UCLM-ESI

MODELOS DE DATOS

CONCEPTOS Y CLASIFICACIÓN

Realizado por: Fernando Lorenzo Castro
Para la asignatura de: Bases de datos (Gestión)
Profesor: Francisco Ruiz
Curso: 1999/2000

INDICE

CAPITULO 1:INTRODUCCION

El modelado es una parte central de todas las actividades que conducen a la producción de buen software. Construimos modelos:

- Para comunicar la estructura deseada y el comportamiento de nuestro sistema.
- Para visualizar y controlar la arquitectura del sistema.
- Para comprender mejor el sistema que estamos construyendo, muchas veces descubriendo oportunidades para la simplificación y la reutilización.
- Para controlar el riesgo.

¿Qué es, entonces, un modelo? Un modelo es una simplificación de la realidad.

Un modelo proporciona los planos del sistema. Los modelos pueden involucrar planos detallados, así como planos más generales que ofrecen una visión global del sistema en consideración. Un buen modelo incluye aquellos elementos menores que no son relevantes para el nivel de abstracción dado. Todo sistema puede ser descrito desde diferentes perspectivas utilizando diferentes modelos, y cada modelo es por tanto una abstracción semánticamente cerrada del sistema. Un modelo puede ser estructural, destacando la organización del sistema, o puede ser de comportamiento, resaltando su dinámica.

¿Por qué modelamos? Construimos modelos para comprender mejor el sistema que estamos desarrollando.

A través del modelado conseguimos cuatro objetivos:

- Los modelos nos ayudan a visualizar cómo es o queremos que sea un sistema.
- Los modelos nos permiten especificar la estructura o el comportamiento de un sistema.
- Los modelos nos proporcionan plantillas que nos guían en la construcción de un sistema.
- Los modelos documentan las decisiones que hemos adoptado.

El modelado no es sólo para los grandes sistemas. Sin embargo, es absolutamente cierto que, cuanto más grande y complejo es el sistema, el modelado se hace más importante, y todo ello es porque construimos modelos de sistemas complejos porque no podemos comprender el sistema en su totalidad.

Hay límites a la capacidad humana de comprender la complejidad. A través del modelado, se reduce el problema que se está estudiando, centrándose solo en un aspecto a la vez. Este es, esencialmente el enfoque “divide y vencerás”: acometer un problema difícil dividiéndolo en una serie de subproblemas más pequeños que se pueden resolver. Además, a través del modelado, se potencia la mente humana. Un modelo escogido adecuadamente puede permitir al modelador trabajar a mayores niveles de abstracción.

1.1. PERSPECTIVA HISTÓRICA

Podemos definir una Base de Datos como un conjunto exhaustivo y no redundante de datos estructurados, organizados de forma independiente a su utilización o implantación en máquina, accesibles en tiempo real y compatibles con usuarios concurrentes y sus respectivas necesidades (peticiones) de información.

Es sabido que en el ámbito empresarial, llegó un momento en el que la información de negocios alcanzó un volumen superior al que la capacidad humana puede procesar, en cuestiones de búsqueda, análisis y cruces entre los datos. A partir de ese momento, el ordenador y el proceso computerizado tomaron el testigo, y mediante las Bases de Datos se comenzó a almacenar, procesar y gestionar los datos de las empresas más importantes.

La propia evolución de la Informática, en todos sus aspectos, fue abaratando costes y simplificando procesos. Las herramientas de administración de Bases de Datos se desarrollaron incesantemente (aún hoy día siguen avanzando) hasta convertirse en poderosos motores capaces de manejar enormes cantidades de información en pocos segundos.

La primera solución que proporcionó el tratamiento automatizado de la información para el almacenamiento de datos fueron los sistemas de ficheros. Estos sistemas de ficheros estaban limitados a la escasa potencia de los equipos informáticos, de la época. Los accesos a los medios de

almacenamiento eran muy lentos y solo se podía utilizar un fichero simultáneamente, lo que hacia lentas las referencias entre los datos.

A medida que la necesidad de almacenar y gestionar la información fue creciendo, la Informática ha procurado diferentes soluciones mediante la definición de una serie de modelos.

Estos modelos son un conjunto de pautas, mediante las cuales podemos representar una realidad de forma abstracta mediante datos y la forma de estructurar éstos. Existen tres grandes grupos en los que podríamos englobar estas pautas de representación, y cada uno de ellos genera lo que se ha llamado un modelo de base de datos.

1.2. LOS TRES NIVELES DE CONCEPTO DE ARQUITECTURA

La arquitectura de un sistema de base de datos se divide en 3 niveles comunes, nivel interno, conceptual y externo.

1.2.1. Nivel Interno

Es el más cercano al almacenamiento físico, es decir, es el que se ocupa de la forma como se almacenan físicamente los datos. Representación de bajo nivel de toda la base de datos, se compone de varias ocurrencias, de varios tipos de registros, el nivel interno todavía esta aún paso del nivel físico ya que no se manejan los registros fijos. La vista interna se define a través de un esquema interno el cual no sólo define los diversos tipos de registros almacenados, si no, también especifica los índices asociados, representación de los campos almacenados, secuencia física de los registros, etc.

1.2.2. Nivel Externo

Visión de los usuarios particulares. Habrá tantos niveles externos como usuarios distintos. En definitiva, es el mas cercano a los usuarios, es decir, es el que se ocupa de la forma como los usuarios reciben los datos. Es el nivel del usuario individual, es decir, los usuarios pueden ser programadores en algunos casos usuarios finales, cada usuario dispone de un lenguaje y en el caso de un programador dispone de un lenguaje convencional. En el caso de un usuario final, será un lenguaje de consulta o un Lenguaje orientado hacia los usuarios. El punto importante de todos estos lenguajes es que debe incluir un sublenguaje de datos del cual estará inmerso o dentro de un lenguaje anfitrión, un lenguaje dado, cualquiera va a permitir el empleo de varios lenguajes anfitriones y varios sublenguajes para datos.

Ejemplo:

lenguaje VB o “C” >>>> lenguaje anfitrión
acces >>>>> Sublenguaje.
DBSE >>>>> lenguaje arquitectónico
ASSIST >>>> SUB-LENGUAJE.
SQL >>>> sub-lenguaje.

1.2.3. Nivel Conceptual

Es el nivel de mediación entre los 2 anteriores:

externo	(aplicaciones)
Conceptual	(modelo,(entidad/relación))
Interno	(Hardware)

La vista conceptual es una presentación de toda la información contenida en la base de datos. Además puede ser muy diferente en la forma en que percibe los datos cualquier usuario final, es decir, debe ser un panorama de los datos. Tal como son y no como los percibe los usuarios. Debido a las limitaciones del lenguaje o bien al equipo que se esta utilizando. El nivel conceptual se define mediante un esquema conceptual el cual incluye la definición de cada uno de los tipos de registros

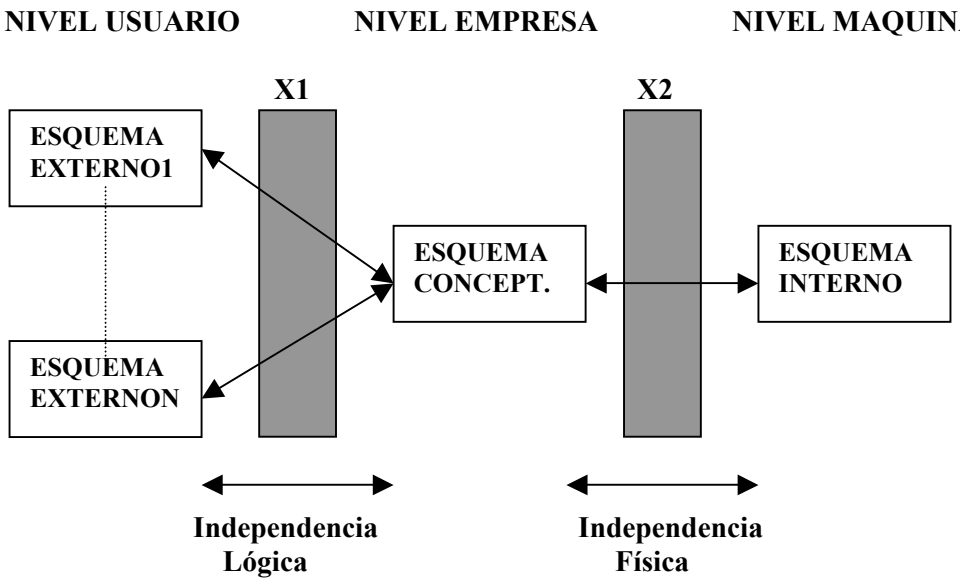
(entidades), además, el esquema conceptual no debe asociarse a representaciones de campos almacenados tales como punteros, índices, etc., si el esquema conceptual se desarrolla en forma independiente de los datos entonces el esquema externo definido en base al esquema conceptual será también independiente de los datos.

1.2.4. Correspondencia entre niveles

1. LA CORRESPONDENCIA ENTRE NIVEL EXTERNO Y NIVEL CONCEPTUAL: Es la que existe entre una determinada Vista Externa y la Vista Conceptual. La diferencia que puede existir entre estos dos niveles son similares a las que pueden existir entre la vista conceptual y la vista interna. Ejemplo: Los campos pueden tener distintos tipos de datos, los nombres de los campos y registros pueden diferir entre sí, pueden combinarse varios campos conceptuales para formar un campo externo.

