Modelo Conceptual

El proceso de diseño de la BD transita a través de una serie de pasos en los cuales se va avanzando de un nivel de abstracción menor a otro más profundo, mediante la elaboración de una sucesión de modelos. En los últimos años se ha generalizado la concepción del diseño de las BD propuestas por el grupo ANSI/SPARC, la cual constituye, al mismo tiempo, una arquitectura para los SBD, tal y como la acabamos de estudiar.

Hemos visto en esta arquitectura que cada nivel de la misma es una cierta forma de representación abstracta de la información y una de las funciones más importantes del SGBD consiste precisamente en permitirle al usuario la interacción con los datos en estos términos abstractos, en lugar de tenerlo que hacer directamente con la forma en que esos datos están físicamente almacenados. Es por ello que, al acometerse la tarea de diseño de una BD, la atención se debe centrar en el aspecto lógico de la información, ya que los detalles relacionados con el almacenamiento físico son parte de todo SGBD comercial que se utilice, y por tanto, no pueden ser modificados.

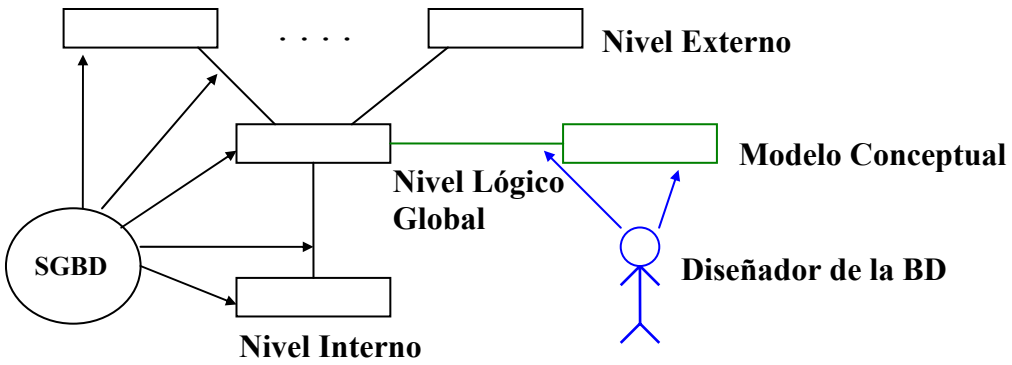
Los SGBD existentes utilizan diferentes modelos de datos para la representación en el nivel lógico global. Son comunes a todos ellos las siguientes características:

1. La representación de la información se basa en el uso de determinadas estructuras de datos que poseen una capacidad descriptiva limitada (sólo diferencian un rasgo semántico: el tipo de proyección (**1 : 1** , **1 : N** , **N : M**).

2. Utilizan una terminología que no es familiar al usuario del sistema, por lo que dificultan la comunicación usuario- diseñador.

Además, cada uno de estos modelos está vinculado con un tipo particular de SGBD.

Por todo ello, es necesario tratar con otro tipo de modelo cuando se aborda el problema del diseño de las BD, el cual debe superar los problemas anteriores y constituye un nivel de abstracción intermedio entre la realidad informativa y el nivel lógico global de la arquitectura. A este nuevo tipo de modelo se le denomina modelo conceptual. O sea, el modelo conceptual se define exteriormente al SGBD, realizándose manualmente la transformación entre el modelo conceptual y el lógico global.



El proceso de modelación conceptual es denominado también modelación semántica, ya que con estos modelos se pretende reflejar en mayor medida la semántica, el significado de los datos y sus interrelaciones.

El Modelo Entidad-Relación (MER)

Este modelo fue propuesto en 1976 y ha encontrado una amplia aceptación como instrumento para modelar el mundo real en el proceso de diseño de las bases de datos.

El **MER** opera con los conceptos de entidad y relación que vimos anteriormente.

Las entidades (ocurrencia de entidades) se clasifican en distintos conjuntos entidades (entidades). Ejemplos: "empleado", "departamento", etc. Existirá un predicado asociado con cada conjunto entidad (entidad) que permitirá comparar si una entidad arbitraria pertenece a un conjunto dado. Las entidades pueden pertenecer a más de un conjunto, o sea, los conjuntos entidad no son mutuamente disjuntos. Por ejemplo: Una entidad del conjunto "mujeres" también pertenece al conjunto "personas".

Un conjunto relación es una relación matemática entre n entidades.

$$\{ (e1, e2, ...en) \mid e1 \in E1, e2 \in E2, ..., en \in En \}$$

y cada tuplo (e1, e2, ...en) es una relación, donde las Ei y ei no tienen que ser necesariamente diferentes. El rol de una entidad en una relación expresa la función que desempeña dicha entidad en la relación. En el conjunto de relación "matrimonio" definido entre entidades del conjunto "personas", o sea "matrimonio" { [e1, e2] | e1 ∈ "persona", e2 ∈ "persona" }, el primer elemento en el tuplo puede aparecer en el rol de "esposo" y el segundo, en el rol de "esposa".

Información adicional sobre una entidad (además de los predicados y las relaciones) se obtiene mediante un conjunto de pares atributo-valor (atributo) asociados con la entidad. Ejemplos de valores son "rojo", "3", "Juan", etc. y ellos se clasifican en conjuntos valor mutuamente disjuntos, tales como "color", "edad", etc.

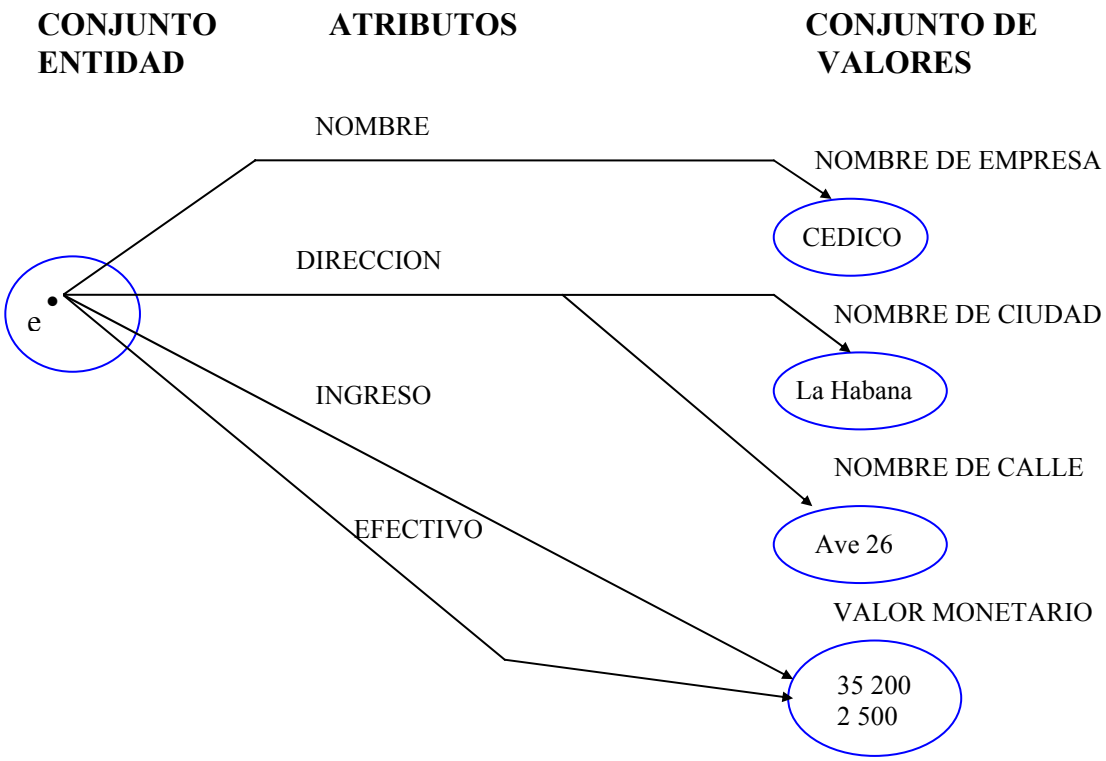
Un valor de un conjunto puede ser equivalente a otro valor en un conjunto diferente. Por ejemplo, "100" en el conjunto valor "centímetros" es equivalente a "1" en el conjunto valor "metro".

Un atributo se define en el MER como una función matemática que establece una correspondencia desde un conjunto entidad o conjunto relación hacia un conjunto valor o un producto cartesiano de conjuntos valor:

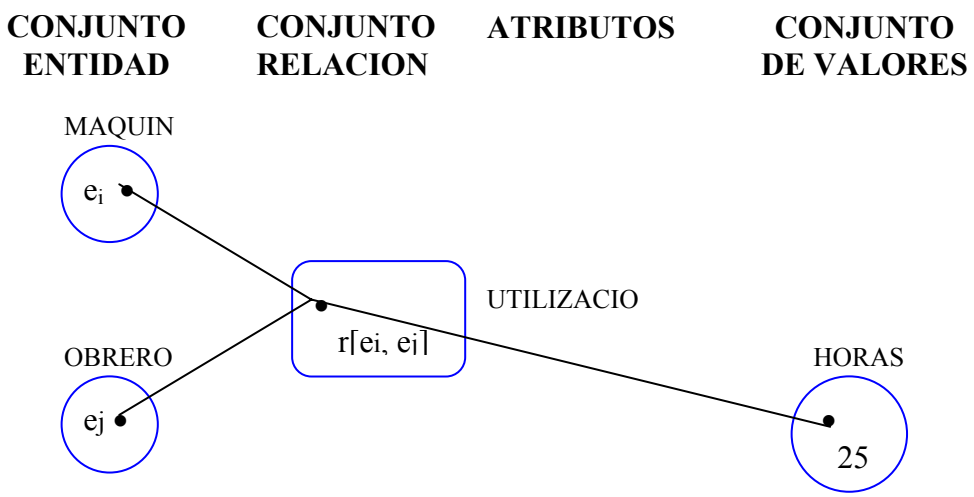
$atrib_1$: E_i ---> V_{i1} x V_{i2} x .....x V_{in}

$atrib_2$: R_i ---> V_{i1} x V_{i2} x .....x V_{in}

En la figura siguiente se muestran los atributos definidos en el conjunto entidad (entidad) EMPRESA. El atributo NOMBRE hace corresponder a las entidades empresas (ocurrencias de empresa) con elementos del conjunto valor (dominio) NOMBRE DE EMPRESA. El atributo DIRECCIÓN establece una correspondencia desde el conjunto entidad (entidad) EMPRESA hacia el par de conjuntos valor NOMBRE DE CIUDAD, NOMBRE DE CALLE. INGRESO Y EFECTIVO establecen ambos una correspondencia desde el conjunto entidad (entidad) EMPRESA hacia el conjunto valor VALOR MONETARIO. Nótese que un atributo se define siempre como una función, por lo que siempre hace corresponder a una entidad dada (ocurrencia) con un único valor de una tupla, pues se define un producto cartesiano de conjuntos valor (dominios).



Las relaciones pueden también tener atributos. En la figura siguiente, el atributo UTILIZACIÓN define el número de horas que un obrero específico e_j usa una máquina e_i y constituye un atributo de la relación correspondiente. El no es ni un atributo del OBRERO ni de la MAQUINA, ya que su significado depende de la relación entre ellos dos.



Es importante destacar las siguientes características de los atributos en este modelo:

1. Los atributos sólo son correspondencias funcionales. Así, por ejemplo, si tenemos el conjunto entidad (entidad) AUTOMÓVIL y el atributo COLOR, el hecho de que un auto pueda tener más de un color no se puede representar como un atributo en este modelo.

2. El único hecho que puede ser registrado sobre los valores en este modelo es su pertenencia a un conjunto valor. Si se desea representar otra propiedad, el valor tiene que ser convertido en una entidad. Por ejemplo, si queremos registrar la longitud de onda de cada color no podemos hacerlo en el MER, sino convirtiendo el conjunto de valores COLOR en un conjunto entidad (entidad).

Representación gráfica del MER

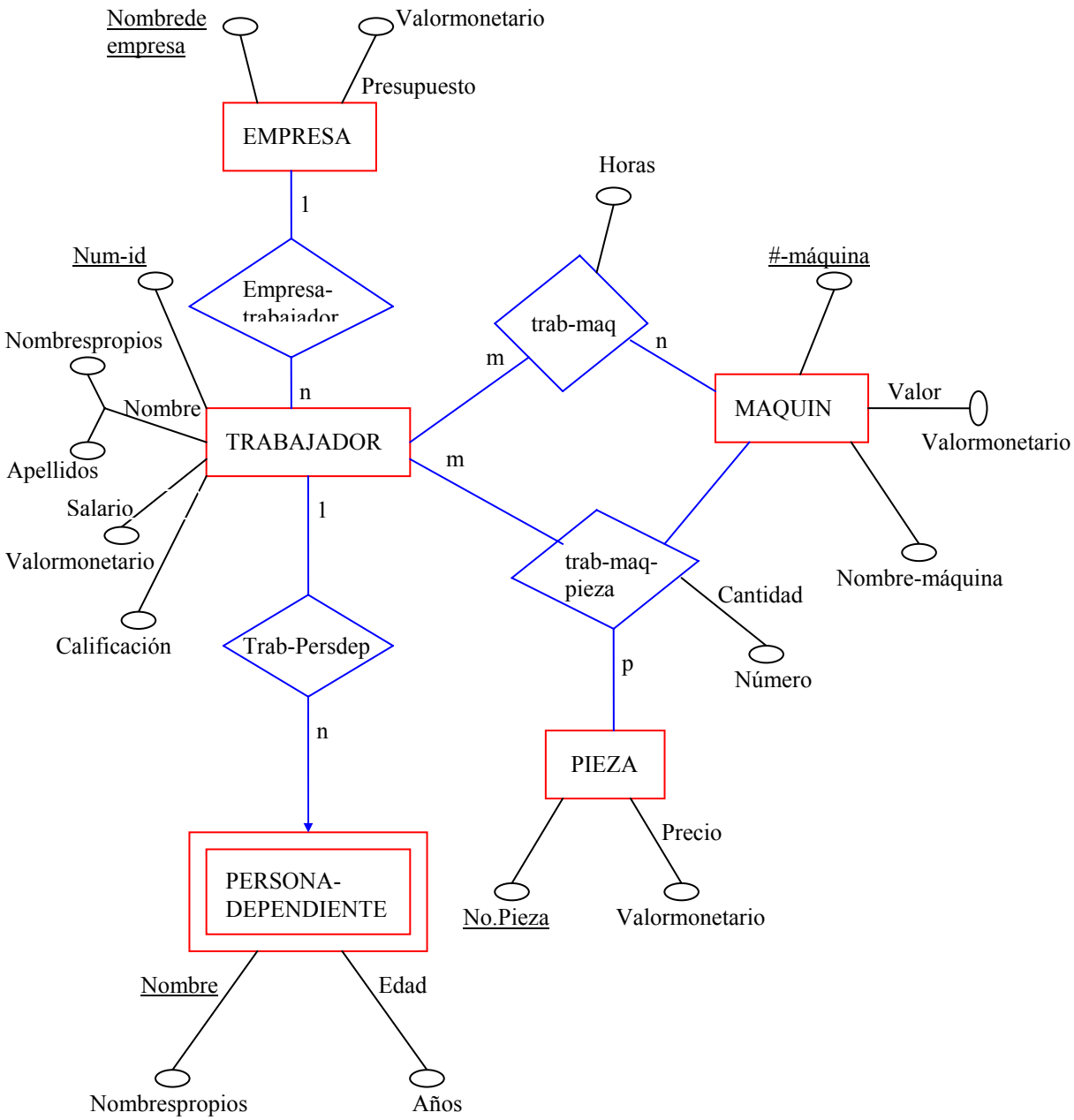
El MER tiene asociada una representación gráfica denominada Diagrama Entidad-Relación (DER).

En un DER, cada entidad se representa mediante un rectángulo, cada relación mediante un rombo y cada dominio mediante un círculo. Mediante líneas se conectan las entidades con las relaciones, igual que las entidades con los dominios, representando a los atributos.

Los atributos llaves se representan subrayando el correspondiente conjunto de valores.

En ocasiones, una entidad no puede ser identificada únicamente por el valor de sus propios atributos. En estos casos, se utilizan conjuntamente las relaciones con los atributos para lograr la requerida identificación unívoca. Estas entidades reciben el nombre de entidades débiles y se representan en el DER con un doble rectángulo. El MER restringe las relaciones a usar para identificar las entidades débiles a relaciones binarias del tipo 1:n. Así, por ejemplo, una ocurrencia de "trabajador" puede tener n ocurrencias "persona-dependiente" asociadas, donde además, la existencia de las ocurrencias en la segunda entidad depende de la existencia de una ocurrencia que le corresponda en la primera entidad. Por ejemplo, en el modelo habrá personas dependientes de un trabajador sólo si ese trabajador existe. Para indicar esa dependencia en la existencia se usa una saeta en el DER. La llave de una entidad débil se forma combinando la llave de la entidad regular que la determina con algún otro atributo que defina unívocamente cada entidad débil asociada a una entidad regular dada. (Una entidad se denomina regular si no es débil).

En una relación, la llave es la combinación de las llaves de todas las entidades asociadas. Para cada relación se determina su tipo (simple o complejo) y en el DER se escribe el tipo de correspondencia. Por ejemplo, una empresa puede tener varios (n) trabajadores asociados y un trabajador pertenece a una sola empresa (1). En la relación Trabajador-Máquina-Pieza, un trabajador puede trabajar en n máquinas, produciendo p piezas, o una pieza puede ser producida por m trabajadores en n máquinas. Aquí, m, n y p no identifican un número específico, sino solamente el tipo de correspondencia que se establece en la relación.