2. CORRESPONDENCIA ENTRE EL NIVEL CONCEPTUAL Y EL NIVEL INTERNO: Es la que existe entre la vista conceptual y la vista interna especifica como se representan los registros y campos conceptuales, si se modifica la estructura de la Base de Datos, es decir, nivel interno, debe también modificarse la correspondencia para no variar el esquema conceptual.

En la siguiente figura podemos ver las dos funciones de correspondencia, donde los usuarios están aislados de los datos almacenados físicamente en la máquina por las pantallas X1 y X2. Así, la primera representará la independencia lógica, mientras que la segunda representará la independencia física:



1.3. COMPONENTES DE UN DBMS

En un sistema de Base de Datos se distinguen 4 componentes:

- Datos:Los datos almacenados se dividen en una o más Bases de Datos. Por lo tanto, una Base de Datos es un recipiente de datos almacenados y en general es tanto integrada como compartida:
  - Integrada: Se entiende que la Base de Datos puede considerarse como una unificación de varios archivos de datos independientes de donde se elimina parcial o totalmente cualquier redundancia entre los mismos.
  - Compartida: Partes individuales de las Bases de Datos pueden compartirse entre varios usuarios distintos en el sentido de que cada uno de ellos puede tener acceso a la misma parte de la Base de datos y usarla con propósitos distintos. Tal comportamiento

es en verdad consecuencia del hecho de que la base de Datos es integrada. (privilegio que se puede dar por programa a de administración de red).

- Software: Entre las Bases de Datos físicas en sí, es decir, el almacenamiento real de datos y los usuarios existe un nivel de software que a menudo recibe el nombre de DBMS (Software de interfaz entre las bases de datos y usuario). Su objetivo es manejar todas las solicitudes formuladas por los usuarios y/o programas de acuerdo a una función general del DBMS, otro objetivo es proteger a los usuarios contra los detalles a nivel de hardware, es decir, el DBMS ofrece una visión a los usuarios que está por encima del hardware y apoya las operaciones de éste. Ejemplo: Obtener el registro del empleado 205

```
SELECT * NOMBRE
        FROM Xi (TABLA)
WHERE COD_EMPL = 205
```

  El DBMS va a la tabla de la Base de Datos, la busca y lo muestra.
- Hardware: Compuesto por volumen de disco, tambores, etc., donde está residente la Base de Datos. Junto con los dispositivos asociados como unidades de control, canales, etc., ya que se asume que la Base de Datos es demasiado grande para localizarla en memoria.
- Usuarios: Existen 3 tipos de usuarios:
  - Usuario final: Usa la Base de Datos realizando consultas, mantenimiento, etc., a través de un lenguaje de consulta, o a través de un programa mediante el lenguaje de consulta el usuario queda libre para poder hacer cualquier operación y mediante un programa el usuario queda restringido a lo que en éste se estableció.
  - Segundo usuario: Programador de aplicaciones; desarrolla los sistemas necesarios para permitir la posibilidad de comunicación o extensión de información desde la Base de Datos.
  - Administrador de la Base de Datos: Dentro de sus funciones podemos mencionar:-- . . . Mantener en forma optima y eficiente la base de datos controlando procedimientos, instalaciones, procesos, etc.,
    . Realizar funciones de auditoría, manejando la seguridad de la Base de Datos. Además de crear usuarios y accesos permitidos.

Todos estos componentes del DBMS se ocupan de la definición, actualización y recuperación de datos estructurados. En definitiva, las funciones específicas del DBMS son:

- Definición de Datos: Debe ser capaz de aceptar definiciones de datos (esquema externo, conceptual Interno y todas las correspondencias asociadas en versión Fuente) y convertirla en una versión de objeto apropiada. Debe incluir componentes de procesadores de lenguajes para cada uno de los diversos lenguajes de definición de datos, además debe tener las definiciones en DDL (Lenguaje Definición de datos) para poder interpretar y resolver las solicitudes.

- Manipulación de Datos: Debe atender las solicitudes del usuario tales como eliminación, modificación, extracción etc., es decir, debe incluir un componente procesador de lenguajes de manipulación de datos (DML). En general las solicitudes en DML pueden ser planificadas y no planificadas.

EJEMPLOS:

1.-SOLICTUD PLANIFICADA: Es aquella que se estima con anterioridad antes de que sea ejecutada por primera vez y para estos casos el DBA debe garantizar un buen desempeño de estas solicitudes.

2.-SOLICITUD NO PLANIFICADA: Es una consulta, cuya necesidad no estimo en el diseño físico de la BD puede no estar preparada ante este tipo de consultas y la mejor prueba para el DBMS es que pueda responder ante dicha solicitud con un buen desempeño.

3.- SEGURIDAD E INTEGRIDAD DE LOS DATOS: El DBMS debe supervisar las solicitudes de los usuarios y rechazar los intentos de violar las necesidades de seguridad e integridad de los datos definidos para el DBA.

4.- RECUPERACION Y CONCURRENCIA DE LOS DATOS: El DBMS debe cuidar el cumplimiento de ciertos controles de recuperación y concurrencia. El Administrador de transacciones actúa en caso de que el DBMS no funciones.

5.- DICCIONARIO DE DATOS: El DBMS debiera incluir una función de diccionarios de datos (metadatos), se puede decir que es una base de Datos del sistema y no del usuario cuyo contenido

puede considerarse como “Datos acerca de los Datos” que en el fondo son definiciones de objetos de datos y otros objetos del sistema.
6.- DESEMPEÑO. El DBMS debiera ejecutar todas las funciones anteriores en la forma mas eficiente.
7.- ADMINISTRADOR DE COMUNICACIONES DE DATOS: Las solicitudes de un usuario final son dirigidas a la Base de datos donde son transmitidas en forma de mensajes de comunicación o a la inversa las respuestas al usuario son tomados como mensajes del mismo tipo, todas las transmisiones se efectúan bajo el control de otros grupos de programas llamados administrador de comunicación de datos el cual no forma parte del DBMS, sino que es un programa autónomo pero debe trabajar en forma conjunta con el DBMS para poder satisfacer los requerimientos de los usuarios.

Según el comportamiento, tenemos dos tipos de SGBD’s:

- Activos. Tenemos dos objetos enfrentados que son el observador y el observado. Tienen un enfoque no intrusivo, es decir, uno de los objetos no necesita conocer de la existencia del otro. Para describir el comportamiento activo empleamos el modelo de conocimiento. Como ejemplos de sistemas de información que se amoldan al comportamiento activo tenemos la gestión de stocks, la gestión de productos perecederos y la gestión de carteras de valores. Este comportamiento cuenta con varias ventajas como:
    - Promueve la reusabilidad del código.
    - Incrementa la consistencia.
    - Reduce el esfuerzo total del mantenimiento del código.
    - Mejora la eficiencia al reducir el tráfico por la red.
- Pasivos. Al igual que en el anterior, tenemos dos objetos enfrentados que son el fuente y el receptor de la petición. Tienen un enfoque opuesto al anterior comportamiento, es decir, tienen un enfoque intrusivo, con lo que uno de los objetos necesita conocer de la existencia del otro para poder comunicar con él. Para gestionar el comportamiento pasivo empleamos el modelo de ejecución.

CAPITULO 2: CONCEPTOS BÀSICOS

2.1. FUNDAMENTOS TEORICOS

2.1.1. Datos y modelos de datos

2.1.1.1. Significado de los datos

Los datos, que en un primer momento se organizan atendiendo a las necesidades de un proceso para después atender a los requisitos de un conjunto de procesos, ahora, buscan una interpretación de la realidad, para así poder captar la semántica del mundo real.

La percepción del mundo puede ser descrita como una sucesión de fenómenos, que del comienzo de los tiempos el hombre ha tratado de descubrirlos, ya sea, que los entienda completamente o no. La descripción de estos fenómenos recibe el nombre de datos. Los datos corresponden a registros de hechos acerca de un fenómeno, con lo cual ganamos información acerca del mundo que nos rodea. La información pasa a ser el incremento del conocimiento que puede ser inferido de los datos. Usualmente el dato y su significado son registrados juntos, ya que el lenguaje natural es lo suficientemente poderoso para hacerlo.

Por tanto, podemos asegurar que el dato se convierte en información bajo el contexto de una situación, y que las Bases de Datos son las encargadas de mantener y reunir toda la información por medio de unas estructuras de datos sobre las que se puede actualizar, borrar... la información.

2.1.1.2. Definición de modelo

"Conjunto de conceptos que permite construir una representación organizacional de la empresa" Flory (1982).

"Instrumento que aplicado al Universo de Discurso (parte del mundo real) nos da como resultado una estructura de datos (esquema)".

Universo de discurso: Visión del mundo real del diseñador de la Base de Datos. Su definición, constituye el primer paso del diseño para poder conseguir los objetivos del diseñador.

En definitiva, para nosotros, usuarios de las Bases de Datos, el modelo va a ser la herramienta. Para poder hacer una distinción clara de lo que es el modelo y de lo que es el esquema, diremos que este último va a ser el resultado de aplicar la herramienta (modelo).

Modelo = Herramienta.

Esquema = Resultado.

Pero ¿a qué llamamos Universo de Discurso? Pues, se trata de la visiòn que el diseñador tiene acerca del mundo real. Para un mismo mundo real puede haber varios Universos de Discurso; tantos como objetivos estime el diseñador.

2.1.1.3. Objetivos del modelado de datos

Modelar: "Hacer que las conclusiones de un mundo abstracto y teòrico coincidan con las manifestaciones aparentes del mundo real". Flory (1982).

Los principales objetivos del modelado de datos son:

- *Facilitar la interpretación de nuestro Universo de Discurso.*
- *Facilitar la representación de los datos en nuestro sistema de información. Para ello, los modelos de datos se encargan de definir formalmente las estructuras permitidas y las no permitidas (restricciones), estableciendo al mismo tiempo la base para la definición de un lenguaje de datos (modelo de datos + sintaxis) y de desarrollar una metodología de diseño de Bases de Datos que permita prever el impacto de los cambios en un sistema de información.*

2.1.1.4. Definición de modelo de datos

El concepto de modelo de datos es necesario para describir la estructura de una base de datos.

Podemos definirlo, según Flory (1982), como un dispositivo de abstracción que nos permite ver la información de los datos más que su valor concreto.

Podemos definirlo también como: "Dispositivo de abstracción que nos permite ver el bosque (información contenida en los datos) en oposición a los árboles (valores individuales de los datos)".Tsichiritzis y Lochovsky (1982). Es decir, tengo la posibilidad de ver toda la información global.

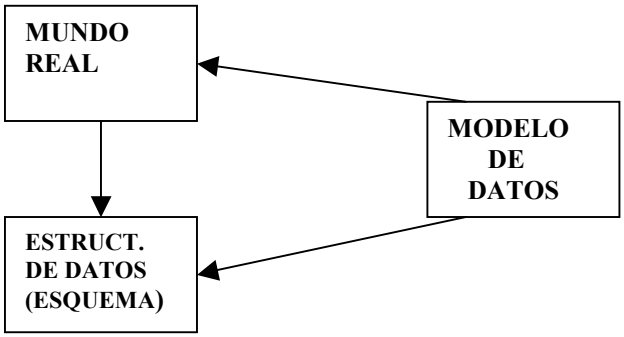
Otra definición posible sería la de un conjunto de conceptos, reglas y convenciones que permiten describir, a distintos niveles de abstracción, los datos del Universo de Discurso o la estructura de una base de datos, a la cual denominamos esquema, o con otras palabras, que permiten construir una representación organizada de un sistema real. Por lo tanto, consideraremos un modelo de datos como una herramienta que me facilita la interpretación del Universo de Discurso y la representación de un sistema de información.

Así, podemos concluir que todos los lenguajes de datos están basados en estos modelos.

Ej:

SQL - Modelo relacional + Sintaxis (SQL)
DL/I – Modelo jerárquico + Sintaxis (DL/I)

Podemos ver gráficamente cómo se puede aplicar un modelo de datos a un mundo real para obtener un esquema:



2.1.2. Estructuras

2.1.2.1. Abstracciones

Un sistema de gestión de bases de datos es una colección de archivos interrelacionados y un conjunto de programas que permiten a los usuarios acceder y modificar esos archivos. Un objetivo importante de un SDBDD es proporcionar a los usuarios una *visión abstracta* de los datos. Es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos.

Existen tres niveles de abstracción:

- Nivel físico. El nivel más bajo de abstracción describe cómo se almacena realmente los datos. En el nivel físico, se describen en detalle las estructuras de datos complejas del nivel bajo.
- Nivel conceptual. El siguiente nivel más alto de abstracción describe qué datos son realmente almacenados en la base de datos y las relaciones que existen entre los datos. Aquí se describen la base de datos completa en términos de un número pequeño de estructuras relativamente. Aunque la implementación de las estructuras sencillas del nivel conceptual puede implicar estructuras complejas del nivel físico, el usuario no necesita darse cuenta de esto. Este nivel es usado por los administradores de bases de datos, quienes deben decidir qué información se va a guardar en la base de datos.

- Nivel de visión. El nivel más alto de abstracción describe sólo parte de la base de datos completa. A pesar del uso de estructuras más sencillas en el nivel conceptual, permanece algo de complejidad debido al gran tamaño de la base de datos. Muchos usuarios del sistema de bases de datos no se interesarán por toda la información. En cambio, dichos usuarios sólo necesitan una parte de la base de datos. Para simplificar su interacción con el sistema, se define el nivel de abstracción de visión. El sistema puede proporcionar muchas visiones de la misma base de datos.

2.1.2.2. Conjuntos- Dominios y atributos

Un atributo es el calificador de la entidad. Es decir, las propiedades de una entidad están definidas por sus atributos.

El concepto de atributo es equivalente al de campo, o para entenderlo mejor al de columna de una tabla.

Para cada atributo hay un conjunto de valores permitidos llamado dominio de ese atributo.

Ejemplo: El Dominio del atributo Nombre >>Cliente podría ser el conjunto de todas las cadenas de texto de una determinada longitud. El dominio del atributo Saldo >>>Cta.Cte. Podría ser el conjunto de todos los enteros positivos.

Formalmente, un atributo es una función que asigna un conjunto de entidades a un dominio. Así cada entidad se describe por medio de un conjunto de pares.

Dominio: (Atributo, valor del dato) Un par para cada atributo del conjunto de entidades. Una entidad cliente determinada se describe por medio del conjunto.

Ejemplo:

(calle, Alameda Nº 271)
(Ciudad, Santiago)
(Rut,101224-6)
(Nombre, Carlos)

Todo esto significa que la Entidad describe a una persona llamada Carlos con Rut 1012214-6, que vive en la ciudad de Santiago en la calle Alameda Nº 271.

Los dominios pueden ser:

- Dominios simples: Sea un escalar como la menor unidad semántica de información, es decir, un dato atómico, se define dominio, o dominio simple, como un conjunto de escalares. Es aconsejable que el nombre de los dominios y de los atributos coincidan, siempre y cuando no exista ambigüedad.
- Dominios compuestos: Es una combinación de dominios simples.

Podemos interpretar los dominios, desde un punto de vista intuitivo, como un tipo de datos. De este modo, se definen atributos (variables) que pertenezcan a un dominio (tipo de datos).

2.1.2.3. Relaciones- Entidades e Interrelaciones

Se define por entidad a cualquier objeto distinguible que pueda representarse en una Base de Datos y entendemos por relación a la correspondencia entre esas entidades en base a un atributo o atributos. Generalmente, estas son bidireccionales y se clasifican de 1 a 1, de 1 a N, de N a N...

Una relación sobre un conjunto de dominios se compone de dos partes: la cabecera y el cuerpo. La cabecera, o esquema de la relación, se compone de un conjunto fijo de pares atributo-dominio. El cuerpo, o extensión de la relación, está formado por un conjunto de tuplas.

Un conjunto de entidades es un conjunto de entidades del mismo tipo. El conjunto de todas las personas que tienen una cuenta en un banco puede definirse como el conjunto de entidades clientes. También podríamos definir el conjunto de todas las cuentas de un banco como el conjunto de entidades de cuentas.

Los conjuntos de entidades no necesitan ser disjuntos Por ejemplo, es posible definir el conjunto de entidades de todos los clientes de banco, y el conjunto de entidades todos los empleados de un banco. Una persona puede ser una entidad empleado, una entidad cliente, ambas o ninguna de las dos.

2.1.2.4. Representación- Tablas y Grafos

En las tablas podemos interpretar:
- Existe un dominio subyacente asociado a cada columna de la tabla.
- Las dos partes de la tabla corresponden a las dos partes de la relación.
- Las filas se forman de pares atributo-valor.

Las propiedades de las tablas son:
- No existen tuplas repetidas. En una relación aparece un conjunto de tuplas, y por lo anterior, no puede haber dos tuplas iguales. En una tabla si pueden haber filas repetidas.
- Las tuplas no están ordenadas.
- Los atributos no están ordenados.
- Los valores de los atributos son atómicos. Una relación que cumple esta cuarta propiedad se dice que está normalizada.

2.1.3. Reglas de integridad

2.1.3.1. Claves primarias

- Definiciones:
  - Una Superclave es un conjunto de atributos que identifican de modo único las tuplas de una relación.
  - Una Clave Candidata es el menor subconjunto de atributos de una superclave que sigue siendo un identificador único.
  - En una relación pueden existir diferentes claves candidatas que se compongan de un número diferente de atributos.
  - De todas las claves candidatas se elige una que será la Clave Primaria.
  - El resto de claves candidatas se definen como Claves Alternativas.
- Propiedades: Por lo anterior, existen dos propiedades básicas asociadas a una clave candidata:
  - Unicidad: no existen dos tuplas que posean el mismo valor de la clave candidata.
  - Minimalidad: no se puede eliminar ningún atributo de la clave candidata sin destruir la unicidad de la clave candidata.

2.1.3.2. Regla de Integridad de Entidades

- Definiciones:
  - Ningún Componente de la Clave Primaria de una Relación Base puede aceptar Nulos.
  - En el modelo relacional Nulo se refiere a ausencia de valor.
  - En SQL dicha definición varía ligeramente.
- Justificación:
  - Las entidades se identifican de modo único en la realidad, y también deben de poderse identificar en el modelo relacional.
  - Dicha identificación es realizada por las claves primarias.
  - Si una clave primaria contiene un nulo quiere decir que no se puede aplicar la definición de clave primaria sobre la entidad asociada.
  - Por lo tanto, esta entidad no se puede identificar, lo que entre en contradicción con la definición de entidad.
- Así pues, la regla también se puede formular:

- En una base de datos relacional, no se puede almacenar información sobre algo que no se pueda identificar.
- Las claves primarias compuestas debe de ser no nulas en su totalidad, es decir, ninguno de sus atributos puede ser nulo.