Nombrede empresa
Valormonetario
Presupuesto
EMPRESA
1
Num-id
Empresa-trabajador
n
Nombrespropios
Nombre
Apellidos
TRABAJADOR
Salario
Valormonetario
Calificación
1
Trab-Persdep
n
PERSONA-DEPENDIENTE
Nombre
Edad
Nombrespropios
Años
Horas
trab-maq
m
n
#-máquina
MAQUIN
Valor
Valormonetario
Nombre-máquina
m
trab-maq-pieza
Cantidad
Número
p
PIEZA
Precio
No.Pieza
Valormonetario

Operaciones sobre los objetos básicos del MER

Es posible extender la capacidad semántica del MER aplicando sobre sus objetos básicos (entidad y relación) diferentes operaciones:

1. **Agregación:** Construye una nueva entidad sobre la base de una relación.

2. **Generalización:** Permite formar una nueva entidad, mediante la unión de otras entidades. El proceso inverso se denomina especialización y divide una entidad en cierto número de otras entidades.

3**. Agrupamiento:** Define una nueva entidad, donde cada ocurrencia es un grupo de ocurrencias de la entidad fuente.

A las entidades, relaciones y conjuntos definidos hasta ahora les llamaremos tipos básicos para distinguirlos de los nuevos tipos de datos que se obtendrán con las operaciones anteriores.

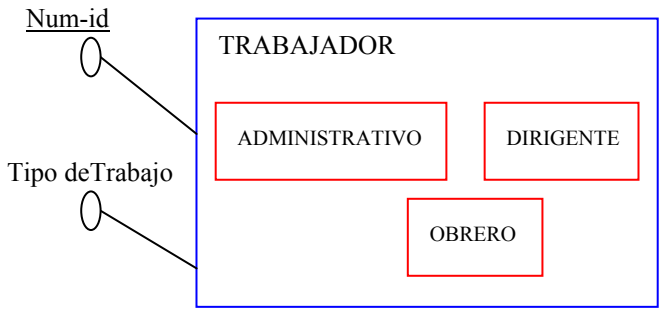
Veamos cada una de las operaciones:

1. **Generalización / Especialización**

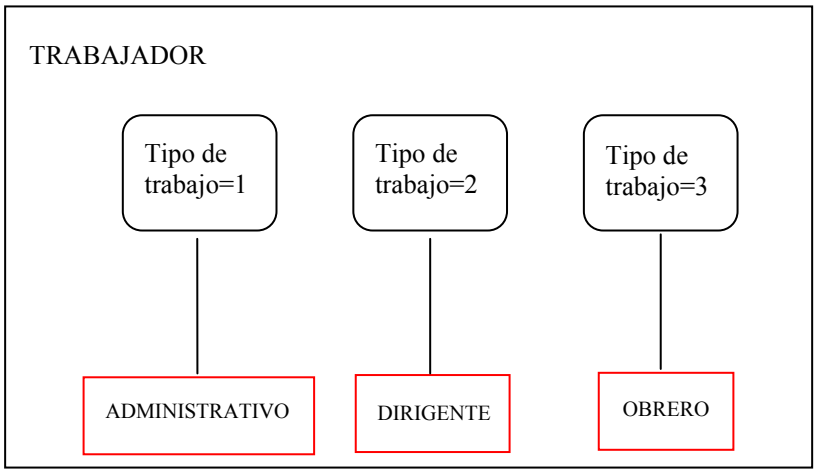
Si T1, T2, ...Tn son conjuntos entidad (que pueden a su vez ser resultado de una generalización), la generalización define un nuevo conjunto entidad T con el sgte significado:

$$\mathbf{T = \{ t \mid t \in Ti\ ,\ 1 \leq i \leq n\}}$$

o sea, existe para cada entidad (ocurrencia) t en T al menos un conjunto Ti que contiene a esa entidad (ocurrencia). Por ejemplo, en el DER anterior, puede ser necesario distinguir los trabajadores de una empresa de acuerdo a su ocupación como obreros, dirigentes y administrativos. Esto no puede ser representado en el modelo anterior y sólo a través del hecho de que el conjunto entidad (entidad) "obrero", por ejemplo, es siempre (o sea, en todo momento) un subconjunto del conjunto entidad (entidad) "trabajador", podemos deducir cierta clase de dependencia entre los dos tipos. Usando la generalización podemos obtener un nuevo diagrama como se muestra parcialmente en la figura siguiente:



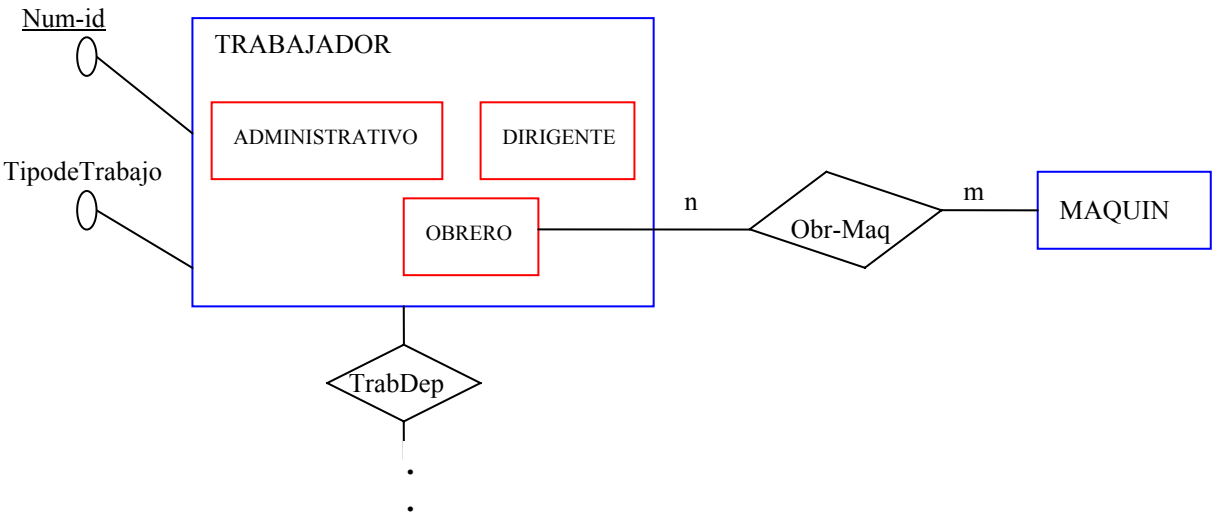
Nótese que hemos introducido un nuevo par adicional atributo-valor (atributo) para el conjunto trabajador. Este atributo nos permite distinguir entre los miembros de diferentes clases de trabajadores. Si tenemos un conjunto entidad (entidad) TRABAJADOR y queremos usar la operación de Especialización como inversa a la generalización, tenemos que especificar "roles" en el modelo, o sea, reglas que definan cuándo una entidad (ocurrencia) TRABAJADOR pertenece a uno u otro componente del conjunto entidad (entidad). Entonces la representación de esta operación en el DER se generaliza como se muestra en la figura siguiente:



Si por cada entidad Trabajador nosotros podemos siempre deducir a cuál conjunto entidad componente pertenece usando alguna propiedad ya representada, entonces no es necesario introducir un nuevo par atributo-valor (atributo) Tipo de Trabajo.

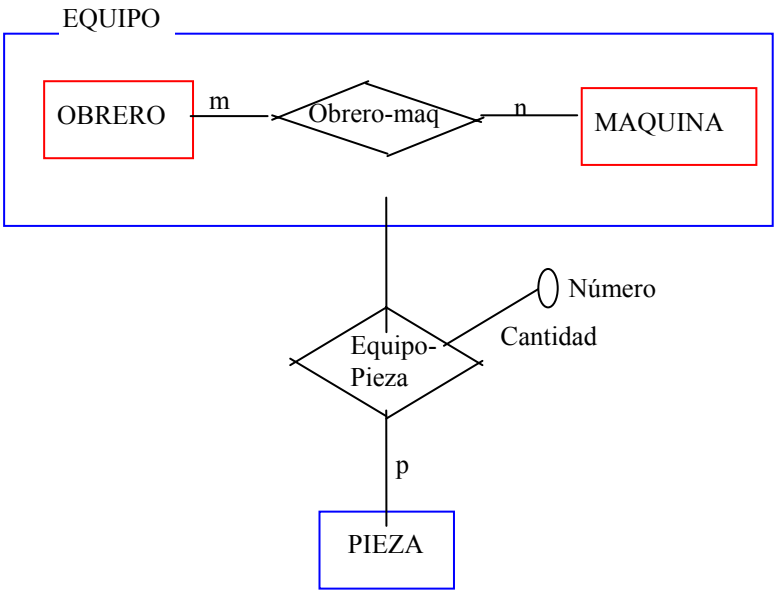
Las reglas que definen la especialización de un conjunto entidad (entidad) se denominan "caracterizaciones". Por ejemplo, Tipo de Trabajo = 1 es la caracterización del conjunto entidad (entidad) Administrativo dentro del conjunto entidad (entidad) Trabajador.

En una Generalización / Especialización, los atributos y relaciones del conjunto entidad "generalizado" (entidad generalizada) son heredados por los conjuntos entidad componentes (entidades especializadas). Además, se pueden definir nuevos atributos y relaciones para cada conjunto entidad especializada. Por ejemplo, la relación Obrero-Máquina se define ahora sólo para la entidad especializada Obrero, componente de la entidad generalizada Trabajador:



2. **Agregación**

Obsérvese en el ejemplo que representa la situación de la producción en las empresas, que la relación ternaria Trab-Máq-Pieza representa la idea de que una actividad en la empresa se describe en términos de "un obrero en alguna máquina produce una pieza dada en alguna cantidad específica". Sin embargo, la misma situación puede ser vista de forma algo diferente. En la empresa las máquinas pueden estar asignadas a los obreros y estos "equipos", producir piezas en cierta cantidad. En el MER original esta situación no hubiera podido ser modelada correctamente, ya que una relación no puede relacionarse con otra relación o entidad. Con la operación de Agregación esta situación se resuelve fácilmente, tal y como se muestra en la figura siguiente:



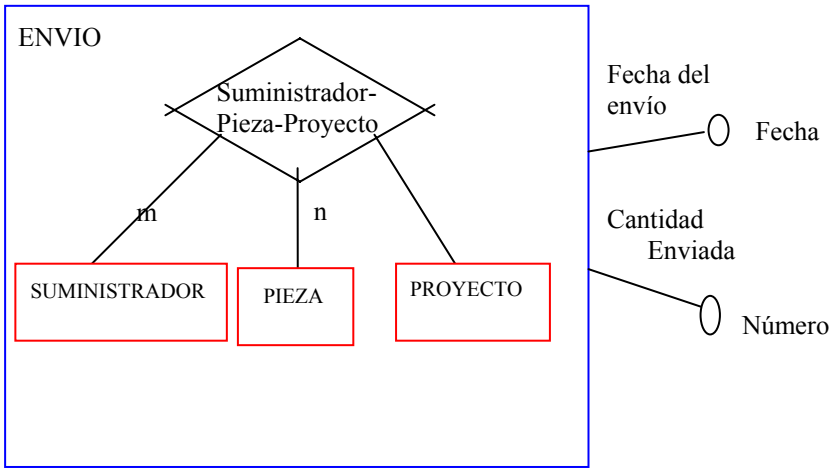
La agregación se define de la siguiente forma:

Si T1, T2, ...Tn son conjuntos entidad (entidad), la operación define un nuevo conjunto entidad (entidad) T con el significado sgte.:

$$T = \{t \mid \exists \quad t1, t2, ...tn\ (t1 \in T1 \wedge t2 \in T2 \ldots \wedge tn \in Tn \wedge <t1,t2,..tn> = t)\}$$

O sea, las nuevas entidades (ocurrencias) se forman como tuplas de entidades (ocurrencias) a partir de los conjuntos entidad (entidades) componentes. Para que la operación tenga sentido, los conjuntos entidad (entidades) T1,T2,...Tn tienen que formar parte en alguna relación común y esa relación siempre será incluida en la representación del conjunto entidad (entidad) generado.

Al nuevo conjunto entidad (entidad) se le pueden asignar atributos. También puede tomar parte en cualquier relación. Otro ejemplo de Agregación se muestra a continuación:

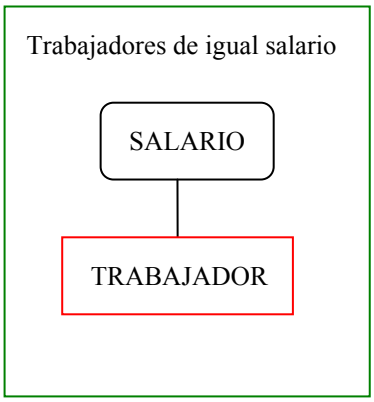


El nuevo conjunto entidad (entidad) ENVÍO se define como una agregación de tres conjuntos entidad (entidades): Suministrador, Pieza y Proyecto con los nuevos atributos Fecha de envío y Cantidad enviada. Hay una diferencia importante entre estos dos atributos: Esta claro que la Fecha de envío no puede pertenecer a ninguno de los conjuntos entidad (entidad) componentes, sin embargo, la Cantidad enviada se refiere claramente a las piezas. Diremos entonces, que la Cantidad enviada es una "caracterización" del conjunto entidad (entidad) PIEZA con respecto al ENVÍO.

3. **Agrupamiento**

Si T designa a algún conjunto entidad (entidad) y T1, T2 ...Tn son bien conjuntos valor (dominios) asociados con T o conjuntos entidad (entidades) relacionados con T vía alguna relación, entonces el operador de Agrupamiento construye un nuevo conjunto entidad agrupado (entidad agrupada o agrupamiento) Tg donde cada elemento es un conjunto de entidades (ocurrencias) de T, tales que, dentro de cada uno de tales conjuntos, todas las entidades (ocurrencias) tienen los mismos valores y entidades relacionadas desde los conjuntos entidad (entidad) T1, T2,..Tn asociados. Los tipos T1, T2,...Tn se llamarán tipos índice y T se llamará base.

Por ejemplo, podemos usar el par Salario-Valor Monetario del DER anterior para formar un conjunto entidad (entidad) agrupado a partir del conjunto entidad Trabajador. Cada entidad (ocurrencia) Trabajador dentro de un grupo, que representa a una entidad (ocurrencia) en Trabajadores de igual salario, tiene el mismo valor del salario.



Para el MER, incluyendo las tres operaciones estudiadas pueden plantearse una serie de restricciones de integridad:

1. Al aplicar la generalización/especialización, una entidad puede pertenecer a una jerarquía de diferentes conjuntos entidad (entidades). Por ejemplo, los conjuntos entidad (entidades) PERSONA, TRABAJADOR, OBRERO forman una jerarquía de conjuntos (entidades) sucesivamente más especializados. Entonces, una entidad existente en un nivel dado, tiene que existir en todos los niveles superiores. De forma inversa, si una entidad se elimina de un conjunto en un nivel i, debe ser eliminada también en los niveles más bajos.

2. La agregación constituye un conjunto entidad (entidad agregada) sobre la base de una relación, por lo que dicho conjunto se comportará de forma similar a como se comporta la relación. Entonces, para que el conjunto agregado exista, deben existir todos los conjuntos entidad (entidades) que toman parte en la relación. Lo inverso no tiene que ocurrir necesariamente, ya que, por ejemplo, en el caso visto del ENVÍO, pueden existir suministradores que no abastezcan a ningún proyecto, sino que se registran como tales porque en determinado momento pudieran estar activos. Desde luego, si la política de la organización es tal que un suministrador se considera como tal sólo si realmente suministra piezas a algún proyecto, entonces la existencia de la entidad agregación ENVÍO es indispensable para la existencia del conjunto entidad SUMINISTRADOR.

3. Al aplicar el agrupamiento, por lo general la existencia de todos los componentes del conj. índice es necesaria para que exista la entidad agrupada. Por otra parte la base es indispensable sólo en el

sentido de que para que exista cada entidad agrupada en el conj. de entidad obtenido. al menos tiene que existir una entidad en la base. Lo inverso no se requiere, o sea, no es necesario que cada entidad en el conjunto base sea miembro de alguna entidad en el conjunto agrupado.

EL Modelo Relacional y sus Componentes

Uno de los modelos matemáticos más importantes, actuales y con mayores perspectivas para la representación de las bases de datos, es el enfoque relacional. Este se basa en la teoría matemática de las relaciones, suministrándose con ello una fundamentación teórica que permite aplicar todos los resultados de dicha teoría a problemas tales como el diseño de sublenguajes de datos y otros.

Componente Estructural del Modelo Relacional

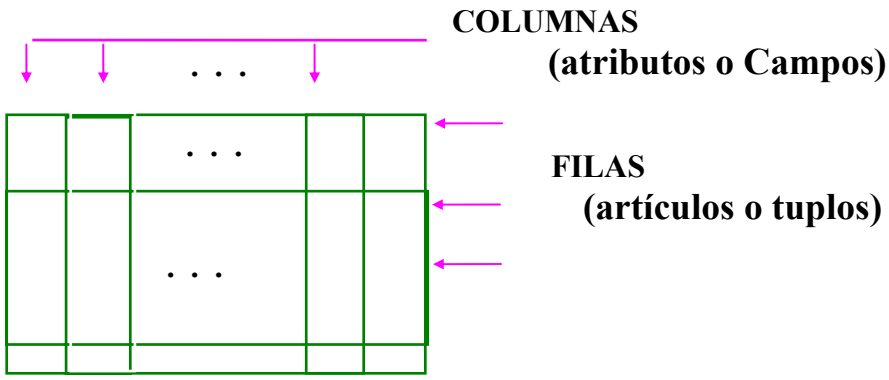
La única componente estructural del modelo relacional es la relación o tabla. El término relación se puede definir matemáticamente como sigue:

Definición: **Relación**

Dados los conjuntos de dominios D1, D2, ...Dn (no necesariamente distintos), R es una relación sobre esos n conjuntos si está constituida por un conjunto de n-tuplos ordenados d1,d2,...dn tales que d1 $\in$ D1, d2 $\in$ D2, ..., dn $\in$ Dn.