2.1.3.3. Claves ajenas

- Definiciones:
  - Una Clave Ajena es un conjunto de atributos de una relación R2 cuyos valores o son completamente nulos o coinciden que los de la clave primaria de una relación R1.
  - De un modo gráfico se representa mediante una flecha entre las dos relaciones que se etiqueta por el conjunto de atributos que la definen:

    $R_2 \longrightarrow R_1$
  - La relación donde se encuentra la clave ajena se denomina Relación Referencial.
  - La Relación Referida u Objetivo indica la relación donde se encuentra la clave primaria.
- Comentarios:
  - Una relación referida puede ser también una relación referencial respecto de otro conjunto de atributos.
  - De este modo se puede definir una Ruta Referencial entre diferentes relaciones de la base de datos.
  - Si alguna de las relaciones aparece más de una vez en la ruta referencial se dice que aparecen Ciclos Referenciales.
  - La Relación Autorreferencial es un caso particular del anterior, donde coinciden la relación referida y la relación referencial.
  - Las claves ajenas pueden admitir nulos, a diferencia de las claves primarias.

2.1.3.4. Regla de integridad referencial

- La Base de Datos no debe contener valores de clave ajena sin concordancia.
- Es decir, para cualquier valor no nulo de la clave ajena existe un valor asociado en la clave primaria de la relación objetivo.

2.1.3.5. Reglas para las claves ajenas

- Borrado de Claves Primarias: La decisión depende de los requerimientos, existiendo tres posibilidades:
  - Restringida (Restricted). No se puede borrar una tupla si está referenciada.
  - Se Propaga (Cascades). Se borra la tupla junto con las tuplas que la referencian.
  - Anula (Nillifies). Se borra la tupla y se pone a nulo la clave ajena de las tuplas que la referencian, siempre y cuando los acepte.
- Modificación de Claves Primarias: La decisión depende de los requerimientos, existiendo tres posibilidades:
  - Restringida (Restricted). Una tupla no se puede modificar si está referenciada.

- Se Propaga (Cascades). Se modifica la tupla junto con las tuplas que la referencian.
- Anula (Nullifies). Se modifica la tupla y se pone a nulo la clave ajena de las tuplas que la referencian, siempre y cuando los acepte.

2.1.4. Estática: Estructuras permitidas y no permitidas

Es una de las propiedades que presenta el Universo de Discurso. Se representa por la letra 'S'y trata la parte relativamente invariante en el tiempo, es decir, las estructuras. Se corresponden con el Lenguaje de Definición de Datos (LDD) y sus estructuras pueden ser:

- Estructuras permitidas:
  - Entidades. Las entidades son objetos, es decir, cualquier cosa con existencia propia o independiente como puedan ser persona, animal, concepto o suceso. Pueden ser simples (irreducibles) y compuestos. Tienen atributos.
  - Atributos (propiedades de las entidades). Los atributos son las propiedades de un objeto (cualquier información que me interese sobre ese objeto).
  - Dominios (rango de definición de los atributos). Rango de valores posibles que un atributo puede aceptar.
  - Interrelaciones (asociaciones entre objetos). Al igual que las entidades también pueden poseer atributos.
- Estructuras no permitidas (restricciones):
  - Inherentes. Impuestas por la naturaleza del modelo, haciendo que la forma de modelar sea rígida.
  - De usuario. Captan la semántica del Universo de Discurso, es decir, la visiòn del diseñador:
    . Reconocidas por el propio SGBD.
    . Desconocidas por el SGBD. Responsabilidad del usuario.

Las estructuras no permitidas no inherentes, es decir, las restricciones de usuario, se representan por 'S_r', y al resto de estructuras, es decir , a las estructuras permitidas y a las no permitidas inherentes, se las designa por 'S_e', dando a la propiedad estática la siguiente representación: $S=<S_e,S_r>$.

2.1.5. Dinámica

Es otra de las propiedades del Universo de Discurso. Se representa por la letra 'O' y tiene la característica de variar en el transcurso del tiempo (datos o valores). Se corresponde con el Lenguaje de Manipulación de Datos (LMD).

Siendo t_i (BD_i) los valores de los objetos de un esquema en un determinado momento, es decir, ocurrencia del esquema o BD`s en el tiempo $t_i$, $t_j$ será la ocurrencia del esquema en otro determinado momento. Esto se produce al realizar operaciones como las de cambiar algún valor en la BD`s o cambiar algún indicador.

Las operaciones admitidas sobre la estructura del correspondiente modelo de datos hacen transformar la ocurrencia de un esquema en otra ocurrencia distinta: $O(BD_i)=BD_j$. Estas operaciones pueden ser de tres tipos:

- Selección:
  - Localizar una ocurrencia de una entidad indicando un camino (navegacional).
  - Localizar varias ocurrencias de una entidad especificando condición (de especificación).
- Acción: Tras la selección podemos realizar una recuperación o una actualización (alta, modificación o borrado.

- Transacción: Agrupamiento lógico de operaciones simples, con la peculiaridad de que actúan como un todo indivisible, es decir, la transacción o se realiza entera o no se realiza.

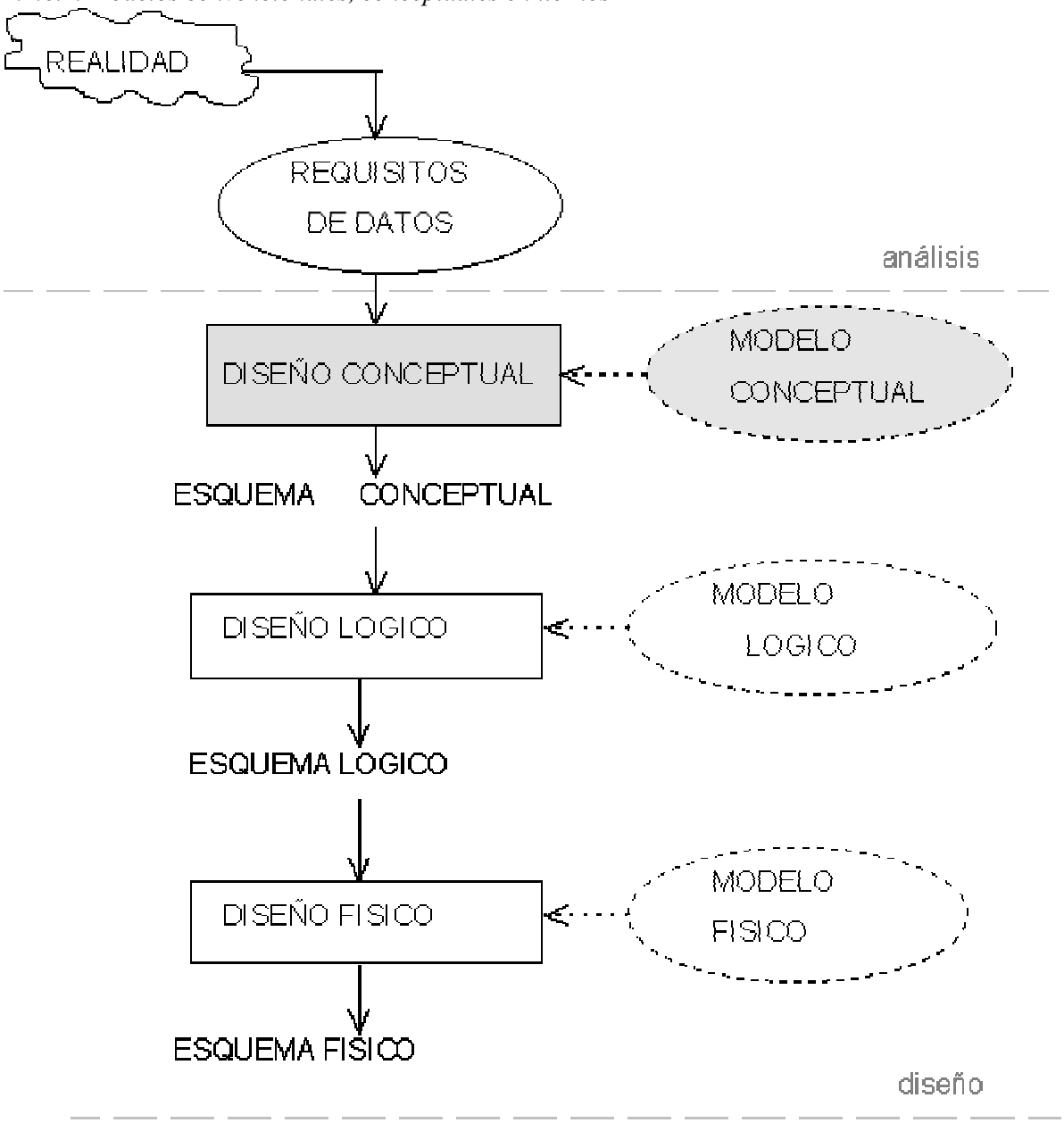
Las Bases de Datos, en cuanto a operaciones se refiere, van a evolucionar en el tiempo de forma discreta, pasando por una serie de estados, todos ellos correctos. Es decir, si la Base de Datos estaba en un estado correcto antes de una operación, lo seguirá estando después de ella.
Entre todas estas operaciones existe una relación (en el orden de ejecución), la cual recibe el nombre de estructura de control.

Al igual que en las estáticas, en las dinámicas también existen una serie de restricciones que nos avisan de los cambios permitidos o no permitidos. Así por ejemplo, una persona no puede poseer una tarjeta de crédito si antes no se ha hecho cliente de la entidad bancaria. Para modelizar todas estas restricciones utilizamos las condiciones iniciales (precondiciones) y las condiciones finales (poscondiciones).

La dinámica no aparece en el resultado (esquema) de la BD's; ni en el modelo relacional, ni en el modelo entidad-interrelación.

2.1.6. Clasificación de los modelos de datos

2.1.6.1. Modelos convencionales, conceptuales e internos



Con la aparición de los niveles de abstracción de la arquitectura ANSI, distinguimos tres tipos de modelos:

- Modelos lógicos externos: Se corresponderían con los modelos clásicos y convencionales. Están basados en registros y buscan la eficiencia humana construyendo esquemas externos o de usuario. Existe poco nivel de abstracción y una interfaz entre el usuario y el ordenador/sistema. Han sido y son los más utilizados. Se utiliza el concepto de BD's, y su evolución en el tiempo nos lleva hasta tres modelos:
  - Modelo jerárquico.
  - Modelo en red.
  - Modelo relacional.
- Modelos lógicos conceptuales (globales): Se conrresponderían con los modelos semánticos. Están basados en objetos y buscan la eficiencia de los recursos de información

de la organización construyendo esquemas lógicos globales. Con estos modelos se intenta almacenar información sobre los datos que poseemos, se insiste en la necesidad de incorporar la semántica. Existe un mayor nivel de abstracción, mayor interfaz usuario-informático, y correspondería a las primeras fases del diseño. Estos modelos son:

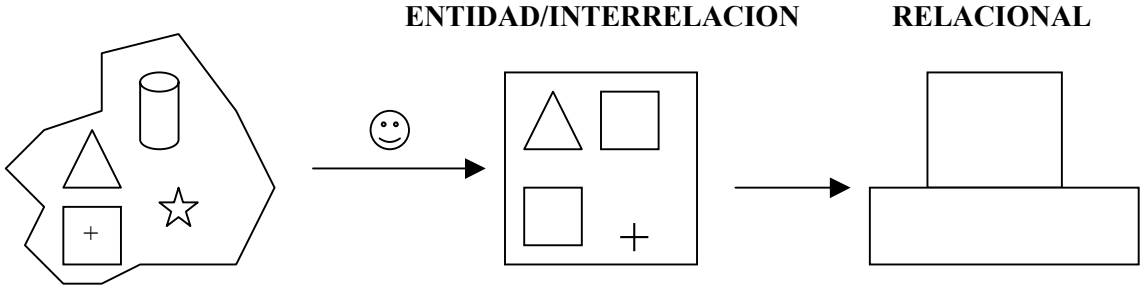
- Modelo entidad-relación.
- Modelo orientado a objetos.
- Modelo binario.
- Modelo semántico de datos.
- Modelo infológico.
- Modelo funcional de datos.

- Modelos internos: Se corresponderían con los modelos físicos de datos y buscan la eficiencia de los recursos informáticos construyendo el esquema físico o interno. Son muy pobres en cuanto a abstracción se refiere, y además están demasiado basados en la máquina. Describen los datos en el nivel más bajo y permiten identificar algunos detalles de implantación para el manejo del hardware de almacenamiento. Capturan aspectos de la realización final y quedan muy pocos modelos de este tipo. Para el diseño físico se recibe como entrada el esquema lógico y da como resultado un esquema físico, que es una descripción de la implementación de una base de datos en la memoria secundaria, describe las estructuras de almacenamiento y los métodos usados para tener un acceso efectivo a los datos. No vamos a profundizar en estos modelos, ya que no ha sido posible encontrar el material suficiente, y tampoco son de vital importancia en nuestro estudio sobre modelos de datos. Dos de estos modelos son:
  - Modelo unificador.
  - Memoria de elementos.

2.1.7. Modelos conceptuales

2.1.7.1. Modelos de datos conceptuales frente a convencionales

No van a ser los modelos convencionales mejores que los conceptuales, ni los modelos conceptuales mejores que los convencionales; simplemente serán diferentes. Está diferencia está marcada por la sencilla razón de que los convencionales van a estar más próximos a la máquina, es decir, serán más fáciles de implementar, y los conceptuales van a estar más distantes de la máquina, es decir, serán más difíciles de implementar. Y siendo así, ¿por qué no dejamos de pensar en los conceptuales? Pues, porque al mismo tiempo, estos disponen de una ventaja que se convierte en desventaja en el convencional, y esta es la capacidad semántica, o lo que es lo mismo, la gran perceptibilidad de la realidad. Así por ejemplo, yo no podré implementar un conjunto en la computadora, pero si podré implementar una estructura de datos que simule a ese conjunto.



En un principio, eran considerados como modelos convencionales aquellos modelos instrumentados en SGBD, pero ya se han llegado a instrumentar modelos conceptuales como el de Entidad/Interrelación o el semántico de datos en distintos prototipos, proyectos y herramientas CASE

y empiezan a formar parte de algunos SGBD, por lo que sirven, al igual que los convencionales, de conexión entre los esquemas externos e interno.

Los modelos conceptuales presentan una mayor flexibilidad y una mayor capacidad semántica como consecuencia también de su mayor nivel de abstracción. Por ello, a la vez pueden constituir una interfaz útil entre el informático y los usuarios finales frente a la interfaz entre el informático y el sistema de ordenador que únicamente puede constituir el modelo convencional.

2.1.7.2. Propiedades

Los modelos de datos han de ser:

- Expresivos. Que pueda representar el mayor número de conceptos posibles.
- Simples. Cuanto mayor sea la simplicidad, mejor entenderemos los esquemas.
- Mínimo. El número de conceptos que maneje será el mínimo y necesario, es decir, habrá una sola forma de representación para un mismo concepto.
- Formal. Que no haya ambigüedad, es decir, que la interpretación sea única, precisa y esté muy bien definida.
- Representativo. Ha de existir un sistema de representación gráfica, el cual debe cumplir:
  - Completitud. Que no quede gráficamente sin representar ningún concepto que tenga en mi modelo.
  - Legibilidad. Utilizar símbolos gráficos que hagan fácil la lectura.

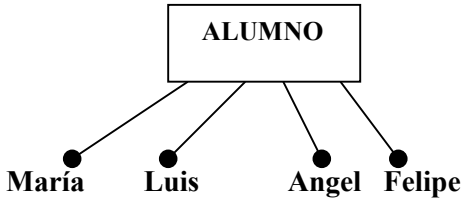
2.1.8. Mecanismos de abstracción

La abstracción consiste en fijarnos solamente en la parte de la realidad que a nosotros nos interesa, obviando propiedades, que no son interesantes, de algo demasiado complejo para nuestro entender. Existen tres mecanismos básicos que van a ser independientes, es decir, no podremos llegar a ninguno de ellos utilizando los dos restantes. Habría otros mecanismos de abstracción, pero ya serían derivados de estos tres.

2.1.8.1. Clasificación

Es el mecanismo opuesto a la instanciación. Tengo el nombre de una clase y una colección de objetos (todos ellos con las mismas propiedades: las de la clase). Se establece una relación "instancia-de" entre los objetos individuales (la colección de objetos) y el objeto general (la propia clase). Además, un mismo objeto individual, puede ser "instancia-de" varios objetos clase.

Ejemplo:



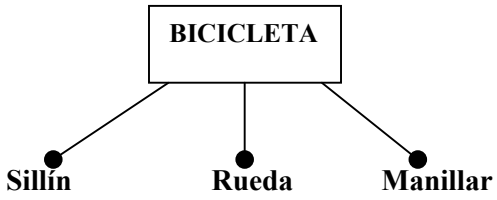
Su objetivo es clasificar los objetos de la realidad caracterizados por propiedades comunes. Ej.: MES= Enero, Febrero,..., Diciembre.

Estas abstracciones se representan mediante un árbol, cuya raíz es la clase y las hojas son los miembros de la clase.

2.1.8.2. Agregación

Tengo una colección de objetos llamados componentes y otro objeto llamado compuesto o agregado. Se establece una relación "parte-de" entre esa colección de objetos y el objeto compuesto.

Ejemplo:

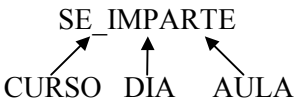


Una asociación normal entre dos clases representa una relación estructural entre iguales, es decir, ambas clases están conceptualmente en el mismo nivel, sin ser ninguna más importante que la otra. Así, la agregación surge de la necesidad de representar una relación en la que un objeto del todo tenga objetos de la parte.

En realidad, la agregación es sólo un tipo especial de asociación.

Se definen nuevas clases a partir del conjunto de clases asociadas a las partes que la componen.

Ej.:

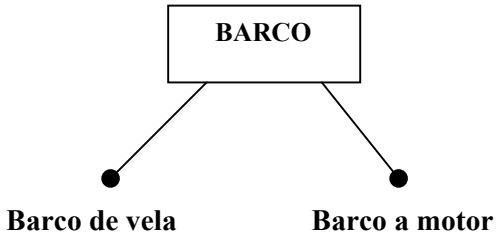


Las clases hoja son parte de la clase-raíz, y también podemos decir que mediante la clasificación se identifican tipos de atributos y con la agregación se identifican tipos de entidades.