Los conjuntos D1, D2, ...Dn se llaman **dominios de R** y n constituye el grado de la relación.

Las relaciones son representadas mediante tablas bidimensionales donde cada fila representa un n-tuplo o articulo y cada columna un atributo o campo.



En el modelo relacional, tanto los objetos o entidades, como las relaciones que se establecen entre ellos, se representan a través de las "**tablas**", que en la terminología relacional se denominan **relaciones**.

Cada **relación** está compuesta de filas (las ocurrencias de las entidades) y se les denomina, en la terminología relacional, como **tuplos** o **uplos** (en realidad, n-tuplos, pero en muchos casos se suprime la n cuando no existe posibilidad de confusión).

También la relación está compuesta por columnas (los atributos o campos) que toman valores en sus respectivos dominios.

Denotándose la relación por : **$R (D_1, D_2, D_3, \ldots\ldots\ldots\ldots D_n)$**. Entonces se tendrá que:

$$R (D_1, D_2, D_3, \ldots\ldots\ldots\ldots D_n) = \{ (d_1, d_2, \ldots\ldots.., d_n) \mid d_1 \in D_1 , d_2 \in D_2 , \ldots\ldots\ldots, d_n \in D_n \}$$

Cada valor dentro de la relación (cada valor de un atributo) es un dato atómico (o elemental), es decir, no descomponible; por ejemplo: un número, una cadena de caracteres. En otras palabras, en cada posición (fila, columna) existe un solo valor, nunca un conjunto de valores.

Propiedades de una Relación

Grado.- Cantidad de dominios en que esta definida la relación.

Esquema.- Lo compone el nombre de la relación y el de sus atributos.

Ejemplo: Trabajadores (CI, Edad, Sexo, Ecivil)

Intensión.- Compuesto por el Esquema más un conjunto de Predicados, que nos permite determinar cuando un tuplo pertenece o no a la relación.

Ejemplo: Estudiantes ({CI, Edad }, { Matriculados en Nivel Superior})

La intensión es una propiedad estática de la relación.

Extensión.- Cantidad de tuplos que pertenecen a la relación en un tiempo t.

La extensión es una propiedad dinámica de la relación.

Llave Candidata.- K es una **llave candidata** de la relación **R (D_1, D_2, …… D_n } ,** si y sólo si:

1. $K \subseteq \{ D_{1.}, D_2, \ldots\ldots\ldots D_n \}$

2. Permite identificar sin ambigüedad a cada tuplo de la relación, es decir sea única, o sea no podrá existir dos tuplos de R con igual valor de K. Esto se expresa:

$$K \longrightarrow D_i \quad \forall\ i$$

3. Sea minimal, es decir, no existe $K_1 \subset K$ que cumpla 1.

Nota: Aquellos K que sólo cumplen 1 se le denominan **Super LLave**

Llave Primaria.- Es aquella llave candidata que sea tomada como llave de la relación.

Toda relación tendrá al menos una llave candidata $\{ D_1, D_2, \ldots\ldots\ldots\ldots D_n\}$

Orden.-

- El orden de las filas no es significativo.
- El orden de las columnas no es significativo.

Siendo rigurosos, el orden de las columnas sí es significativo, pues representa el orden de los dominios implicados, pero como siempre nos referimos a una columna por su nombre y nunca por su posición relativa.

Componentes restrictivos del modelo relacional

1. Toda Relación tiene una llave primaria, la que sus valores no pueden estar indefinidos, ni ser repetitivos. Este tipo de integridad del Modelo Relacional se denomina Integridad de Entidades.

2. Donde quiera que aparezca un atributo (o combinación de ellos) en una relación R_2 , cuyos valores casen con la **llave primaria** de una relación R_1 , entonces cada valor de esos atributos d de R_2 tiene que ser igual al valor de la **llave primaria** en alguna tupla de R_1. Este tipo de integridad del Modelo Relacional es llamado **Integridad Referencial.**

Ejemplo:

Veamos cómo nuestro ejemplo de **Suministrador** y **Producto** puede ser representado fácil y claramente mediante el modelo relacional.

Suministrador (NSuministrador, Nombre, Tipo, Municipio)

Producto (NProducto, Nombre, Precio Unitario, Peso)

Se conoce la cantidad de un determinado producto que suministra un suministrador dado.

SUMINISTRADOR

SNUM	SNOM	TIPO	MUN
S1	PÉREZ	30	CERRO
S2	RAMOS	10	PLAZA
S3	ARENAS	20	CERRO
S4	VALLE	20	PLAYA
S5	LÓPEZ	15	PLAYA

PRODUCTO

PNUM	PNOM	PRECIO	PESO
P1	CLAVO	0.10	12
P2	TUERCA	0.15	17
P3	MARTILO	3.50	80
P4	TORNILLO	0.20	10
P5	ALICATE	2.00	50
P6	SERRUCHO	4.00	90

SP

S	P	CANT
S1	P1	3
S1	P2	2
S1	P3	4
S1	P4	2
S1	P5	1
S1	P6	1
S2	P1	3
S2	P2	4
S3	P3	4
S3	P5	2
S4	P2	2
S4	P4	3
S4	P5	4

Con estas 3 tablas se tiene todo el modelo representado.

Una de la principales ventajas del Modelo Relacional es su simplicidad, pues el usuario formula sus demandas en términos del contenido informativo de la Base Datos, sin tener que atender a las complejidades de la realización del sistema, lo que implica gran independencia de los datos.

- La información se maneja en forma de tablas, lo que constituye una manera familiar de representarla.

Veamos en el modelo del **SUMINISTRADOR-PRODUCTO** visto anteriormente un ejemplo de cada tipo de operación de actualización:

Inserción: Añadir un producto **P7**, se agregaría una nueva ocurrencia en la tabla **PRODUCTO**. Lo cual será posible hacerlo aunque ningún suministrador lo suministre.

Supresión: Se puede eliminar el suministrador **S1** sin perder el producto **P6**, a pesar de que es el único suministrador que lo suministra sea **S1**.

Modificación: Se puede cambiar el precio del producto P2 sin necesidad de búsquedas adicionales ni posibles inconsistencias.

La Desventajas del Modelo Relacional consiste en la dificultad de lograr productividad adecuada de los sistemas, ya que no se fabrican los medios técnicos idóneos, tales como las memorias asociativas, siendo necesario simular este proceso, pero, en realidad, la eficiencia y productividad de los sistemas actuales resultan realmente satisfactorias.

Ejemplo de SGBD relacionales

Query By Example (QBE) (IBM)
DBase, DataEase, FoxPro, Visual FoxPro, Clipper, Informix, MSAccess (micros)

Componentes Operacionales del Modelo Relacional.

En el modelo relacional el resultado de una demanda es también una relación. Sólo tiene que indicar qué relación desea recuperar. Las diversas formas de hacer las recuperaciones dan lugar a los lenguajes relacionales cuyas formas más representativas son:

- **Álgebra relacional** (basado en las operaciones del álgebra de relaciones).
- **Cálculo relacional** (basado en el cálculo de predicados)

El Álgebra Relacional y sus operadores.

Cada operador del álgebra relacional toma una o dos relaciones como entrada y produce una nueva relación como salida. Originalmente se definen ocho operadores.

- Los operadores tradicionales conocidos de la teoría de conjuntos: **unión**, **intersección**, **diferencia** y **producto cartesiano**.
- Los operadores relacionales especiales: **selección**, **proyección**, **concatenación** y **división**.

Para simplificar la discusión de cada una de estas operaciones, asumiremos que el orden de izquierda a derecha de los atributos dentro de una relación es significativo.

Además, asumiremos que un atributo de una relación puede tener siempre un nombre calificado, o sea, si **R** es el nombre de relación y **A** es de un atributo dentro de R, entonces éste puede ser referenciado con el nombre calificado de **R.A.**

A.- Operadores tradicionales

Sean **R1** y **R2** dos relaciones

1. **UNION (Operador Binario Básico)**

R1 UNION R2 es una nueva relación **T** cuyas tuplas pertenecen a la relación **R1** o a la **R2** o a ambas, es decir, $\mathbf{R1} \cup \mathbf{R2} = \{ \mathbf{t} \mid \mathbf{t} \in \mathbf{R1} \vee \mathbf{t} \in \mathbf{R2} \}$

Nota: **R1** y **R2** tienen que tener el mismo grado y el i-ésimo atributo ($1 \leq i \geq n$) de cada relación tiene que estar definido sobre el mismo dominio.

Externos

No Estudiante	Nombre	Edad	Año
1	Carlos	16	2
2	Luis	23	3

Becados

No Estudiante	Nombre	Edad	Año
3	Heriberto	19	2
4	Teresa	21	2
5	Pedro	22	4
6	Sandra	20	5

T = Estudiantes = Externos $\cup$ Becados

No Estudiante	Nombre	Edad	Año
1	Carlos	16	1
2	Luis	23	3
3	Heriberto	19	2
4	Teresa	21	2
5	Pedro	22	4
6	Sandra	20	5

Numero de Elementos de T = Numero de elementos de R1 + Numero de Elementos de R2 - Elementos Comunes a R1 y R2

2.- DIFERENCIA (Operador Binario Básico)

R1 MENOS R2 es una nueva relación **T** en que sus tuplas pertenecen a la relación **R1** y no a la **R2,** es decir: **$R1 - R2 = \{ t \mid t \in R1 \wedge t \notin R2 \}$**

Ejemplo:

Estudiantes Becados

No Estudiante	Nombre
3	Heriberto
4	Teresa
5	Pedro
6	Sandra

Estudiantes de 2do Año

No Estudiantes	Nombre
1	Carlos
3	Heriberto
4	Teresa

Entonces: T = Estudiantes Becados que no son de 2do Año =
Estudiantes Becados - Estudiantes de 2do Año

No Estudiante	Nombre
5	Pedro
6	Sandra

No Elementos T = No de Elementos R1 - No de Elementos Comunes

3.- INTERSECCION (Operador Binario no básico)

R1 INTERSECCION R2 es una nueva relación **T** cuyas tuplas pertenecen a la relaciones **R1** y **R2**, es decir, $\mathbf{R1} \bigcap \mathbf{R2} = \{ \mathbf{t} \mid \mathbf{t} \in \mathbf{R1} \wedge \mathbf{t} \in \mathbf{R2} \}$

Nota: **R1** y **R2** tienen que tener el mismo grado y el i-ésimo atributo ($\mathbf{1} \leq \mathbf{i} \geq \mathbf{n}$) de cada relación tiene que estar definido sobre el mismo dominio.

Ejemplo:

Estudiantes Becados

No Estudiante	Nombre
3	Heriberto
4	Teresa
5	Pedro
6	Sandra

Estudiantes de 2do Año

No Estudiante	Nombre
1	Carlos
3	Heriberto
4	Teresa

Entonces **T = Estudiantes becados de 2do Año =**
Estudiantes Becados $\bigcap$ Estudiantes 2do Año

No Estudiante	Nombre
3	Heriberto
4	Teresa

La operación de Intersección se deriva de la operación de diferencia como sigue:

$$A \bigcap B = A - (A - B)$$

de ahí que sea una operación no básica, le proponemos la comprobación de dicha igualdad.

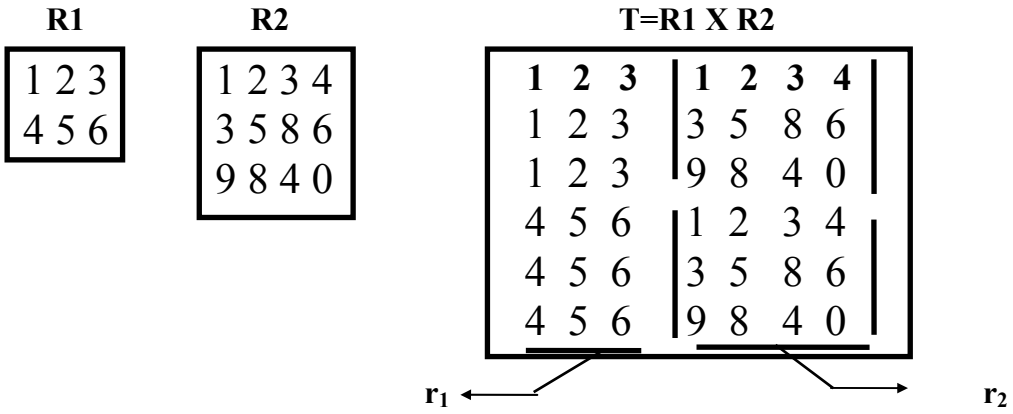
No de Elementos de T = No de Elementos Comunes

.

4.- PRODUCTO CARTESIANO (Operador Binario básico)

R1 X R2 es una nueva relación **T** en que sus tuplos **t** son la concatenación de un tuplo $r_1 \in$ **R1** y un tuplo $r_2 \in$ **R2,** es decir:

$$R1 \text{ X } R2 = \{ (r_1 , r_2) \mid r_1 \in R1 \wedge r_2 \in R2 \}$$



En este caso **R1** y **R2** no tiene que ser del mismo grado y sus respectivos atributos no tienen que estar definidos en el mismo dominio. El grado de **T** será igual a la suma de los grados de **R1** y de **R2.**

Las operaciones de **unión**, **intersección** y **producto** son asociativas, no así la **diferencia**

B.- Operadores especiales

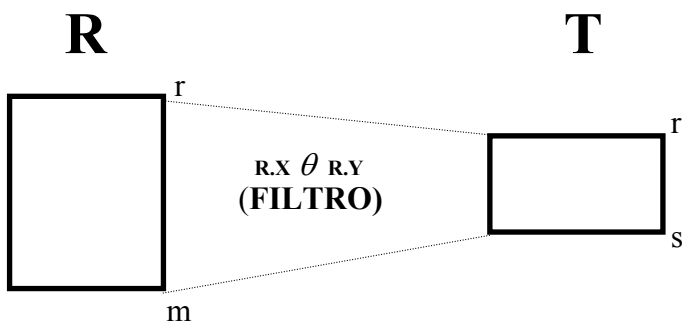
1. Selección (Operador Unario básico)

Sea θ (zita) cualquier operador de comparación **(= , <, >,** $\leq$, $\geq$, $\neq$ **).** Entonces la selección de **R** bajo el predicado **R.X** θ **R.Y,** es una nueva relación **T** formada por las tuplas **t** de **R** tales que el predicado **t.x** θ **t.y** toma valor verdadero. Los atributos X y Y deben estar definidos en el mismo dominio y la operación θ debe tener sentido en ese dominio. En lugar de X ó de Y se puede especificar una constante, o sea **R.X** θ constante.

Denotándose la operación por: $\mathbf{R}_{\mathbf{R.X}\ \theta\ \mathbf{R.Y}}$. Entonces se tendrá que:

$$\mathbf{R}_{\mathbf{R.X}\ \theta\ \mathbf{R.Y}} = \{\, \mathbf{t} \mid \mathbf{t} \in \mathbf{R} \wedge \mathbf{t.X}\ \theta\ \mathbf{t.Y} \text{ sea cierta} \,\}$$

La operación de selección constituye un corte "horizontal" de la relación **R**, o sea,



T tendrá igual dominio y grado **(r)** que **R** pero en general diferente **Extensión (m ≥ s).**

Ejemplo:

Dada la relación Becados (No Estudiante, Nombre, Edad) obtener la relación **T** de Estudiantes becados con edad mayor de 20 años.

$$\mathbf{T = Becados}_{\mathbf{Becados.Edad\ >\ 20}}$$

A su vez los predicados θ pueden ser combinadas con uso de los operadores lógicos **AND, OR y NOT,** así se **C1** y **C2** son predicados

Entonces:

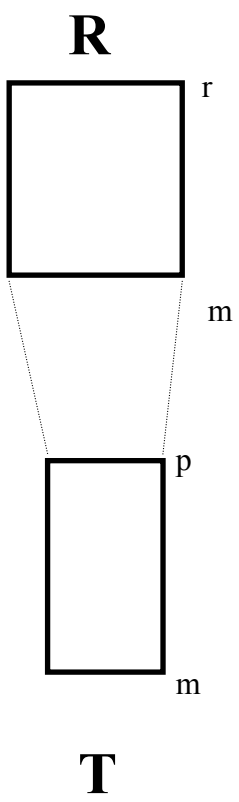
- $\mathbf{R_{C1\ AND\ C2}}$ equivalente a $\mathbf{R_{C1} \cap R_{C2}}$

- $\mathbf{R_{C1\ OR\ C2}}$ equivalente a $\mathbf{R_{C1} \cup R_{C2}}$

- $\mathbf{R_{NOT\ C}}$ quivalente a $\mathbf{R - R_C}$

2. Proyección (Operador unario básico)

La proyección de una relación **R** sobre los atributos **X, Y,..., Z** es una nueva relación **T** cuyos tuplos (x, y, …, z) aparecen en la relación R con el valor x en X, y en Y, …, z en Z, es decir, la proyección de una relación R constituye un corte vertical de la misma, o sea:



El grado de **T** (**p**) será por lo general menor o igual al de **R** (**r**).

Esta operación se denotará por **R** [X, Y, ...,Z]

Ejemplo:

Dada la relación **Externos (NoEstudiante, Nombre, Edad, año),** la nueva relación en la que solo aparece los atributos **NoEstudiante** y **año** se obtendría:

Externos [NoEstudiante,Año]

Puesto que consideramos significativo el orden de los atributos en la relación, la proyección nos brinda una forma de reordenar los atributos de esta.

Ningún atributo puede ser especificado más de una vez en la lista de atributos de la proyección. Omitir dicha lista es equivalente a especificar todos los atributos en su correspondiente orden de izquierda a derecha, o sea, dicha proyección sería idéntica a la relación dada, eliminándose las tuplas

repetidas.

Es común la existencia de situaciones en las que sean necesario hacer uso tanto la selección como la proyección, por ejemplo, asi por ejemplo que necesitemos obtener el nombre de los estudiantes becados de 2do año.

$$(\mathbf{Becados}_{\mathbf{Becados.Año = 2}})_{\mathbf{[Nombre]}}$$

3. Concatenación (Operador binario no básico)

Sea θ cualquier operador de comparación (al igual que en la selección). Sean **R1** y **R2** dos relaciones. La concatenación, mediante el predicado **R1.X** θ **R2.Y** , de la relación **R1** sobre su atributo **X** , con la relación **R2** sobre su atributo **Y**, es una nueva relación **T** cuyos tuplos **t** son la concatenación de un tuplo r_1 de R1 con un tuplo r_2 de R2 para los cuales el predicado $\mathbf{r_1.x}\ \theta\ \mathbf{r_2.y}$ toma valor verdadero. $\mathbf{r_1.x}$ y $\mathbf{r_2.y}$ deben estar definidos sobre el mismo dominio y la operación θ debe tener sentido en él.

Denotándose esta operación por : $\mathbf{R1} \underset{\mathbf{R1.X}\ \theta\ \mathbf{R2.Y}}{\infty} \mathbf{R2}$

Entonces tendremos que:

$$\mathbf{R1} \underset{\mathbf{R1.X}\ \theta\ \mathbf{R2.Y}}{\infty} \mathbf{R2} = \{ (r_1 , r_2) \mid r_1 \in R1 , r_2 \in R2 \wedge r_{1.X}\ \theta\ r_{2.Y} \text{ sea cierta} \}$$

La operación de concatenación es asociativa.

Si θ es el operador = , la operación se denomina "**equijoin**". Entonces, el resultado de un equijoin tiene que incluir a dos atributos idénticos. Si uno de ellos se elimina en la relación resultante, entonces el resultado se denomina **concatenación natural** (**join natural**) o simplemente **Join.**

El **JOIN** es equivalente a realizar una selección sobre el producto cartesiano de las relaciones, veamos:

Sean **R1** y **R2** las relaciones siguientes:

R1

A	B
a1	b1
a2	b2
a3	b2

R2

B	C
b1	c1
b2	c2
b3	c3

El **JOIN** de las relaciones R1 y R2 sobre el atributo **B** común de ambas seria la relación:

R1 JOIN R2

A	B	C
a1	b1	c1
a2	b2	c2
a3	b2	c2

lo que equivale a :

R1 X R2

a1	b1	b1	c1
a1	b1	b2	c2
a1	b1	b3	c3
a2	b2	b1	c1
a2	b2	b2	c2
a2	b2	b3	c3
a3	b2	b1	c1
a3	b2	b2	c2
a3	b2	b3	c3

T = R1 X R2 $_{\text{R1. B = R2.B}}$

a1	b1	b1	c1
a2	b2	b2	c2
a3	b2	b2	c2

a1	b1	c1
a2	b2	c2
a3	b2	c2

T $_{\text{A, B, C}}$

Así tendremos que en nuestro modelo Suministradores - Productos:

SUMINISRADOR JOIN SP sobre los atributos **SNUM** en **SUMINISTRADOR** y **S** en **SP**, que están definidos sobre el mismo dominio, que equivaldría a:

((SUMINISTRADOR X SP)$_{SUM.SNUM=SP.S}$) [SUM.SNUM, SUM.SNOM, SUM.TIPO, SUM.MUN, SP.P, SP.CANT]

Si las relaciones sobre las que se aplica la operación **JOIN** no tienen ningún atributo con igual nombre, entonces se supone que la concatenación se realice según el último atributo de la primera relación y el primero de la segunda. En este caso puede ser necesario plantear las correspondientes proyecciones antes de realizar el Join con el objetivo de reordenar los atributos convenientemente, así por ejemplo en el caso anterior de **SUMINISTRADOR JOIN SP** debió plantear:

SUMINISTRADOR $_{[SNOM, TIPO, MUN, SNUM]}$ JOIN SP

De las tres operaciones relacionales especiales vistas, la selección es la de mayor prioridad, seguida de la proyección y por último la unión. Como es frecuente, es posible usar los paréntesis para combinar dichas operaciones alternando sus prioridades.

Ejemplos:

Usando el modelo Suministrador - Producto

Obtengamos el nombre de los suministradores que suministran el producto P2

((SUMINISTRADORES$_{[SNOM,SNUM]}$ JOIN SP)$_{P = 'P2'}$)$_{[SNOM]}$

Se proyecta SUMINISTRADORES con el objetivo de reordenar sus atributos con vista al JOIN y a la vez para escoger los atributos necesarios. Luego se concatena la relación obtenida de la proyección con la relación SP (destaquemos que como los atributos que se toman para realizar la concatenación no tienen igual nombre, en la relación obtenida de la proyección de SUMINISTRADOR es nombrado por SNUM y en la relación SP por S, al no especificarse el predicado del JOIN se tomará el último atributo de la proyección es decir SNUM y el primer atributo de SP, o sea , S), obteniéndose una nueva relación con los atributos: SNOM, SNUM, P, CANT. De esta se realiza la selección para escoger los tuplos que hagan verdadero el predicado P = ´P2´ y por último la relación obtenida se proyecta sobre SNOM, puesto que este es el atributo que nos interesa recuperar.

Una forma más eficiente de realizarlo sería:

(SUMINISTRADOR $_{[SNOM, SNUM]}$ JOIN SP $_{P = 'P2'}$)$_{[SNOM]}$

¿ Por qué ?

2.- Obtengamos los nombres de los suministradores que suministran al menos un producto de precio igual a 0.10.

$$(((\text{PRODUCTO}_{\text{PRECIO} = 0.10})_{[\text{PNUM}]} \text{ JOIN SP}_{[\text{S, P}]})_{[\text{S}]} \text{ JOIN SUMINISTRADOR})_{[\text{SNOM}]}$$

Se realiza una selección de la relación PRODUCTO de los productos cuyo precio es de 10 centavos. En esta selección mediante una proyección se toma sólo el atributo PNUM (número del producto). La relación obtenida de la proyección se concatena con la relación obtenida mediante el reordenamiento de la relación SP mediante una proyección en la cual se toma de SP sólo los atributos necesarios: S y P. La relación obtenida como resultado de la concatenación se le realiza una proyección con vista a obtener otra relación que contenga sólo el atributo S. Esta última relación obtenida se concatena con la relación SUMISTRADORES obteniéndose una nueva relación de la cual se obtiene el nombre de los suministradores mediante una proyección

4. **División (Operador binario no básico)**

La operación de división divide una relación dividiendo **R1** de grado **m + n** entre una relación divisor **R2** de grado **n**, produciendo una relación cociente **T** de grado **m**. El atributo **m + i** de **R1** y el atributo **i-ésimo** de **R2** (i= 1,2,...n) deben estar definidos sobre el mismo dominio, es decir si la relación **R1** esta definida en el dominio **X y la** relación **R2** esta definida en el dominio **Y,** será necesario para que la relación cociente **T = R1 / R2** exista, que $\mathbf{Y} \subset \mathbf{X}$, encontrándose **T** definida en el dominio **X - Y**

La relación cociente **T** estará formada por todos los tuplos **t** tales que para todos los tuplos $\mathbf{r_2}$ de la relación **R2,** el tuplo $\mathbf{(t , r_2)}$ pertenece a la relación **R1,** es decir:

$$\mathbf{T \times R2 = R1}$$

Luego:

$$\mathbf{R1 / R2 = \{ t \mid \forall\ r_2 \in R2 \rightarrow (t , r_2) \in R1 \}}$$

Ejemplos:

1.-

Sean las relaciones:

Estudiantes (NumEst, NomEst)

Asignaturas (NumAsig, NomAsig, Creditos)

Notas (NumEst, NumAsig, Nota)

Obtener el nombre de todos los estudiantes que tengan examinadas todas las asignaturas.

(((NOTAS $_{[\ NUMEST,\ NUMASIG]}$) / (ASIGNATURA $_{[\ NUMASIG]}$)) JOIN ESTUDIANTES) $_{[NOMEST]}$

Si por ejemplo se tiene que:

Estudiantes

NUMEST	NOMEST
1	Salvador
2	Antonio
3	Susana
4	Luisa

Asignaturas

NUMASIG	NOMASIG	Créditos
1	Estructura de Datos	3
2	Sistema Operativo	3
3	Programación Avanzada	3
4	Base de Datos	3

NOTAS

NUMEST	NUMASIG	NOTA
1	1	3
1	2	4
1	3	3
1	4	4
2	1	5
2	2	2
2	4	4
3	1	5
3	3	2
4	1	2
4	4	3

Entonces el resultado de:

$$C = (\text{Notas}_{[\text{NUMEST, NUMASIG}]}) / (\text{Asignatura}_{[\text{NUMASIG}]}) = [\ 1\ ]$$

Entonces:

C JOIN ESTUDIANTES = [1, Salvador]

La que al proyectarse sobre NOMEST se obtendría **Salvador**

El cociente tiene la misma prioridad que la operación de concatenación

Usando el modelo SUMISTRADOR - PRODUCTO obtenga:

2.- El nombre de los suministradores que suministran todos los productos

$(SP_{[S,P]} / PRODUCTOS_{[PNUM]}$ JOIN SUMINISTRADORES$)_{[SNOM]}$

Al proyectar PRODUCTOS sobre PNUM se obtienen todos los PNUM existentes, por tanto, cuando se hace la división se obtienen los números de suministradores tales que en SP cada uno de ellos tiene asociados todos los valores de la relación divisora (la proyección de PRODUCTOS). Sólo resta concatenar con SUMINISTRADORES para obtener los correspondientes nombres.

3. - El número de los suministradores que sirven al menos aquellas piezas suministradas por S2

$$\mathbf{SP}_{[S,P]} / (\mathbf{SP}_{S='S2'})_{[P]}$$

La relación cociente se forma tomando como divisor la relación obtenida por los números de productos servidos por S2. Para obtenerla, utilizamos la selección con el predicado **S = " S2 "** y la relación obtenida la proyectamos sobre **P**. El dividendo será la relación que contendrá sólo los atributos **S** y **P** ya que dividiremos entre **P** y sólo nos interesa obtener **S**.

Al efectuar el cociente, obtendremos una relación que contendrá los números de suministrador **S** que en la relación **SP** tienen asociadas cada uno todos los números de productos del divisor (al menos).

Como vimos estas operaciones permiten realizar la recuperación de datos, aunque este no es su único objetivo pues en general el álgebra relacional posibilita también la escritura de expresiones que pueden ser utilizadas con diversos fines y no sólo para la recuperación, por ejemplo: definir derechos de acceso, restricciones de integridad, datos virtuales, etc

Por último diremos que las expresiones algebraicas pueden ser manipuladas para encontrar otras equivalentes, pero más eficientes para realizar las recuperaciones, por lo que el álgebra puede ser utilizada como base para el proceso de optimización de las solicitudes de información a la BD. El proceso de optimización debe realizarlo el SGBD y no ser objeto de estudio en nuestro curso. Por ejemplo:

$$(\mathbf{SUMINISTRADOR}_{[SNUM]}\ \mathbf{JOIN}\ \mathbf{SP})_{P='P1'}$$

es equivalente a:

$$\mathbf{SUM}_{[SNUM]}\ \mathbf{JOIN}\ \mathbf{SP}_{P='P1'}$$

Sin embargo la segunda expresión es más eficiente. ¿ Por qué ?

Otra alternativa para definir la parte manipulativa del modelo relacional es el Cálculo Relacional.

El Cálculo Relacional y sus Expresiones

El cálculo relacional representa una alternativa al álgebra para la parte manipulativa del modelo relacional. La diferencia entre ellos es la siguiente:

El álgebra ofrece una colección de operaciones explícitas (join, unión, proyección, selección, etc) que pueden ser usadas para construir una relación deseada a partir de las relaciones existentes en la Base de Datos.

El cálculo sólo ofrece una notación para formular la definición de la relación deseada a partir de las relaciones existentes en la Base de Datos.

Por ejemplo, consideremos la solicitud siguiente:

"Obtener el número y municipio de los suministradores que sirven la Pieza P2"

Una formulación hecha con uso del Álgebra Relacional sería la siguiente:

- **Proyectar SUMINISTRADORES** en **MUN** y **SNUM**
- Hacer un **JOIN** de la **proyección** obtenida con la relación **SP**
- Realizar en la relación obtenida con el JOIN anterior una **selección** de los tuplos con **P = 'P2'**
- **Proyectar** la relación obtenida de la selección anterior sobre **SNUM** y **MUN**

Una formulación hecha con el cálculo sería algo así:

- Obtener **SNUM** y **MUN** para los suministradores tales que exista un suministro **SP** con el mismo valor de **SNUM** y con el valor de **P = 'P2'**

Aquí se estable las características definitorias de la relación deseada y se le deja al sistema la decisión de que operaciones deben ejecutar para construirla.

Podemos decir entonces que la formulación del cálculo es "**descriptiva**" mientras que la del álgebra es "**deductiva**"; el cálculo simplemente establece cuál es el problema, mientras que el álgebra da un procedimiento para resolver el problema. O sea el álgebra es procedural y el cálculo no es procedural.

Sin embargo, debe recalcarse que esta distinción entre uno y otro es superficial. Es un hecho que el álgebra y el cálculo son equivalentes uno al otro. Para cada expresión del álgebra existe una expresión equivalente en el cálculo. Igualmente para cada expresión del cálculo existe una expresión equivalente del álgebra. Hay una correspondencia uno a uno entre las dos. Los diferentes formalismos simplemente representan diferentes estilos de expresión (el cálculo es más cercano al lenguaje natural, el álgebra es quizás más parecida a un lenguaje de programación convencional), pero estas diferencias son más aparentes que reales, en particular, ningún enfoque es realmente menos procedural que otro.

El cálculo relacional se basa en una rama de la lógica matemática llamada cálculo de predicados.

El elemento fundamental del Cálculo Relacional es la noción de "**variable de tupla**" (también conocida como "**variable de rango**"). Una variable de tupla es una variable que toma valores sobre una relación, o sea, una variable cuyos únicos valores permitidos son tuplas de una relación determinada. En otras palabras, si la variable de tupla T toma valores sobre la relación R, entonces en cualquier momento del tiempo, T representa a algún tuplo t de la relación R.

Una variable de tupla se define con la siguiente instrucción: RANGE OF T IS R (siendo T la variable de tupla y R una relación). Así por ejemplo las variables **SX, SY, PZ, PX, SPY** serán definidas:

RANGE OF **SX** IS SUMINISTRADOR
RANGE OF **SY** IS SUMINISTRADOR
RANGE OF **PZ** IS PRODUCTO
RANGE OF **PX** IS PRODUCTO

RANGE OF **SPY** IS SP

Expresiones del cálculo relacional

Una instrucción de recuperación del cálculo relacional tendrá la siguiente forma general:

lista objeto | predicado

donde:

lista objeto: Especifica qué atributos y de qué relaciones se desean recuperar. Está formada por nombres calificados de atributos separados por comas (la calificación se hace sobre variables de tuplas).

predicado: Especifica las condiciones que deben verificar los tuplos seleccionados y es opcional.

El predicado puede estar formado por relaciones, con uso de un operador de comparación (>, <, =, etc), entre dos nombres de atributos o entre un nombre de atributo y una constante, siendo también predicado:

- **NOT** predicado
- predicado **AND** predicado
- predicado **OR** predicado
- ∃ Var_Tupla (predicado existencial)
- ∀ Var_Tupla (predicado universal)

La "ocurrencia de una variable de tupla" será la aparición del nombre de la variable dentro del predicado en cuestión. Luego una variable de tupla **T** ocurre dentro de un predicado bien en el contexto de una referencia a un atributo, como por ejemplo **T.A**., donde **A** es un atributo de la relación sobre la que **T** toma valor, o como la variable que sigue inmediatamente a uno de los cuantificadores ∃ (existencial) o ∀ (universal), es decir, ∃ **T ó** ∀ **T.**

Así si se tiene que las variables tuplas:

RANGE OF **SPX** IS SP

RANGE OF **SX** IS SUMINISTRADOR

RANGE OF **PX** IS PRODUCTO

Tendrán las ocurrencias:

a) ∃ **SPX** (**SPX**.S=**SX**.SNUM **SPX**.P='P2')

Cuya interpretación es: "Existe un tuplo de **SP** con valor de **S** igual al valor de **SX**.SNUM "(cualquiera que sea éste)" y el valor del PNUM es igual a P2" .

b) ∀ **PX** (**PX**.PRECIO='1.00')

Cuya interpretación es: "Para todos los tuplos de la relación PRODUCTO, su precio es de $1.00".