2.1.8.3. Generalización

Es el mecanismo opuesto a la especialización. Tengo una colección de objetos llamados subclases o categorías (hijos) y otro objeto genérico (superclase o padre). Se establece una relación "es-un" entre esa colección de objetos y el objeto genérico.

Ejemplo:



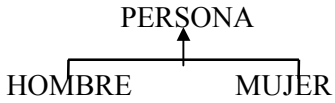
La generalización significa que los objetos hijos se pueden emplear en cualquier lugar que pueda aparecer el padre, pero no a la inversa. En otras palabras, la generalización significa que el hijo puede sustituir al padre. Una clase hija hereda propiedades de sus clases padres, especialmente sus atributos y operaciones. A menudo (no siempre) el hijo añade atributos y operaciones a los que hereda de sus padres. Una operación de un hijo con la misma signatura que una operación del padre redefine la operación del padre; esto se conoce como polimorfismo.

Una clase puede tener ninguno, uno o más padres. Una clase sin padres y uno o más hijos se denomina clase raíz o clase base. Una clase sin hijos se llama clase hoja. Una clase con un único padre se dice que utiliza herencia simple; una clase con más de un padre se dice que utiliza herencia múltiple.

En el modelado son muy frecuentes las generalizaciones entre clases e interfaces para reflejar relaciones de herencia.

Permite definir una relación de subconjunto entre dos o más clases.

Ej.:

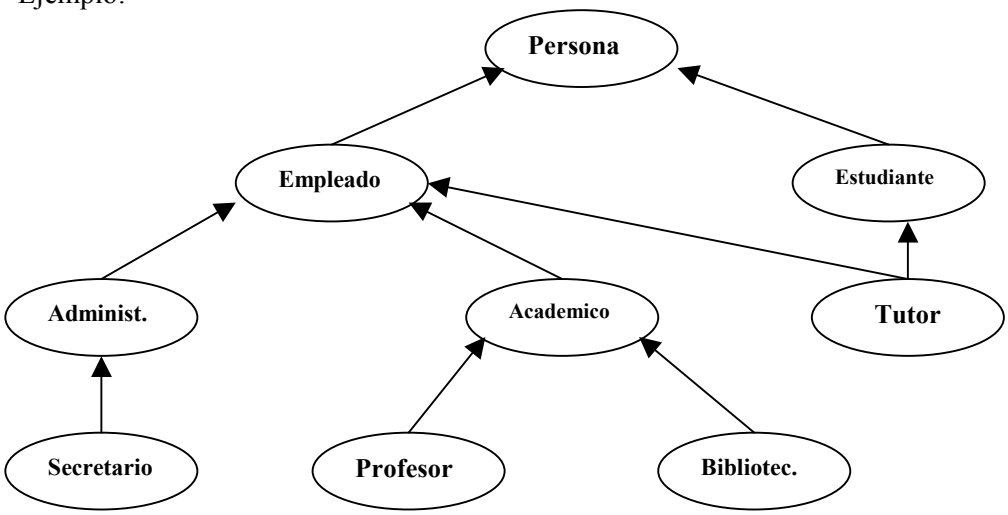


En una generalización, todas las abstracciones definidas por la clase raíz son heredadas por las clases subconjunto.

En resumen, las generalizaciones serán utilizadas para mostrar relaciones padre/hijo.

2.1.8.4. Jerarquías de abstracciones.Jerarquía de generalización

- Se representa mediante flechas que parten de las entidades E_i y llegan a la entidad E.
- Las generalizaciones se caracterizan por las propiedades anteriormente comentadas,
  - . Pueden ser Totales o Parciales.
  - . Pueden ser Exclusivas o Superpuestas.
- Las propiedades de la entidad genérica son heredadas por las entidades subconjunto.
  Ejemplo:



2.1.9. Operaciones

Una operación es una acción elemental e indivisible que permite consultar o modificar el estado actual de una Base de Datos. No todos los modelos permiten las mismas operaciones, pero hay algunas que todos los modelos permiten:

- Selección: Localizar un conjunto de ocurrencias de objetos especificando unas condiciones.
- Acción: Se realiza sobre las ocurrencias de objetos localizados por una operación de selección.
- Transacción: Es un agrupamiento lógico de operaciones simples, pero lo más importante es que actúan como un todo indivisible. La transacción o se hace entera o no se hace.

2.2. ESTANDARES EXISTENTES

Se buscaba un modelo con mayor número de ventajas, pero sobre todo se buscaba la estandarización, para así poder satisfacer las necesidades de todos los usuarios.

Se intenta estandarizar para así conseguir que una vez desarrollado un sistema e instrumentado en un determinado SGBD, el cambio de éste producto comercial no implique tener que diseñar de nuevo la base de datos, ni tampoco programas que acceden a la misma tengan que ser reescritos. También se pretende conseguir la independencia frente a los proveedores, es decir la posibilidad de que cualquier suministrador pueda resolver nuestro problema en nuestro SGBD. Con todos estos elementos llegaríamos a conseguir algo tan importante como es la adquisición de sistemas abiertos.

En resumen, el objetivo de la estandarización es proteger las inversiones y defender la independencia del usuario frente a los suministradores de SGBD. Por esta razón, los estándares se concretan en especificaciones de cara al usuario, es decir, en la interfaz del sistema con el entorno.

Este tema que estamos tratando de estandarización se convierte en un caso controvertido debido a los pros y los contras de dicho proceso. Como ventaja, tenemos que una normalización a posteriori tendría una incidencia favorable en el desarrollo de bases de datos al no introducir cortapisas que dificulten el avance en distintas direcciones, pero a la misma vez como inconveniente tenemos que será prácticamente imposible imponer unas normas a sistemas que han sido ya desarrollados y que se encuentran en el mercado. Como caso contrario a éste, vemos como ventaja que una estandarización prematura orientará a los diseñadores y será más fácil de aplicar, pero al mismo tiempo como inconveniente tenemos que probablemente no dejará que surjan nuevas ideas y será un freno a la imaginación de los creadores de los SGBD.

Desde principios de los setenta han sido varios los grupos informáticos que se han ocupado de este tema. De entre todos ellos cabe destacar el Grupo Guide/Share de usuarios de IBM, el Club de Banco de Datos del INRIA (Institut National de Recherche en Informatique et Automatique),el comité nacional de estandarización canadiense (GESC), el comité nacional de estandarización británico (BSI),etc. Pero no nos ocuparemos de todos ellos y solamente veremos un poco las actividades de tres de ellos:

- Actividades de ISO/IEC. Estos han creado un comité conjunto para las Tecnologías de la Información(JTC1), el cual a su vez contiene a otro comité dedicado a los sistemas abiertos(SC 21). Dentro de este último se organizan grupos de trabajo entre los que se encuentra el dedicado a las bases de datos(WG 3), y que se dedica a cuatro proyectos principales como son el de los lenguajes de Bases de Datos (lenguajes para Bases de Datos en red y SQL), los modelos de referencia (se provee de una base común para el desarrollo de estándares proporcionando un marco conceptual para los sistemas de bases de datos, identificando funciones, procesos e interfaces), el acceso remoto a datos (estudia el comportamiento de un "servidor de BD" que proporciona las facilidades de almacenamiento de datos y los serviciosde procesamiento de BD a sistemas "clientes", y los servicios y protocolos de comunicación para que los clientes puedan acceder a los servidores), y los sistemas de diccionarios de recursos de información (se ha aprobado ya un amplio conjunto de estándares, entre los que destacan el marco conceptual y la interfaz de servicios).
- Actividades de Codasyl. Sus especificaciones definen y detallan los lenguajes de descripción y manipulación de los SGBD, y aunque han sido aplicadas en mayor o menor medida a diversos SGBD comerciales, no pueden considerarse un verdadero estándar al no ser Codasyl un grupo oficial de estandarización.
- Actividades del grupo ANSI/X3/SPARC. Este comité empieza a desempeñar sus tareas en 1972, y siendo ya conscientes de que una normalización a destiempo podría fácilmente constituir un freno para los avances tecnológicos, deciden establecer una serie de aspectos de los SGBD como candidatos a una estandarización. Así, se produjo una serie de informes parciales, hasta que en 1975 se publicó el informe provisional (interim report), ampliamente difundido y discutido, en el que se presentaba una arquitectura de SGBD que incluía la identificación y descripción de sus múltiples interfaces. La finalidad de este informe era presentar un marco para el análisis y refinamiento de dicha arquitectura y de

sus interfaces. En el año 1977 se obtuvo el informe final, en el que se detallaba el análisis de la arquitectura y de algunas de las 42 interfaces que habían sido identificadas. A diferencia de las primitivas especificaciones del grupo Codasyl, que establecían solamente dos estructuras; lógica y física, el grupo ANSI/X3/SPARC introduce en la arquitectura de bases de datos un tercer nivel, el conceptual, que se interpone entre las dos estructuras anteriores ayudando a conseguir el objetivo de independencia. Posteriormente, en el año 1978, el grupo Codasyl, presenta un nuevo informe con la arquitectura a tres niveles.

2.3. SITUACION DE MERCADO

En virtud de su poder, las bases de datos se han afianzado en dos áreas clave para el desarrollo de los negocios: el procesamiento de transacciones y el DataWareHousing. Gracias a la simplificación de los procesos y a la evolución de los ordenadores personales, las bases de datos cubren también las necesidades domésticas de proceso de información, así como las necesidades de gestión de Pymes.