Ejemplos

Dada las variables tuplas**:**

RANGE SX IS SUMINISTRADOR

RANGE SY IS SUMINISTRADOR

RANGE SZ IS SUMINISTRADOR

RANGE PX IS PRODUCTO

RANGE PY IS PRODUCTO

RANGE PZ IS PRODUCTO

RANGE SPX IS SP

RANGE SPY IS SP

RANGE SPZ IS SP

1. Obtener el número de los suministradores del municipio "10 de Oct" con tipo > 20

SX.SNUM | **SX**.MUN = "10 de Octubre" AND **SX**.TIPO > 20

2. Obtener todos los pares ordenados de números de suministradores tales que en cada par los dos suministradores estén en el mismo municipio

SX.SNUM, **SY**.SNUM | **SX**.MUN = **SY**.MUN AND **SX**.SNUM < **SY**.SNUM

3. Obtener los nombres de los suministradores que sirven el producto P2

SX.SNOM | ∃ **SPX** (**SPX**.S = **SX**.SNUM AND **SPX**.P = 'P2')

4. Obtener los nombres de los suministradores que sirven, al menos, un producto de PESO = 30

SX.SNOM | ∃ **SPX** (**SX**.SNUM=**SPX**.S AND ∃ **PX** (**SPX**.P=**PX**.PNUM AND **PX**.PESO=30))

5. Obtener los nombres de los suministradores que sirven al menos un producto suministrado también por S2.

SX.SNOM | ∃ **SPX** (SPX.S=**SX**.SNUM AND ∃ **SPY** (**SPX**.P=**SPY**.P AND **SPY**.S='S2'))

6. Obtener los nombres de los suministradores que sirven todos los productos

SX.SNOM | ∀ **PX** (∃ **SPX** (**SPX**.S=**SX**.SNUM AND **SPX**.P = **PX**.PNUM))

7. Obtener los nombres de los suministradores que no sirven el producto P2

SX.SNOM | NOT ∃ **SPX** (**SPX**.S=**SX**.SNUM AND **SPX**.P = 'P2')

Lenguaje SQL

Entre los lenguajes relacionales comerciales se encuentra el **SQL** (STRUCTURE QUERY LANGUAGE). Este al igual que todo lenguaje estará compuesto por:

- **L**enguaje de **D**efinición de **D**atos (**DDL**)
- **L**enguaje de **M**anipulación de **D**atos (**DML**)
- **L**enguaje de **C**ontrol de **D**atos (**DCL**)

Por lo que nos permitirá: consultar actualizar y administrar la Base de Datos.

Lenguaje Manipulación de Datos (DDL)

CREATE TABLE | INDEX (permite crear una nueva tabla en la Base de Datos o Indice en una Tabla)

Ejemplos:

1. **CREATE TABLE** [Esta tabla] ([nombre]**TEXT**, [Apellidos]**TEXT**)

Crea una nueva tabla llamada **Esta Tabla** con dos campos de texto.

2. **CREATE TABLE** MiTabla ([Nombre]**TEXT**, [Apellidos]**Text**, [Fecha de Nac] **DATETIME**) **CONSTRAINT** IndiceMitabla **UNIQUE** ([Nombre], [Apellidos])

Crea una nueva tabla llamada **MiTabla** con dos campos texto, un campo fecha y un índice único formado por dos campos.

3. **CREATE TABLE** NuevaTabla ([nombre]**TEXT**, [Apellidos]**TEXT**, [NSUM]**INTEGER CONSTRAINT** IndiceMiCampo **PRIMARY**, [Fecha de Nacimiento]**DATETIME**

   Crea una nueva tabla con dos campos de texto, uno de tipo entero y uno de tipo fecha. El campo de tipo entero es llave principal.

4. **CREATE INDEX** NuevoIndice **ON** Empleados ([Telefono domocilio] Extension)

   Crea en la Tabla Empleados un indice consistente en los campos Telefono domicilio y Extension

5. **CREATE UNIQUE INDEX** MiIndice **ON** Empleados (NSS) **WITH DISALLOW NULL**

   Crea un indice en la tabla Empleados utilizando el campo NSS (Numero de la Seguridad Social). No puede haber dos registros con el mismo valor en el campo NSS, y no se admite en este el valor Nulo.

ALTER TABLE (permite modificar el diseño de una tabla.)

Ejemplos:

1. **ALTER TABLE** Empleados **ADD COLUMN** Salario **CURRENCY**

Agrega el campo Salario de tipo Moneda a la tabla Empleados

2. **ALTER TABLE** Empleados **DROP COLUMN** Salario

Elimina el campo Salario de la tabla Empleados

3. **ALTER TABLE** Pedidos **ADD CONSTRAINT** RelacionPedidos **FOREIGNKEY** ([Id de Empleado]) **REFERENCES** Empleados ([Id de Empleado])

Agrega una llave externa a la tabla Pedidos. La llave externa se basa en el campo ID de Empleado y se refiere al campo ID de Empleado de la tabla Empleados. En caso que el campo de la tabla REFERENCES sea llave principal de esta no será necesario señalar su nombre.

4. **ALTER TABLE** Pedidos **DROP CONSTRAINT** RelacionPedidos
   Elimina la llave externa de la Tabla Pedidos

DROP TABLE | INDEX (permite eliminar una tabla de una Base de Datos o un Indice de una tabla)

Ejemplos:

1. **DROP INDEX** MiIndice **ON** Empleados

Elimina MiIndice de la tabla Empleados

2. **DROP TABLE** [Esta Tabla]
   Elimina "Esta Tabla" en la Base de Datos

Lenguaje de Manipulación de Datos (DML)

SELECT (permite realizar consultas sobre tablas existentes)

SELECT generalmente es la primera palabra de una instrucción SQL. La mayoría de las instrucciones **SQL** son del tipo **SELECT.** La sintaxis mínima de una instrucción **SELECT** es:

SELECT [predicados] Lista*Atributos* FROM *Listatablas* [IN nombre*baseDatosExterna*]

es posible el uso de asterisco para seleccionar todos los atributos de una tabla, así por Ejemplo:

SELECT Empleados.* **FROM** Empleados

Cuando es usada una instrucción **SELECT** se incluye todas las tuplas de la relación. Para realizar una selección sobre la relación es usada la cláusula **WHERE**.

Ejemplos:

1. **SELECT** [Apellidos], [Nombre] **FROM** Empleados **WHERE** [Apellidos] = " King"

Realiza una proyección sobre los atributos Apellidos y Nombre de la relación Empleados para la selección donde el apellido es King

2. **SELECT** [Apellidos], [Nombre] **FROM** Empleados **WHERE** [Apellidos] **LIKE** "S*"

Realiza una proyección sobre los atributos Apellidos y Nombre de la relación Empleados para la selección donde el apellido comience con la letra S

3. **SELECT** [Apellidos], Salario **FROM** Empleados **WHERE** Salario **BETWEEN** 20000 AND 30000

Realiza una Proyección sobre los atributos Apellidos y Salario de la relación Empleados para la selección donde el atributo salario este comprendido entre el $200.00 y $ 300.00 inclusive.

4. **SELECT** [ÌD de Pedido], [Fecha de Pedido] **FROM** Pedidos **WHERE** [Fecha de pedido] **BETWEEN** #1-1-94# AND #30-6-94#

Realiza la proyección de los atributos "ID de Pedido" y "Fecha de Pedido" de la relación Pedidos para la selección en que el atributo "Fecha de Pedido" sea del primer semestre.

5. **SELECT** [Apellidos], [Nombre], Cuidad **FORM** Empleados **WHERE** Cuidad **IN** ("Sevilla", "Los Angeles", "Barcelona")

Realiza la proyección sobre los atributos Apellidos, Nombre y Cuidad de la relación Empleados para la selección en que el atributo cuidad tome el valor de "Sevilla" ó "Los Angeles" ó de "Barcelona".

6. **SELECT** [Apellidos], Salario*****1.1 **AS** Propuesto **FROM** Empleados

Realiza la proyección de los atributos Apellidos y Salario de la relación Empleados mostrado el valor tomado por el atributo Salario multiplicado por 1.1 y con el nombre de Propuesto (no cambia los valores originales del atributo Salario)

Algunos predicados nos permiten realizar determinadas selecciones así con los predicados:

- **DISTINCT,** nos permitirá omitir tuplas con valores duplicados en los campos seleccionados. La salida de una consulta que use **DISTINCT** no podrá actualizarse

Ejemplo: SELECT **DISTINCT** Apellidos **FROM** Empleados

- **TOP,** nos permitirá obtener el número de tuplas especificado, los que serán los primeros o últimos según la cláusula **ORDER BY**

Ejemplo:

SELECT TOP 25 Nombre, Apellidos **FROM** Estudiantes **WHERE** [Año de graduación] = 1994
ORDER BY [Puntuación Media] **DESC**

En este caso se obtendrá los 25 mejores estudiantes de la promoción 1994

- **DISTINCTROW,** cuando se desee omitir datos en base a tuplas completas duplicadas de la relación y no solamente en atributos duplicados.

Ejemplo:

Sea una consulta que una las relaciones Clientes y Pedidos según el atributo "ID del Cliente". La tabla Clientes no tiene valores del atributo " ID del Cliente" duplicados, sin embargo la tabla Pedidos si puesto que un Cliente puede tener varios pedidos. La siguiente expresión SQL muestra como obtener una relación de las compañías que tengan al menos un pedido, haciendo uso de la cláusula DISTINCTROW.

SELECT DISTINCTROW [Nombre de la compañía] **FROM** Clientes, Pedidos, Clientes **INNER JOIN** Pedidos **ON** Clientes.[ID de Cliente] = Pedidos.[ID de Cliente]

El ordenamiento de los resultados de la consulta es posible realizarlo usando la cláusula **ORDER BY** la cual es opcional salvo en el caso que sea utilizado el predicado **TOP**

Ejemplos:

1. **SELECT** Apellidos, Salario **FROM** Empleados **ORDER BY** Salario **DESC**, [Apellidos]
   Ordena los registros de la proyección obtenida de la relación Empleados por el atributo Salario de forma Descendente y para las tuplas con igual valor en el atributo salario las ordenará por el atributo apellido de forma ascendente
2. **SELECT** [Id de categoría] , [Nombre de Producto], [Precio Unitario] **FROM** Productos **ORDER BY** [Id de categoría], [Nombre de Producto]
   Primero ordena por "ID de categoría" y después por "Nombre de Producto"

El SQL contiene diversas funciones de agrupamiento como:

- **COUNT** (cuenta el número de registros seleccionados)
- **AVG** (calcula la media aritmética del conjunto de valores tomados por el atributo especificado)
- **SUM** (suma el conjunto de valores tomados por el atributo especificado)
- **MAX** (calcula el valor máximo del conjunto de valores tomado por el atributo especificado)
- **MIN** (calcula el valor mínimo del conjunto de valores tomado por el atributo especificado)

Ejemplos:

1. **SELECT COUNT** ([País destinatario]) **AS** [Pedidos España] **FROM** Pedidos

   Calcula el número de pedidos enviados a España.

2. **SELECT COUNT** (*) **AS** [Total Empleados], **AVG** (Salario) **AS** [Salario Medio], **MAX** (Salario) **AS** [Salario Máximo] **FROM** Empleados

   Muestra el número de empleados y los salarios promedio y máximo

3. **SELECT AVG** ([gastos de envío]) **AS** [Gastos promedios] **FROM** Pedidos **WHERE** [gastos de envío] > 100

   Calcula el gasto promedio de los envío con gasto de más de $ 100.00

4. **SELECT SUM** ([Precio Unitario]*[Cantidad]) AS [Ventas totales a España] **FROM** Pedidos **INNER JOIN** [Detalles de Pedidos] **ON** Pedidos.[ID de Pedidos] = [Detalles de Pedidos].[ID de Pedidos] **WHERE** ([País destinatario] = "España")

Tema 3

Diseño de Base de Datos Relacionales

Contenido:

- **Normalización**
- **Primera Forma Normal (1FN)**
- **Segunda Forma Normal (2FN)**
- **Tercera Forma Normal (3FN)**
- **Forma Normal de Boyce-Codd (FNBC)**
- **Obtención del Modelo Lógico Global a partir del DER.**
- **Metodología para el diseño de bases de datos.**

Normalización

La teoría de la normalización se ha desarrollado para obtener estructuras de datos eficientes que eviten las anomalías de actualización. El concepto de normalización fue introducido por E.D. Codd para aplicarse a sistemas relaciónales. Sin embargo, tiene aplicaciones más amplias.

La normalización es la expresión formal del modo de realizar un buen diseño. Provee los medios necesarios para describir la estructura lógica de los datos en un sistema de información.

Ventajas de la Normalización:

- Evita anomalías en la actualización.

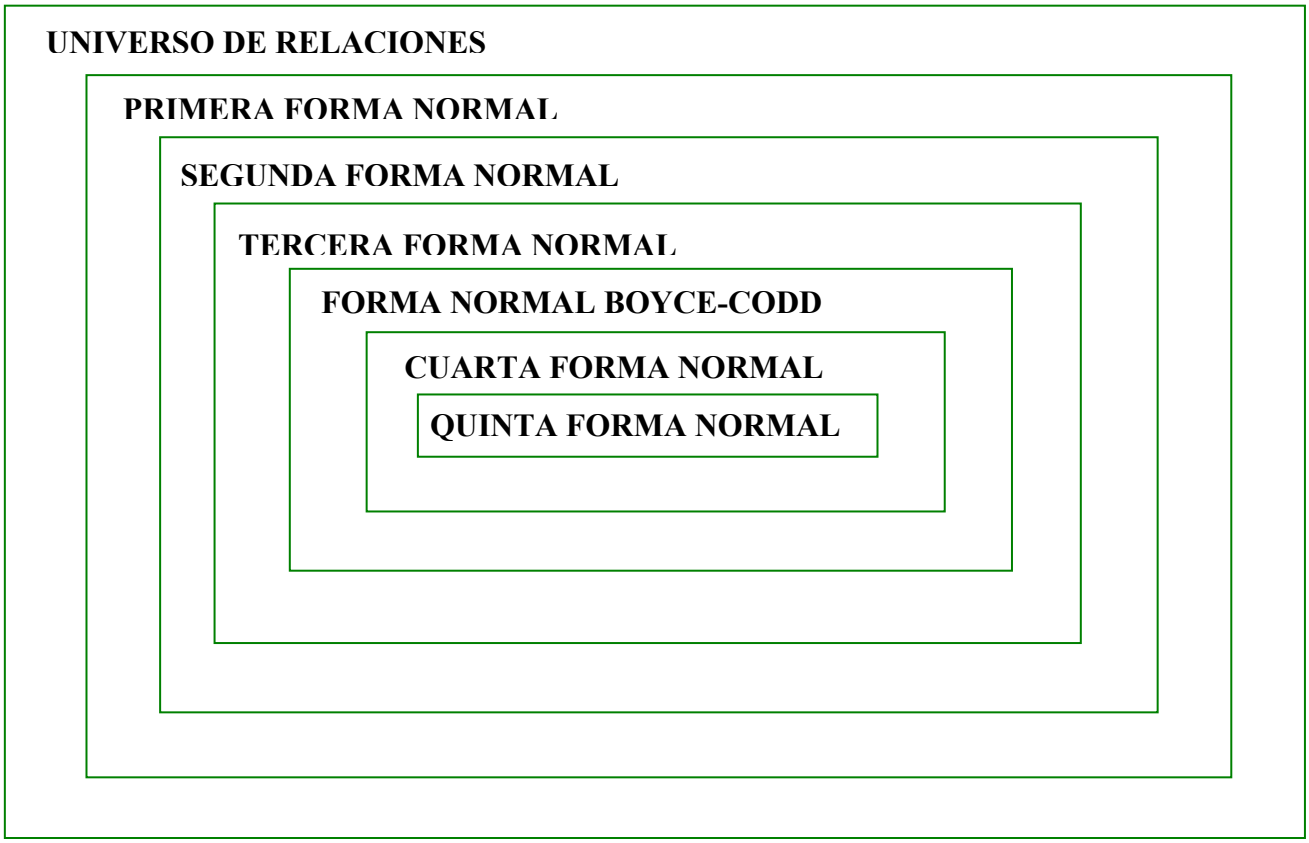
- Mejora la independencia de los datos, permitiendo realizar extensiones de la BD, afectando muy poco, o nada, a los programas de aplicación existentes que accedan a la base de datos.

La normalización involucra varias fases que se realizan en orden. La realización de la 2da fase supone que se ha concluido la 1ra y así sucesivamente. Tras completar cada fase se dice que la relación está en:

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

Existen, además **4FN** y **5FN**

Las relaciones en **1FN** son un subconjunto del universo de todas las relaciones posibles. Las relaciones en **2FN** son un subconjunto de las que están en **1FN** y así sucesivamente, como se muestra en la siguiente figura.


UNIVERSO DE RELACIONES
PRIMERA FORMA NORMAL
SEGUNDA FORMA NORMAL
TERCERA FORMA NORMAL
FORMA NORMAL BOYCE-CODD
CUARTA FORMA NORMAL
QUINTA FORMA NORMAL

Primera Forma Normal:

Una relación se encuentra en 1FN si:
• Sus atributos son atómicos • No contiene grupos repetitivos

Desarrollaremos un ejemplo para ilustrar las fases de normalización desde la **1FN** hasta la **3FN** , el mismo consiste en el diseño de una Base de Datos para la automatización del control de los pedidos de productos:

FECHA: 25/4/86

PEDIDO NO. : **123456**
PROVEEDOR NO. : **75621**
NOMBRE PROVEEDOR: **J. PÉREZ**
DIRECCION **CERRO**

DESEAMOS QUE NOS ENVÍEN:

NÚMERO PRODUCTO	DESCRIPCIÓ	PRECIO UNITARIO	CANTIDAD	TOTAL
969715	TELEVISOR	600	1	600
439124	ANTENA	20	10	200
439126	ENCHUFE	10	10	100

IMPORTE TOTAL: **900**

El análisis del modelo del pedido nos muestra que los siguientes datos son de interés:

NUPED (Número del Pedido, el cual es único y será usado como llave)
FECHA (Fecha del Pedido)

NUMPROV (Número del Proveedor)
NOMPROV (Nombre del Proveedor)
DIREC (Dirección del Proveedor)
NUMPROD (Número del Producto)
DESC (Descripción del Producto)
PRUN (Precio Unitario del Producto)
CANT (Cantidad solicitada del Producto)

Los atributos **PRPROD** (Precio del Producto) y **PRPED** (Precio del Pedido) pueden calcularse mediante otros, por lo cual no los tendremos en cuenta.

En forma de relación se escribiría

PEDIDO (NUMPED, FECHA, NUMPROV, NOMPROV, DIREC, NUMPROD, DESC, PRUN, CANT)

Esta relación contiene 4 grupos repetitivos: **NUMPROD**, **DESC**, **PRUN**, **CANT**, ya que puede estar contenidos en más de una línea del pedido.

Para llevar la relación a **1FN** es necesario eliminar los grupos repetitivos, para lo cual se creará:

1. Una relación para los atributos que sean únicos.
2. Una relación para los grupos repetitivos.

Así se tendrá las relaciones:

1) **PEDIDO** (**NUMPED**, FECHA, NUMPROV, NOMPROV, DIREC).

2) **PED-PROD** (**NUMPED, NUMPROD,** DESC, PRUN, CANT).

Ambas relaciones tienen como llave, o parte de esta, a **NUMPED**. Pero en la relación **PED-PROD** es necesario tomar la llave compuesta para poder identificar a los productos individuales.

Esta **1FN** presenta los siguientes problemas de actualización:

Inserción: La información sobre un nuevo producto no se puede insertar si no hay un pedido que lo incluya.

Supresión: Al eliminar una línea un pedido que sea el único que pida un determinado producto implicará perder también la información de ese producto.

Modificación: Para cambiar la información de un Producto será necesario también hacerlo en todos los tuplos de la relación PED-PRO en que aparezca el Producto.

Segunda Forma Normal

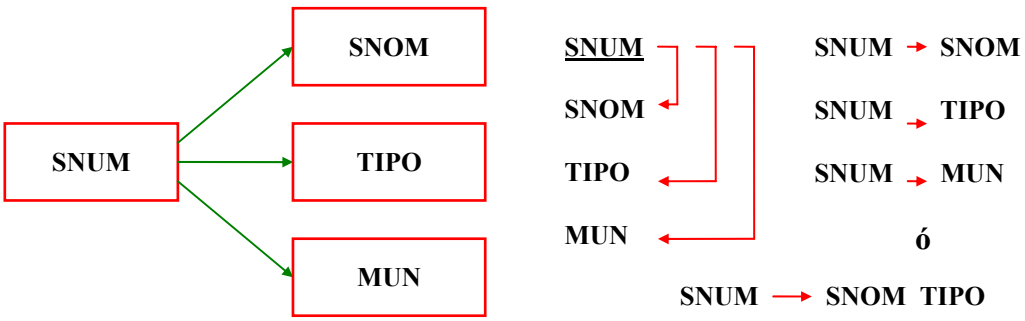
Para poder caracterizar cuando una relación se encuentra en **2FN**, será necesario abordar algunas definiciones previas:

Dependencia Funcional (DF):

Dada una relación R se dice que el atributo Y de R es funcionalmente dependiente del atributo X de R, si y sólo si, cada valor X en R tiene asociado a él, precisamente, un valor de Y en R en cualquier momento del tiempo.

Ejemplo: En la relación **SUMINISTRADOR**

SNOM, **TIPO** y **MUN** son funcionalmente dependientes cada uno de **SNUM**, ya que para un valor de **SNUM** existe un valor correspondiente de **SNOM**, **TIPO** y **MUN**.



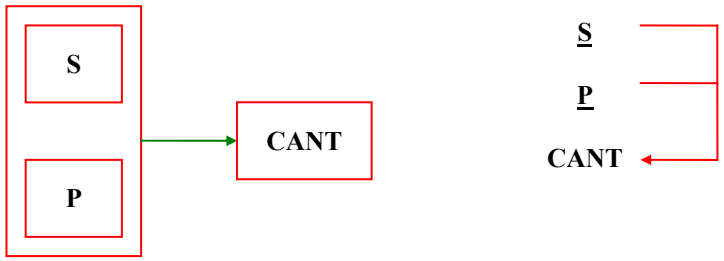
Estas son tres formas posibles de representar las dependencias funcionales.

El reconocimiento de las dependencias funcionales es una parte esencial de la comprensión de la semántica, del significado del dato. El hecho de que **MUN** dependa funcionalmente de **SNUM** significa que cada suministrador esta situado en un municipio.

La noción de dependencia funcional puede ser extendida hasta cubrir el caso en que X, Y o ambos atributos sean compuestos.

Ejemplo: En la relación **SP**

El atributo **CANT** es funcionalmente dependiente del par de atributos **(S,P)**



Dependencia funcional completa

El atributo Y es funcionalmente dependiente y completamente del atributo X, si es funcionalmente dependiente de X y no es funcionalmente dependiente de algún subconjunto de X.

Se representa: **X--->Y**

Ejemplo: En la relación **SUMINISTRADOR**

MUN es funcionalmente dependiente de **(SNUM, SNOM)**, pero no es funcionalmente dependiente y completamente de (**SNUM**, **SNOM**), ya que también es funcionalmente dependiente de **SNUM**.

Ejemplo: En la relación **SP**

CANT es funcional y completamente dependiente de **(S,P)**

Determinante

Cualquier atributo del cual depende funcional y completamente cualquier otro atributo. O sea, la parte izquierda de la implicación cuando la dependencia funcional es completa.

Ejemplo: En la relación **SUMINISTRADOR**:

SNUM es un determinante.

Al hablar de entidad en el MER, se asumió que existe una llave, un conjunto de atributos que definen de forma única una entidad. Un concepto análogo se define para las relaciones.

Llave

Si R es una relación con atributos A1, A2, ...An y X es un subconjunto de A1, A2, ...An, X es una llave de R si:

1. **X---> A1, A2, ...An**

   o sea, todos los atributos de la relación dependen funcionalmente de X.

2. $\nexists$ **Y** $\subset$ **X** | **Y** $\rightarrow$ **A1, A2, ...An**

   (esta condición de minimalidad no se requería para el concepto de llave en el MER).

Una relación puede tener varias llaves. Una de ellas se nombra "**llave primaria**" (la que se escoja para trabajar) y las restantes se nombran "**llaves candidatas**".

Una superllave será cualquier superconjunto de una llave.

Entonces, una llave es un caso especial de superllave.

Procedimiento para hallar la llave

Supongamos una relación R(A,B,C,D,E) con las dependencias funcionales:

1. A-->B 2. BC-->D 3. AB->E

Para comenzar, se parte de que no pueden existir más llaves que dependencias funcionales, pues el concepto de llave incluye la existencia de dependencia funcional. Se analiza, por tanto, cada una de las **DF** presentes en la relación. Añadiendo los atributos que sean imprescindibles en la parte izquierda para lograr determinar a todos los atributos de la relación. El conjunto de atributos así formado debe ser mínimo.

Luego se analiza cada uno de esos conjuntos mínimos, de forma que, si alguno es un superconjunto de otro, ya no es llave, sino superllave. Pueden resultar varias llaves candidatas.

En el ejemplo anterior tendremos:

1. A-->B entonces **AC-->A B E C D** ------> **AC es llave**

2. BC-->D entonces BCA-->B C D A E ------> **BCA es superllave**

3. AB->E entonces ABC-->A B E C D ------> **ABC es superllave**

La única llave es **AC**. No hay ninguna otra llave candidata, pues en las otras **DF** se obtiene el mismo conjunto (**ABC**) el que contiene a **AC**.

Ejemplo: Sea la relación **R** (**ciudad**, **calle**, **código postal**)
C A P

donde se tiene que:

CA--->P (una calle en una ciudad tiene un código postal).

P---->C (el código postal tiene una estructura tal que su valor determina la ciudad, pero en una ciudad, varias calles pueden tener el mismo código, por lo que no se cumple **P--->A**.

Entonces, **CA** es llave, ya que determina a todos los atributos de **R** (determina a P y obviamente a si mismo), o sea, **CA--->CAP**.

P determina a **C** pero no a **A**, pero haciendo: **PA--->CA**, es obvio que **PA--->PA** también, entonces **PA--->CAP**, por tanto, **PA** también es llave.

PA y **CA** son llaves candidatas. En ninguno de estos casos un subconjunto de ellas es también llave.

Un atributo Ai de R es un atributo llave (ó principal) si él es (o es miembro de) una llave (candidata o primaria).

Un atributo Aj de R es un atributo no llave (ó secundario) si él no es miembro de ninguna llave.

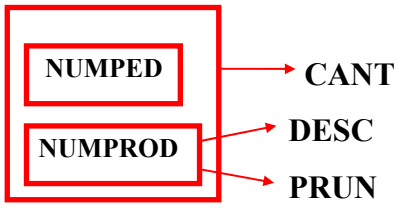
En el ejemplo tendremos que: **C**, **A** y **P** son todos atributos llaves ó principales.

Segunda Forma Normal

Una relación R se dice que está en 2FN si y sólo sí:
- **Está en 1FN**
- **Los atributos no llaves (secundarios) de R, si los hubiese, son funcional y completamente dependientes de la llave primaria de R, o sea no exista dependencia parcial de los atributos secundarios respecto a la llave.**

Entonces, este segundo paso en la normalización se aplica sólo a relaciones que tengan llaves compuestas.

Así en el ejemplo de los Pedidos de Productos, habíamos visto que en la relación **PED-PROD** subsistían problemas de actualización. Analicemos las **DF** existentes en dicha relación:



Esta relación no está en **2FN**, pues los atributos secundarios **DESC** y **PRUN** no dependen funcional y completamente de la llave compuesta **(NUMPED, NUMPROD**).

Para llevarla a **2FN** es necesario crear:

1. Una relación que contenga a todos los atributos secundarios que dependen funcional y completamente de la llave y los atributos que forman la llave.

2. Una relación para los atributos secundarios que dependan de parte de la llave, añadiendo a ella el atributo principal del cual estos atributos dependen completamente.

Así se tendrá las relaciones:

PED-PROD (**NUMPED**, **NUMPROD**, CANT)

PRODUCTO (**NUPROD**, DESC, PRUN)

Con la **2FN** serán resueltos los problemas planteados en la **1FN**, veamos:

Inserción: Se puede insertar la información de un nuevo producto sin necesidad que tenga que existir un pedido del mismo.

Supresión: Se puede eliminar una línea de pedido y no se pierde la información sobre el producto, aunque sea el único pedido en que se pide ese producto.

Modificación: Si cambia un atributo del producto, sólo hay que cambiarlo en un lugar. Se elimina redundancia.

Pero aún persisten problemas de actualización similares a lo visto, pero en la relación **PEDIDO** y, específicamente, cuando se trata de insertar, eliminar o modificar la información de PROVEEDORES, veamos:

Inserción: No podemos insertar información de proveedores en que no haya un pedido para él.

Supresión: Se perderá la información sobre un proveedor al borrar un pedido en que sea el único para ese proveedor.

Modificación: Para cambiar la información sobre un proveedor, hay que hacerlo en todos los tuplos de pedidos en que aparezca ese proveedor.

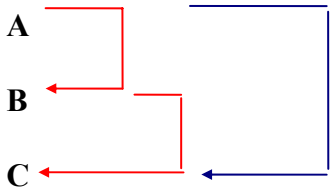
Tercera Forma Normal

Una relación R está en 3FN si y sólo si:
- **Se encuentra en 2FN**
- **No exista dependencia entre los atributos secundarios**

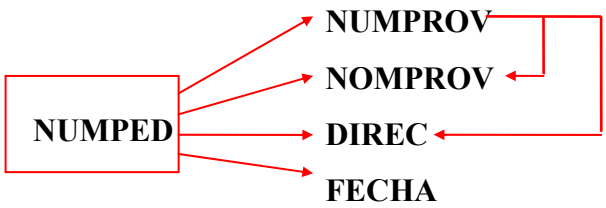
Esto es lo mismo que decir que se deben eliminar las dependencias transitivas de atributos secundarios respecto a la llave, estando ya la relación en **2FN**.

Dependencia transitiva:

Sean **A**, **B** y **C** conjuntos de atributos de una relación **R**. Si **B** es dependiente funcionalmente de **A** y **C** lo es de **B**, entonces C depende transitivamente de A.



Este paso se ejecuta examinando todas las relaciones para ver si hay atributos no secundarios que dependan unos de otros. Si se encuentran, se forma una nueva relación con ellos.



PEDIDO (NUMPED, NUMPROV, FECHA)

PROVEDORES (NUMPROV, NOMPROV, DIREC)

(Si se analizan las otras relaciones, podrá ver que no hay dependencia entre atributos secundarios)

La **3FN** ha elimina los problemas de actualización sobre el proveedor señalados en la **2FN**.

Ya en esta etapa se puede optimizar la **3FN**. Puede haber relaciones degeneradas que contengan sólo la clave, por lo que se pueden eliminar. Puede que otras relaciones tengan la misma clave, por lo que se pueden combinar en una sola, siempre que el resultado sea lógico y tenga sentido.

Los analistas y diseñadores con experiencia producen relaciones en **3FN** casi sin darse cuenta, sin tener que preocuparse para ello, este se debe a que utilizan el sentido común y la experiencia para escribir relaciones ya normalizadas. No obstante, no siempre la instuición es suficiente y la metodología para normalizar las Bases de Datos se convierte en una herramienta imprescindible, con la que se garantiza un diseño idóneo de los datos.

Aún en la **3FN** pueden existir problemas y habrán de ser resueltos con las siguiente forma normal:

Forma Normal de Boyce-Codd (FNBC)

La definición de la 3FN puede resultar inadecuada en el caso de una relación donde ocurre lo siguiente:

1. La relación tiene varias llaves candidatas
2. Las llaves candidatas son compuestas
3. Las llaves candidatas se solapan (o sea, tienen al menos un atributo común).

Una relación donde no tengan lugar las tres condiciones anteriores, la **FNBC** coincidira con la **3FN,** por lo cual no será necesario analizar si la relación cumple la **FBC** en caso contrario será necesario realizar el análisis.

Esas tres condicione**s son necesarias**, **pero no suficientes**, para que la relación viole la **FNBC,** más adelante con un ejemplo veremos esto. Caractericemos primero cuando una relación se dice que cumple la **FNBC**

Forma Normal de Boyce-Codd FNBC)

Una relación R está en FNBC si, y sólo si cada determinante es una llave (candidata o primaria).

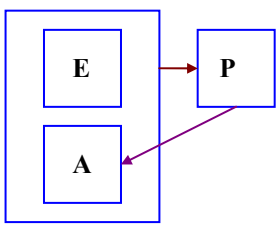
Ejemplo1.-

Sea la relación **EAP** (Estudiante, Asignatura, Profesor), donde una tupla significa que un estudiante **E** recibe la asignatura **A** impartida por el profesor **P** y en la cual se cumple:

- Para cada asignatura, cada estudiante tiene un solo profesor.
- Cada profesor imparte sólo una asignatura.
- Cada asignatura es impartida por varios profesores.

O sea, EA--->P
P--->A

y no existe DF de P con respecto a A



pero si **P--->A**, entonces **EP--->A** también, por lo que **EP** es una llave candidata de la relación, ya que determina a todos los atributos (**E**, **P**, **A**). Ambas llaves candidatas (**EA** y **EP**) son compuestas y se solapan, por lo que debe analizarse la **FNBC**

Esta relación está en **3FN**, pero no en **FNBC**, ya que el determinante **P** no es llave de la relación.

Así por ejemplo si:

E	**A**	**P**
Pérez	Matemática	Blanco
Pérez	Física	Valdés
Rodríguez	Matemática	Blanco
Rodríguez	Física	Hernández

Esta relación presenta anomalías de actualización, por ejemplo:

Eliminación: Si queremos eliminar la información de que Rodríguez estudia Física, perdemos la información de que el profesor Hernández imparte Física.

Estos problemas pueden ser resueltos sustituyendo la relación **EAP** por las relaciones:

EP (E,P) y **PA** (P,A)

O sea, separando la **DF** problemática (**P--->A**) en una relación independiente, donde el determinante será la llave (P), y se forma la otra relación de manera que dicho determinante P participará como atributo, pero en la cual no puede participar el atributo determinado en la dependencia funcional conflictiva o sea **A**

Ejemplo2.-

Sea la relación **R1** (C, A, P)

C- Ciudad
A- calle
P- código postal

donde **CA ---> P** y **P ---> C**, con llaves candidatas **CA** y **PA**.

Esta relación no está en **FNBC**, ya que existe un determinante (P) que no es superllave.