Los programadores no pueden ser ajenos a los nuevos modelos de diseño de bases de datos, tanto en soluciones de escritorio como en los grandes servidores.

Es difícil que en una aplicación no se utilice una base de datos. Incluso en aquellas aplicaciones que no están destinadas al mundo empresarial, la base de datos es algo fundamental, pues siempre hay algo que almacenar. Evidentemente, dada la cantidad de situaciones distintas en las que se puede utilizar una base de datos, es lógico que existan múltiples tipos, así como muchas formas distintas de acceder a ellas.

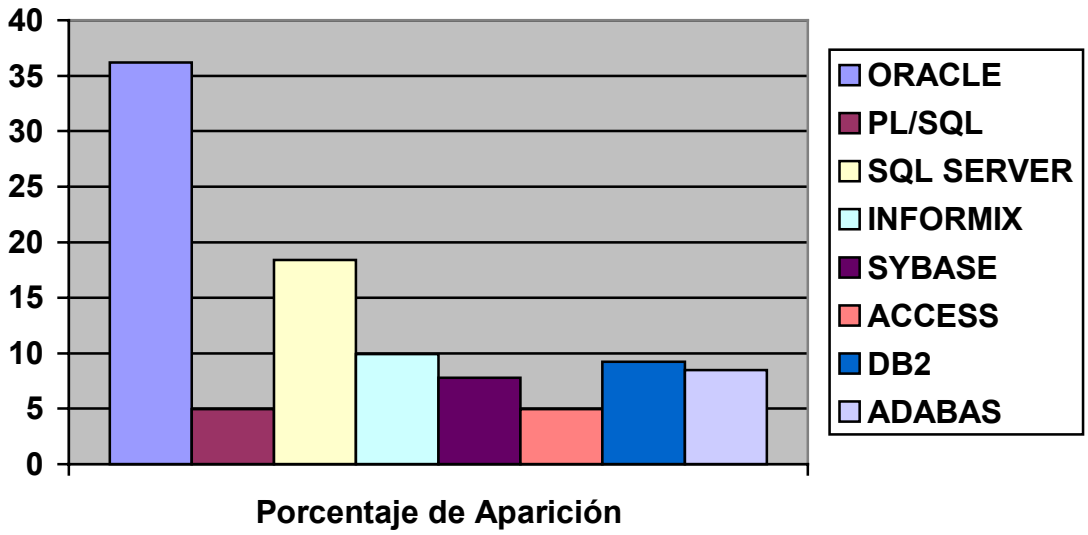
Las bases de datos actuales son todas relacionales, con un conjunto de tablas que se encuentra dividido en filas y columnas. Para acceder a las bases de datos relacionales, IBM desarrollo el lenguaje SQL. Pese a que la gran mayoría de las bases de datos fueron implementando este lenguaje, la forma de controlarlo desde un programa era diferente en función del fabricante. Para evitar estos problemas se desarrollaron controladores de acceso, siendo uno de los más famosos el ODBC, que proporcionan al programador un conjunto de funciones (API) estándar para acceder al motor. Cada fabricante suele desarrollar junto a la base de datos el controlador correspondiente, de igual forma que cada fabricante de impresoras fabrica el controlador de la misma. Con esto se consiguió que los programadores pudiéramos acceder de forma estándar a cualquier base de datos que incorporase el correspondiente controlador. A este tipo de software intermedio se le suele denominar *middleware* (palabra utilizada siempre que existe software intermedio). Sin embargo, con el fin de facilitar aún más las cosas, algunos fabricantes de herramientas de desarrollo dotaron a éstas con una nueva capa intermedia de software (motor de base de datos) intercalada entre el código fuente del programa y el controlador ODBC, o directamente entre el programa y la base de datos. En la actualidad los dos motores de base de datos más comunes son Microsoft Jet y BDE (Borland Database Engine).

Con la introducción de herramientas de desarrollo rápido (RAD) han proliferado los controles y los objetos de acceso a datos. Estos establecen un puente entre la interfaz del usuario y el motor de acceso a datos, por lo que la tarea del programador, tal y como se ve, queda reducida a la mínima expresión. Actualmente nos encontramos en esta etapa, donde los controles y los objetos de acceso a datos gobiernan las nuevas aplicaciones.

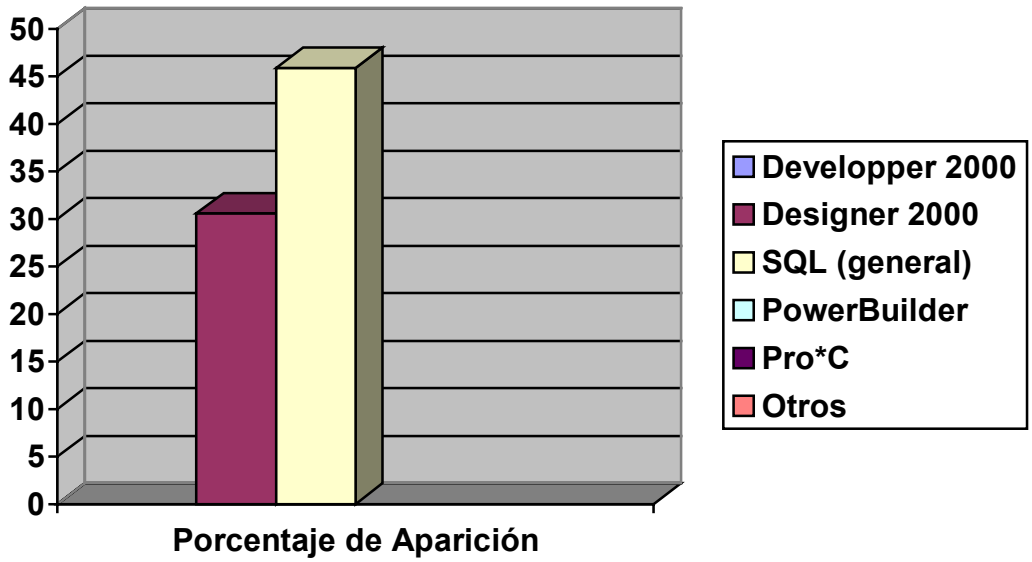
Basándonos en las ofertas de empleo que comúnmente podemos encontrar en las bolsas que aparecen en diarios de prensa, hemos elaborado un recuento (aislando las que pertenecen al sector informático) para obtener una primera estadística de la distribución de los diferentes S.G.BB.DD. en las empresas españolas.

Como conclusión, podemos decir que, como para los restantes tipos de Software, el mercado de los Sistemas Gestores de Bases de Datos está muy diversificado. Existen infinidad de ellos. Como siempre la elección, en el caso de encontrarse ante la necesidad de adquirir uno de ellos, irá en función de las necesidades que tengamos. Siempre será muy importante evaluar éstas necesidades pues el abanico de precios que se nos despliega ante nosotros, no incita a tomarse el asunto a la ligera.

Sistemas Gestores de Bases de Datos



Entornos de Desarrollo y Utilidades



CAPITULO 3:MODELOS LOGICOS BASADOS EN OBJETOS- MODELOS SEMANTICOS

Son aquellos que nos permiten una definición clara y concisa de los esquemas conceptual y de visión. Su característica principal es que permiten definir en forma detallada las fronteras de los datos. Para el diseño conceptual se recibe como entrada la especificación de requerimientos y su resultado es el esquema conceptual de la base de datos, que es una descripción de alto nivel de la estructura de la base de datos, independiente del software que se use para manipularla.

3.1. MODELO ENTIDAD-RELACION

Este modelo está basado en dos conceptos:
- Entidad
- Relación

Idea intuitiva:
- Entidades.- Objetos sobre los cuales queremos guardar información y cuya característica es que tienen existencia por si mismos.
- Relaciones.- Asociaciones entre entidades.

3.1.1 Estática del MER

Elementos:
- Entidades. Objeto real o abstracto del que al menos queremos guardar información en la BD. Sus características son:
  - Que tenga existencia propia.
  - Que se pueda distinguir de los demás elementos del UD.
  - Que todas las ocurrencias tengan los mismos atributos.

  Tenemos dos clases:
  - Fuertes: Tienen existencia por sí mismas.

    Representación gráfica (colocar)

  - Débiles: Tienen existencia gracias a otra entidad.

    Representación gráfica (colocar)

- Relaciones. Correspondencia entre entidades (matemáticamente existe relación).

Representaciones

Etiqueta (Nombre de relación).
Arcos del rombo a las entidades relacionadas.

Entre dos entidades pueden existir más de un tipo de relación:

- Relaciones reflexivas. Solo interesan características de personas sin hacer distinción.
- Relaciones entre más de dos entidades. {Tener cuidado con relaciones engañosas}. Pueden ser falsas relaciones. (Puede haber parte de la relación que no se cumpla semánticamente). El criterio para saber si son falsas es que emite SEMANTICA ENGAÑOSA .

  Ej.: Son falsas respecto de la realidad "Los libros son publicados por los autores y editoriales".

  Relación semánticamente correcta. Existirían relaciones que darían problemas al descomponer porque se pierde la relación directa. La determinación de si una relación es falsa o no es cuestión del diseñador.

  En el ejemplo se quiere guardar información precisa de en que libro concreto, un autor concreto escribe sobre un tema concreto (Seguir un criterio claro indicado en el enunciado).

Los elementos de las relaciones son:

- Nombre.
- Grado .- Número de tipos de entidad que participan en una relación.