La solución será dividir **R1** en:

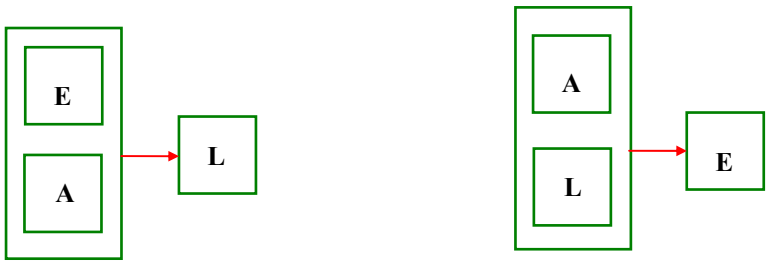
R2 (P, C) y **R3 (A,P)**

Ejemplo3.-

Sea la relación **EXAM** (E, A, L) donde:

- E - Estudiante
- A - Asignatura
- L . Lugar en el escalafón alcanzado por un estudiante en una asignatura y en la que se supone que 2 estudiantes no pueden alcanzar el mismo lugar en una asignatura.

Entonces:



Como se ve, existen dos llaves candidatas, **EA** y **AL**, las cuales se solapan, pero no existen otros determinantes que no sean esas llaves candidatas (que son casos especiales de superllaves), por lo que la relación **EXAM** está en **FNBC**.

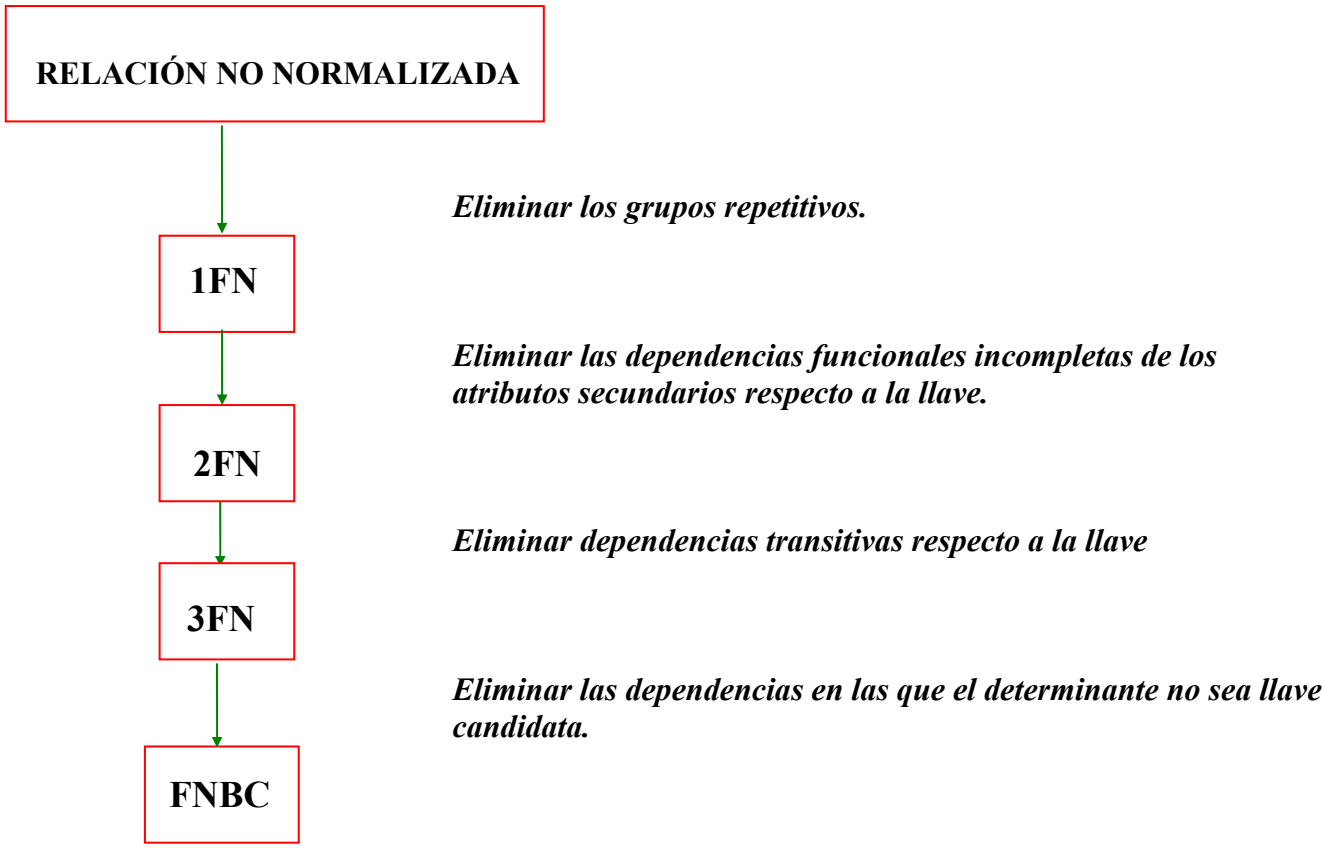
Por último, es necesario destacar que la descomposición de una relación para obtener la **FNBC** puede implicar que se pierdan dependencias funcionales. Por ejemplo en la relación **EAP** donde **EA-->P** y **P-->A** al descomponerse en

EP (E,P) y **PA** (P,A)

se pierde la DF **EA-->P** ya que esta DF no puede ser deducida a partir de las que existen en **EP** y **PA** (que sólo es P-->A), lo cual implica que ambas relaciones **EP** y **PA** no pueden ser actualizadas "**independientemente**", sino que una actualización en un lugar conlleva un chequeo de actualización en el otro lugar (para añadir un profesor a un estudiante hay que chequear la materia que el profesor

imparte y hay que chequear si el estudiante ya tiene un profesor de esa materia y en ese caso no se puede añadir).

Entonces, para el proceso de normalización se deben dar los siguientes pasos:



EJEMPLO DE NORMALIZACIÓN HASTA FNBC.

Se desea diseñar una BD para controlar la disponibilidad de materiales de construcción. De cada proveedor de materiales se conoce su código (**CPROV**), que lo identifica, su nombre (**NOMPROV**) y el municipio en que radica (**MUN**).

De cada material se sabe su código (**CMAT**), que lo identifica, su descripción (**DESC**), la unidad de medida que se aplica al material (**UM**) y el precio por unidad de medida (**PRECIO**).

Para guardar estos materiales hasta su posterior distribución existen diversos almacenes. De cada almacén se conoce su código (**CALM**), que lo identifica, su dirección (**DIRALM**) y la capacidad de almacenaje (**CAPAC**).

Un proveedor puede suministrar varios materiales y un material puede ser suministrado por diferentes proveedores. Se sabe que un material suministrado por un proveedor está en un solo almacén y además, se sabe qué cantidad de un material suministrado por un proveedor se encuentra en el almacén (**CANTMAT**). En un almacén sólo se guarda un tipo de material, aunque puede proceder de distintos proveedores y pueden existir varios almacenes donde se guarde un mismo material.

Veamos las dependencias funcionales que se derivan de esta situación:

Como **CPROV** identifica al proveedor entonces,

1) **CPROV --> NOMPROV, MUN** es el determinante de **NOMPROV** y **MUN**

2) **CMAT --> DESC, UM, PRECIO** análogo a lo anterior pero para materiales

3) **CALM --> DIRALM, CAPAC, CMAT, DESC, UM, PRECIO**
1 2 3 4 5 6

Esta última dependencia funcional tiene la siguiente explicación:

Los atributos 1 y 2 dependen funcionalmente de **CALM** pues **CALM** identifica al almacén. Pero se sabe que en un almacén sólo se guarda un material, por lo que conocido **CALM** se conoce **CMAT** y, por lo tanto, el atributo 3 depende funcionalmente de **CALM**, pero como 4, 5 y 6 dependen funcionalmente de 3, ellos están incluidos también en la dependencia funcional expresada anteriormente 3).

4) **CPROV, CMAT --> NOMPROV, MUN, DESC, UM, PRECIO, CALM, DIRALM,**
1 2 3 4 5 6 7

CAPAC, CANTMAT
8 9

En esta dependencia funcional los atributos:

- 1 y 2: por lo explicado para 1)
- 3, 4 y 5 por lo explicado para 2)

- 6 porque dado un proveedor y un material se conoce en qué almacén se guarda ese material suministrado por ese proveedor.

- 7 y 8 porque 6 se incluye y entonces vale lo explicado para 3)
- 9 porque dado un proveedor y un material se conoce también en qué cantidad se guarda en el almacén correspondiente.

5) **CPROV, CALM --> NOMPROV, MUN, DIRALM, CAPAC, CMAT, DESC,**
1 2 3 4 5 6

UM, PRECIO, CANTMAT
7 8 9

Se incluyen en esta dependencia funcional los atributos:

- 1 y 2 por lo explicado para 1)

- 3, 4, 5, 6, 7 y 8: por lo explicado para 3)

- 9 porque se incluyen en los atributos implicados el atributo 5 y **CPROV** y 5 determinan 9, según lo explicado para el atributo 9 en 4)

Entonces:

CPROV, CMAT son ambas llaves candidatas ya que todos los atributos dependen funcionalmente de ellas

y

CPROV, CALM no existe ningún subconjunto propio del cual todos los atributos dependan.

Escojamos como llave primaria **CPROV CMAT**

R (CPROV, CMAT, NOMPROV, MUN, DESC, UM, PRECIO, CALM, DIRALM, CAPAC, CANTMAT)

1FN

La relación original está en 1FN, pues no existen grupos repetitivos.

2FN

No está en 2FN pues existen dependencias funcionales incompletas de atributos no llaves respecto a la llave primaria (según se aprecia en las dependencias funcionales 1) y 2)) por lo que se separan las relaciones siguientes:

(I) **PROVEDOR** (**CPROV**, NOMPROV, MUN)

(II) **MATERIAL** (**CMAT**, DESC, UM, PRECIO)

3FN

Las relaciones anteriores (I y II) están en 3FN, pero la relación restante de la original, no, ya que existe dependencia transitiva respecto a la llave primaria pues:

CPROV, CMAT --> CALM y

CALM --> DIRALM, CAPAC

por lo que se separa en la siguiente relación :

(III) **ALMACEN** (**CALM**, **DIRALM**, **CAPAC**)

FNBC

Luego de haberse obtenido, es decir, separado, las relaciones I, II y III al analizarse la 2FN y la 3FN, resta la relación:

(1) **SUMINISTRO** (**CPROV**, **CMAT**, **CALM**, **CANTMAT**)

que está en **3FN**. Analicemos las dependencias funcionales que existen en esta relación:

a. **CPROV, CMAT --> CALM, CANTMAT**

b. **CALM --> CMAT**

Luego no está en **FNBC**, pues existe un determinante (**CALM**) que no es superllave.

Para llevarla a **FNBC** hay que separar la dependencia funcional problemática (b.), obteniéndose de la relación suministro las siguientes:

(2) **SUMINISTRO** (**CPROV**, **CALM**, CANTMAT) en esta relación no puede aparecer **CMAT.**

(3) **DISTRIBUCION** (**CALM**, **CMAT**)

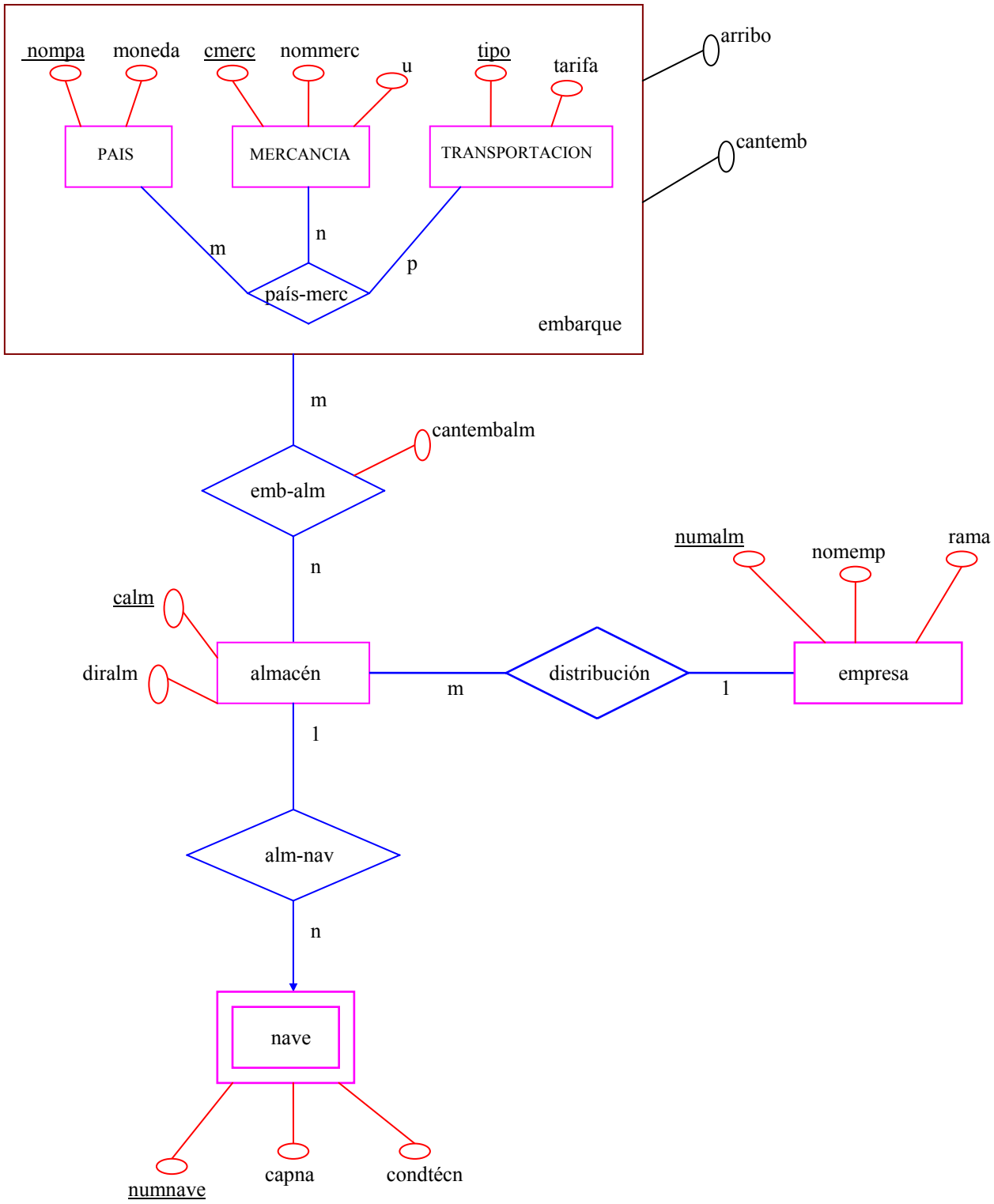
Ahora bien, debe señalarse que al descomponerse la relación suministro (1) de esta manera (suministro (2) y distribución) deja de poder expresarse el hecho de que el **CALM** depende del conjunto **CPROV**, **CMAT**.

Finalmente, analizando la relación distribución (3) y la relación almacén (III) puede apreciarse que ambas pueden unirse: tienen la misma llave y tiene sentido la unión de ambas. Entonces las relaciones obtenidas son:

- **PROVEEDOR (CPROV, NOMPROV, MUN)**
- **MATERIAL (CMAT, DESC, UM, PRECIO)**
- **ALMACEN (CALM, DIRALM, CAPAC, CMAT)**
- **SUMINISTRO (CPROV, CALM, CANTMAT)**

Obtención del Modelo Lógico Global a partir del DER

Para obtener el modelo lógico global de los datos según el enfoque relacional a partir del **DER**, se sigue un procedimiento que iremos describiendo paso a paso a medida que lo apliquemos al ejemplo siguiente:


nompa
moneda
cmerc
nommerc
u
tipo
tarifa
arribo
cantemb
PAIS
MERCANCIA
TRANSPORTACION
m
n
p
país-merc
embarque
m
cantembalm
emb-alm
n
numalm
nomemp
rama
calm
diralm
almacén
m
distribución
l
empresa
l
alm-nav
n
nave
numnave
capna
condtécn

En el **DER** anterior se representa el fenómeno del movimiento mercantil de un organismo. En el organismo existen mercancías de las que se conoce su código, nombre y unidad de medida. Las mercancías proceden de diferentes países de los que se sabe nombre y área de moneda. Para la transportación de las mercancías existen diversas formas, cada una de las cuales se caracteriza por su tipo (barco, avión, tren, etc.) y tarifa. De un país se reciben muchas mercancías y una mercancía puede venir de muchos países. Para cada mercancía de un país existen diferentes formas de transportación y una forma de transportación puede serlo de diferentes mercancías de diferentes países. Una mercancía procedente de un país transportada de una forma dada constituye un embarque y para éste se conoce su fecha de arribo y cantidad.

Un embarque se distribuye entre diferentes almacenes y en un almacén se tienen diferentes embarques, cada uno en cierta cantidad. De cada almacén se tiene su código y dirección. Un almacén envía sus productos a una sola empresa y cada empresa recibe productos de diferentes almacenes. Una empresa se caracteriza por su número, nombre y rama económica.

Cada almacén tiene distintas naves subordinadas. De cada nave se conoce su número (que se puede repetir en diferentes almacenes), capacidad y condiciones técnicas.

PASO # 1
Representar cada entidad regular en una tabla relacional.

a. país (<u>nompa</u>, moneda)
b. mercancía (<u>cmerc</u>, nommerc, um)
c. transportación (<u>tipo</u>, tarifa)
d. almacén (<u>calm</u>, diralm)
e. empresa (<u>numemp</u>, nomemp, rama)

PASO # 2
Representar en una tabla relacional cada entidad agregada con sus correspondientes atributos (entre ellos un identificador si fue definido) y, las llaves de las entidades que forman la agregación

embarque (<u>nompa</u>, <u>cmerc</u>, <u>tipo</u>, arribo, cantemb)

Nótese que la llave estaría formada por las llaves de las 3 entidades regulares que intervienen en la agregación.

Pero podía haberse definido un identificador para la entidad embarque (Añádase en el DER lo que está oscuro: idemb). Entonces se añadiría como atributo llave en la agregación y los 3 atributos nompa, cmerc y tipo permanecerían en la relación pero no como llaves:

embarque (**idemb**, nompa, cmerc, tipo, arribo, cantemb)

PASO # 3

- **Representar cada entidad generalizada en una tabla que contendrá sus atributos (sólo los de la generalizada) y, entre ellos, la llave.**
- **Representar cada entidad especializada en una tabla que contendrá la llave de la generalización y los atributos propios sólo de la especialización.**

En el ejemplo que estamos desarrollando no tenemos Generalizacion-Especialización

PASO # 4

Representar en una tabla relacional cada relación de n : m, incluyendo las llaves de las entidades relacionadas y los atributos de la relación si los hubiese.

emb-alm (nompa, cmerc, tipo, calm, cantembalm)

Esta relación sería de esta forma sin considerar la existencia del idemb. Caso de que éste estuviera definido quedaría:

emb-alm (idemb, calm, cantembalm)

ya que entonces la llave de la entidad embarque sería idemb.

PASO # 5

Para cada relación de 1 : m, añadir la llave de la entidad del extremo "1" como un nuevo atributo a la entidad del extremo "m" y los atributos de la relación si existe n.

En nuestro ejemplo, se da la relación

almacén <<----------> empresa

por lo que la llave de la entidad empresa (numemp) debe añadirse como atributo en la tabla que representa la entidad almacén. Esto hace que la relación d, obtenida en el paso # 1 quede modificada de la siguiente manera:

almacén (calm, diralm, numemp)

Al agregarse el atributo numemp en la relación almacén se sabe entonces a qué empresa abastece el almacén. Está claro que muchas ocurrencias de almacén pueden tener el mismo valor en el atributo numemp, pues una empresa recibe mercancías de varios almacenes, en general.

PASO # 6
Representar cada entidad débil en una tabla relacional que contendrá la llave de la entidad regular determinante y el identificador de la entidad débil con sus atributos.

nave (calm, numnav, capnav, condtécn)

Estas son las tablas relacionales que se derivan del **DER** del ejemplo. Ahora es preciso analizar cada una de las relaciones para asegurarse que están normalizadas hasta **FNBC**.

En este caso, puede verse que todas las relaciones cumplen con la **FNBC**, por lo que no es preciso hacer nada más.

En caso de que no estuvieran debidamente normalizadas, era preciso aplicar el proceso de normalización tal y como lo vimos anteriormente.

Metodología para el diseño de bases de datos.

- Determinación de entidades y atributos
- Normalización de entidades
- Determinación de relaciones (DER)
- Obtención del modelo lógico global de los datos
- Diseño físico de la BD

Cuando se va a realizar el diseño de la base de datos para un sistema determinado, es necesario determinar los datos que es necesario tomar en cuenta y las dependencias funcionales existentes entre ellos.

Rigurosamente, esto se obtiene luego de realizada la etapa de análisis del sistema y partiendo de lo obtenido en ésta, pero trataremos de dar indicaciones que permitan lograr un buen diseño de la BD a partir de lo que conocemos.

La situación a partir de la cual se discutirá la metodología para diseñar la base de datos es la siguiente:

En un hospital Clínico Quirúrgico se desea automatizar el control de la actividad quirúrgica. En el hospital se tienen varios quirófanos y se realizan intervenciones quirúrgicas en diferentes especialidades. Las intervenciones quirúrgicas pueden ser electivas (planificadas) o urgentes (no planificadas). Se tiene la planificación de las operaciones quirúrgicas a realizar cada día. Las operaciones planificadas pueden ser suspendidas por distintas causas. Para cada operación realizada (haya sido planificada o no), se confecciona un informe operatorio. En cada operación realizada se emplean materiales cuyo gasto se controla.

Con el sistema automatizado se pretende obtener las siguientes salidas:

PROGRAMACION QUIRURGICA					**Para el día:** **de** **de 19**			
Paciente	**Hist Clín.**	**Edad**	**Sexo**	**Diagnóstico Pre-Oper.**	**Intervenc. Indicada**	**Hora**	**Salón**	**Cirujano**
.								. .

OPERACIONES SUSPENDIDAS			**En el día:** **de** **de 19**
Paciente	**Historia Clínica**	**Edad**	**Causa**
. .			. .

INFORME OPERATORIO	Fecha de Operación:	
	Hora Comienzo	Hora Terminación

Paciente	Historia Clínica	Edad

Diagnóstico Clínico: _____
Operación Realizada: _____
Diagnóstico Operatorio: _____

Situación Final del Paciente:

☐ **Satisfactoria** ☐ **No Satisfactoria** ☐ **Fallecido**

Especialidad	Tipo de Operación Realizada
	☐ **Electiva** ☐ **Urgente**

Cirujano: _____

RESUMEN DE ACTIVIDADES QUIRURGICAS	Mes:

Cantidad de Operaciones Electivas: _____
Cantidad de Operaciones Urgentes: _____

Especialidad	Cantidad de Operaciones

GASTO DE MATERIALES DE OPERACION						En el día: de de 19		
Paciente	**Hist. Clín.**	**Edad**	**Hora Com.**	**Patología**	**Cód. Mat.**	**Descr. Mater.**	**Cantidad**	**Costo**

Para comenzar a estudiar la metodología de diseño de la base de datos, partiremos de una premisa: lo que se obtiene con un sistema automatizado son sus salidas, es decir, el resultado que se desea obtener, lo que llega al usuario, son justamente las salidas del sistema y para ello se desarrolla; por lo tanto, lo que no sea necesario para obtener las salidas del sistema, no tiene por qué estar almacenado en la base de datos. En fin, lo que debe almacenarse en la base de datos es lo que es imprescindible para obtener las salidas.

Es por esta razón que para el diseño de la base de datos, se parte del análisis de las salidas que se desea obtener con el sistema.

I.- Determinar las entidades y los atributos

Para cada salida:

- Consultar su formato
- Determinar datos que se calculen u obtengan a partir de otros e ir sustituyéndolos hasta llegar a los primarios
- Determinar existencia de ficheros con información normativa y/o de consulta
- Cada salida y cada fichero, tomarlos como entidad y relacionar sus atributos

En el ejemplo, representando cada una de las salidas por una relación, se tienen las sgtes. relaciones o entidades iniciales:

1.
PQ (Fecha, NomPac, HC, Ed, Sexo, DiagPre, Interv, Hora, Salón, Cirujano)

2.

OS (Fecha, NomPac, HC, Ed, Causa)

3.

IO (Fecha, HoraCom, HoraTerm, NomPac, HC, Ed, DiagClín, Operac, DiagOper, SitPac, Espec, TipoOpReal, CirReal)

4**.**

RAQ (Mes, Elect, Urg, Espec, Cant)

5.

GMO (Fecha, NomPac, HC, Ed, HoraCom, Patología, CódMat, DescMat, Cant, Costo)

Analizando cada dato se ve que:

En 1.

Hora se calcula según el tiempo promedio de demora para cada operación., por lo que es necesario un fichero normativo:

=> **TPO** (Operación, Tiempo Prom)

Salón se decide teniendo en cuenta la distribución de los salones, que en este caso, es por día de la semana:

=> **DS** (Salón, DíaSem, Espec)

En 2.

Todos son datos primarios

En 3.

Todos son datos primarios

En 4.

Elect, **Urg** y **Cant** son datos que se calculan a partir de contar en IO (relación 3), por lo que la relación 4 se puede eliminar completamente para el análisis, pues al ser calculables todos los datos contenidos en ella, no aporta información nueva a almacenar en la base de datos.

En 5.

Costo = cant x precio, por lo que se sustituye costo por precio (ya cant está en el modelo)

=> necesidad de existencia de un fichero de consulta **MATERIAL** (puede salir en entrevistas a usuarios)

II. Normalizar las entidades

- Determinar las DF
- Normalizar cada entidad
- Fusionar, de ser lógico, las entidades normalizadas que tengan la misma llave

1.
PQ (Fecha, NomPac, HC, Ed, Sexo, DiagPre, Interv, Hora, Salón, Cirujano)

Normalizando:

a. **Paciente** (HC, NomPac, Ed, Sexo)

b. **PQ** (Fecha, HC, DiagPre, Interv, Hora, Salón, Cirujano)

2.
OS (Fecha, NomPac, HC, Ed, Causa)

Normalizando:

c. **Paciente1** (HC, NomPac, Ed)

d. **OS** (Fecha, HC, Causa)

3.
IO (Fecha, HoraCom, HoraTerm, NomPac, HC, Ed, DiagClín, Operac, DiagOper, SitPac, Espec, TipoOpReal, CirReal)

Normalizando:

2FN:

Paciente2 (HC, NomPac, Ed)

PlanOperac (Fecha, HC, DiagClín, Operac, Espec)

IO (Fecha, HoraCom, HC, HoraTerm, DiagOper, SitPac, TipoOpReal, CirReal, OperReal)

3FN:

e. **Paciente2** (HC, NomPac, Ed)

f. **PlanOperac** (Fecha, HC, DiagClín, Operac)

g. **ClasifOperac** (Operac, Espec)

h. **IO** (Fecha, HoraCom, HC, HoraTerm, DiagOper, SitPac, TipoOpReal, CirReal, OperReal)

4. (Se elimina para este análisis)

5.
GMO (Fecha, NomPac, HC, Ed, HoraCom, Patología, CódMat, DescMat, Cant, Precio) (Precio en lugar de Costo)

i. **Paciente3** (HC, NomPac, Ed)

j. **Diagóstico** (Fecha, HC, Patología)

k. **Material** (CódMat, DescMat, Precio)

l. **GMO** (Fecha, HC, HoraCom, CódMat, Cant)

Además, se tienen los dos ficheros normativos o de consulta:

m. **TPO** (Operación, Tiempo Prom)

n. **DS** (Salón, DíaSem, Espec)

Fusionando las entidades normalizadas con igual llave, siempre que sea lógico, se tiene:

. De a, c, e y i ==>

Paciente (HC, NomPac, Ed, Sexo)

. De b, d, f y j ==>

f está contenida en b y j está contenida en b también

(**DiagPre = DiagClín = Patología**) Son alias diferentes para indicar un mismo atributo, que tiene el mismo dominio

Pero d, a pesar de tener igual llave, está representando otra entidad, la Operación Suspendida; semánticamente, no representa lo mismo que una Operación Planificada y, por lo tanto, no tiene

sentido unirlas. Es más, si se hiciera, todas las operaciones tendrían el atributo Causa, y en la mayoría de las ocurrencias, este atributo estaría vacío, pues sólo son algunas las operaciones que se suspenden.

PQ (Fecha, HC, DiagPre, Interv, Hora, Salón, Cirujano)

OS (Fecha, HC, Causa)

. De m y g ==>

Intervención (Operación, TiempoProm, Espec)

. De n ==>

DS (Salón, DíaSem, Espec)

. De h ==>

IO (Fecha, HoraCom, HC, HoraTerm, DiagOper, SitPac, TipoOpReal, CirReal, OperReal)

. De k ==>

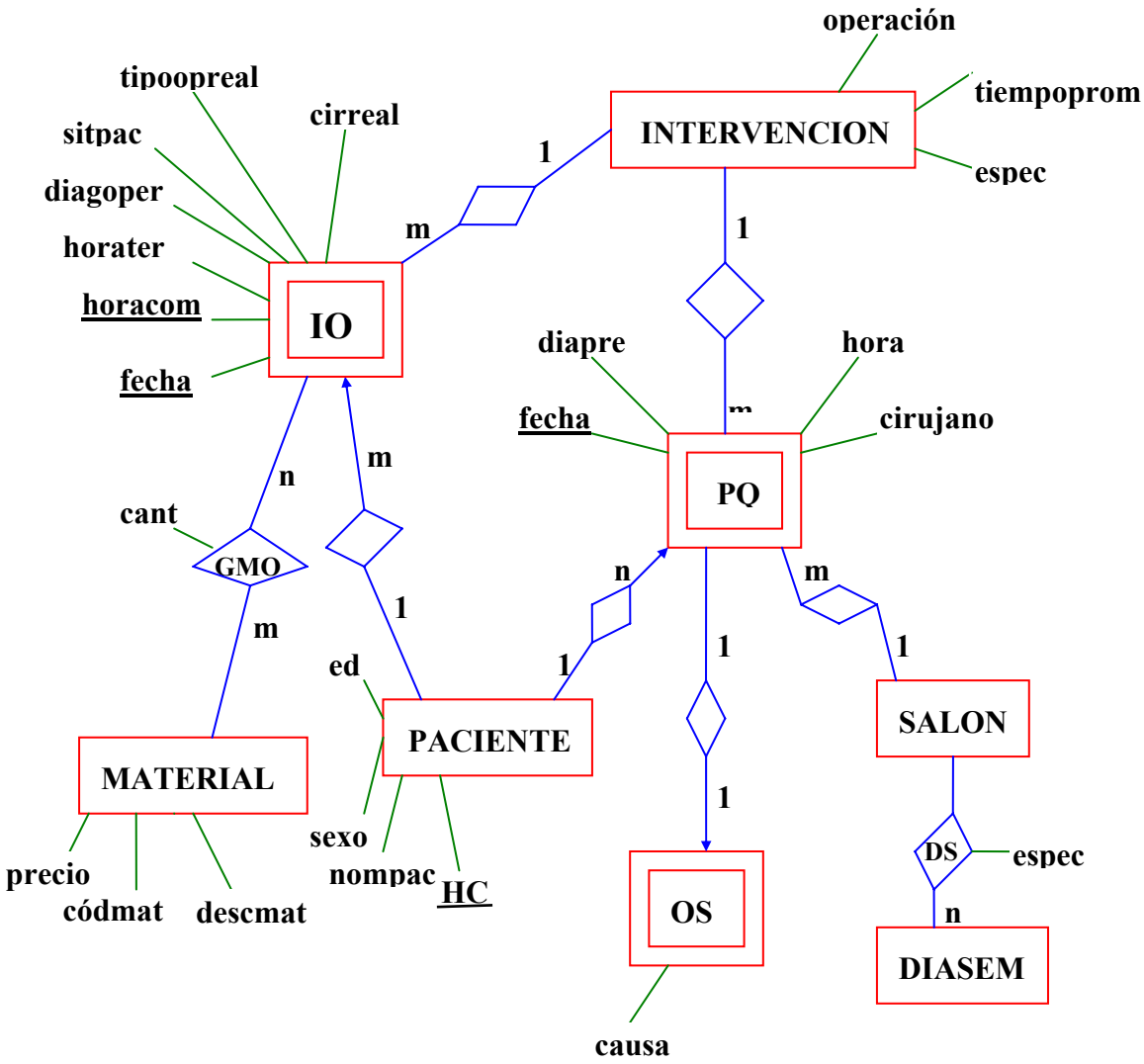
Material (CódMat, DescMat, Precio)

. De l ==>

GMO (Fecha, HC, HoraCom, CódMat, Cant)

III. Determinar las relaciones entre las entidades (DER)

- Si hay alguna entidad cuya llave es combinación de llaves de otras entidades, generalmente, es porque representa una relación entre ellas (n : m)
- Analizar cada entidad con las restantes para determinar si existe relación, determinando tipo de proyección.
- Ir confeccionando el DER
- Analizar entidades agregadas, entidades débiles y sus relaciones con otras



En este paso se puede comprobar si lo que se ha realizado es correcto o si surge alguna contradicción, se puede completar el modelo con nuevos elementos que se hayan podido obtener en intercambios con los usuarios, ya que con el DER se tiene un modelo menos abstracto, más cercano al fenómeno. El analista puede proponer nuevas posibilidades a los usuarios y agregarlas en caso de acordarse esto, etc.

IV. Obtener el modelo lógico global de los datos

Aquí se deben seguir los pasos para llevar de DER al modelo lógico global de los datos.

Si en el paso anterior no se modifica el DER obtenido, entonces las tablas que se obtienen en este paso, en general, son las mismas que las obtenidas en el paso II. Supongamos que en nuestro ejemplo, este es el caso. Entonces:

Diseño lógico:

Paciente (HC, NomPac, Ed, Sexo)

Intervención (operación, TiempoProm, Espec)

Material (CódMat, DescMat, Precio)

PQ (Fecha, HC, DiagPre, Interv, Hora, Salón, Cirujano)

IO (Fecha, HoraCom, HC, HoraTerm, DiagOper, SitPac, TipoOpReal, CirReal, OperReal)

OS (Fecha, HC, Causa)

DS (Salón, DíaSem, Espec)

GMO (Fecha, HC, HoraCom, CódMat, Cant)

V. Diseño físico de la BD

En este paso hay que tener en cuenta:

. Aplicaciones a realizar sobre los datos
. Características particulares del SGBD

Ejemplos:

- ficheros índices
- ficheros para reportes
- considerar la inclusión de campos que sean secundarios debido a la cantidad de veces que habría que calcularlos en una determinada aplicación.
- separar un fichero en varios debido a la cantidad de campos y lo que se accesa cada vez.