  Ejs.- Grado 1.- Reflexivas.
  Grado 2.
  Grado N> 2 Falsas / Ciertas.

Los tipos de correspondencia o de relación serán:

- Def.- Es el nº máximo de ocurrencias de cada tipo de entidad que pueden intervenir en una ocurrencia del tipo de relación.
- Representación.- Se etiqueta con:

  1 : 1
  1 : N
  N : M

Las relaciones son bidireccionales y cada dirección debe tener un nombre. De tal forma que definimos papel como la función que cada tipo de entidad realiza en la relación.

- Valores y dominios. Son las características de las entidades y relaciones.

  Atributos.- Propiedades caracteristicas que tiene un tipo de entidad o relación.
  Valores.- Contenido concreto de los atributos.
  Dominios.- Conjunto de las posibles valores de los atributos.
  Representación gráfica : Solo se representan gráficamente los atributos :

  - Un ovalo por atributo.
  - Un arco de atributo a la entidad o relación a la que pertenece.

  Ej : Decidir por la semántica qué atributos pertenecen a la entidad y cuales a la relación.
  Tendremos varios tipos de atributos:

  - AIP (Atrib. De identificación principal).- Atributos que identifican unívocamente cada ocurrencia de la entidad o relación. Tiene que ser mínimo, de modo que si quito una parte ya no es AIP. Además, no son únicos.
  - AIC (Atrib de Identificación candidato). Para tipos de entidad todos los posibles de AIP.
  - AIA (Atrib de indentficación Alternativos). Los AIC no AIP.

  Observación AIP de relaciones.- Habitualmente son la concatenación de los AIP de las entidades que intervienen.
  Dependiendo de la semántica del problema (Criterio de enunciado) se toman unos u otros atributos como AIP para identificar unívocamente las ocurrencias de la relación.
  El único objetivo de guardar información de entidades es la de identificar unívocamente la relación entre determinadas ocurrencias.

- Restricciones.
  - Inherentes .- No hay.
  - De usuario.

  Las restricciones no tienen una representación gráfica precisa.

3.1.2 Modelo E/R Extendido.

Observaciones.- Completa el modelo E/R básico en algunos aspectos que trataba mal.
Extensiones:

- Cardinalidades. Nº máximo y mínimo de un tipo de entidad que pueden estar interrelacionados con una ocurrencia del otro u otros tipos que participan en el tipo de relación.
  Representación.- (Cardinal min, Cardinal max) .- en cada arco que va a una entidad.
  Valores.- (0,1), (1,1), 0,N), (1,N)
  Para rellenar un cardinal se ha de mirar desde el otro extremo de la relación.
- Dependencias. Al igual que para los tipos de entidad, los tipos de relación pueden ser regulares y débiles, según se asocien a dos entidades fuertes (regulares) o una fuerte y una débil (Débil), respectivamente.
  Los tipos de relación regular o fuerte no tienen ningún tipo de dependencia.
  En los tipos de interrelación débil se distinguen dos tipos de dependencia:
  - En existencia: Cuando las ocurrencias de un tipo de entidad débil no pueden existir si desaparece la ocurrencia de la entidad fuerte de la que dependen.
  - En identificación: Cuando, además de ser una dependencia en existencia, las ocurrencias de la entidad débil no pueden identificarse únicamente mediante los atributos propios de la misma, y hay que añadir el AI (AIP) del tipo de entidad regular de la que dependen.
- Relaciones exclusivas. Dos o más tipos de relaciones son exclusivas cuando cada ocurrencia de un tipo de entidad sólo puede pertenecer a un tipo de relación. No se puede utilizar el modelo básico de representación. O se produce una relación o se produce otra, pero nunca ambas a la vez.
- Generación y herencia:
  - La generalización responde a la necesidad de descomposición de tipos de entidad en varios subtipos. La interrelación que se establece entre un supertipo y sus subtipos corresponde al concepto "ES UN TIPO DE". Esta relación tiene la característica de que toda ocurrencia del subtipo es una ocurrencia del supertipo, pero no al contrario, de modo que las cardinalidades serán siempre (1,1) en el caso del supertipo y (0,1) en el casos de los subtipos. Podemos distinguir entre dos tipos distintos de solapamiento atendiendo a los siguientes criterios:

    . Solapamiento.- Cuando una ocurrencia del supertipo puede pertenecer a más de un subtipo (No Exclusividad). Exclusividad = No solapamiento. Su representación gráfica es similar a la de relación exclusiva.

    . Totalidad.- Si toda ocurrencia del supertipo tiene que pertenecer a algun subtipo. Representación → Añade un circulo al arco Supertipo → Triangulo.

    Así, podremos llegar a encontrarnos con generalización total sin solapamiento, generalización parcial sin solapamiento, generalización total con solapamiento, y generalización parcial con solapamiento.
  - La herencia es otra característica muy importante de este tipo de relaciones. Todo atributo del supertipo pasa a ser atributo de los subtipos. Los atributos comunes a todos los subtipos se asignan al supertipo, mientras que los atributos especificos se asignan al subtipo correspondiente. Lo mismo ocurre con las interrelaciones. EJ.- Las personas se dividen en decentes e indecentes. Interesa guardar Nombre, Apellido y DNI (único).
    Así, para el supertipo tendremos los atributos comunes a todos los subtipos, es decir las relaciones con todos los subtipos se establecen con el supertipo.

Se añade una semántica, con lo que las personas tienen una serie de derechos humanos, las personas indecentes se relacionan con la comisarías (fichan), y las personas decentes se relacionan con las empresas (contratos).
De igual manera que para el supertipo, para el subtipo tendremos los atributos y relaciones que pertenecen o afectan al subtipo.
Los atributos discriminantes serán aquellos atributos en función de valores que deciden la condición de 'Ser un ' subtipo u otro.
Representación .- Atributo de la relación 'Es un '.

- Dimensión temporal: El estudio del tiempo es poco frecuente en el modelo E/R.
  - Primera aproximación:
    . Atributo tipo fecha.- ** Cambia el cálculo de cardinalidades.
    . No es semántica → Establece limitación temporal a relación.
  - Otras aproximaciones:
    . Bases de datos históricas.
    . Incorporar tiempo como entidad.
    . Estado.

Los atributos derivados son aquellos que se calculan a partir de otros. En el modelo E/R se recomienda su eliminación.

Además de la existencia de atributos redundantes se ha de estudiar en los ciclos, si pudieran existir relaciones redundantes. Se suele mirar si la relación que cierra el ciclo es redundante. Se determina que una relación es redundante si la semántica de la relación puede ser determinada a partir de otras. La relación redundante se puede mantener si nos obligan a almacenar información de la relación.

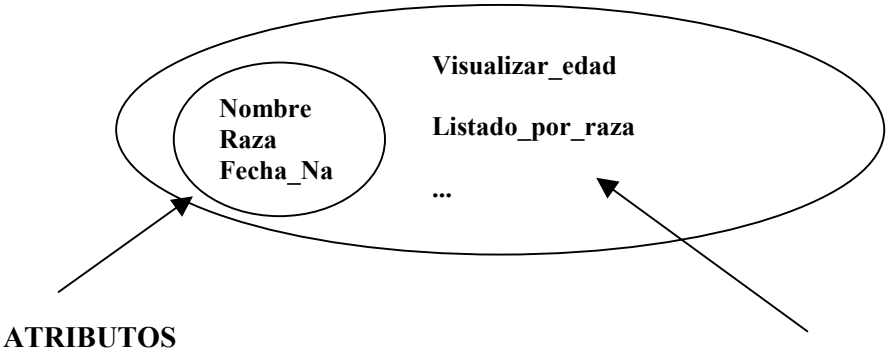
3.2. MODELO ORIENTADO A OBJETOS

Los datos se representan mediante objetos, que contienen variables y métodos, y su manipulación se realiza mediante mensajes. Al igual que el modelo E-R se basa en una colección de objetos. Un objeto contiene valores almacenados en instancias dentro del objeto. Estos valores son objetos por sí mismos, esto es, los objetos contienen objetos a un nivel de anidamiento de profundidad arbitraria.

Un objeto también contiene partes de código que operan sobre el objeto. Estas partes se llaman métodos.

Los objetos que contienen los mismos tipos de valores y los mismos métodos se agrupan en clases.
Una clase puede verse como definición de tipo para objetos.

Ejemplo:



Son los SGBO los que implementan el modelo orientado a objetos. Estos sistemas están basados en el análisis por separado de los datos y de los procesos.

Podemos definir un objeto como:

- A nivel conceptual: Una entidad del sistema que se está desarrollando.
- A nivel de implementación: Encapsulamiento de un conjunto de operaciones (servicios). Se trata de una forma de abstracción en la que se ven involucrados los datos y los procesos, ocultándose los aspectos de la implementación (se permite modificar los aspectos privados de un objeto sin que se vean afectados los demás objetos que interactúan con éste). Con todo esto, el usuario no ve la implementación .

Los objetos se comunican entre sí a través de mensajes y cada uno de ellos se distingue del resto por su estructura (propiedades o atributos) y por su comportamiento (servicios que proporciona). Según sean los valores de los atributos de un objeto, así será su estado.

Los atributos de los objetos toman valores que pueden ser inmutables, pueden no tener identificador (se identifica por sí mismo), y puede no soportar interrelaciones.

Un identificador de objeto es una característica especial y univoca que lo identifica, y un tipo de objeto es una clasificación por objetos que comparten la mismas características. Así, denominaremos clase a la implementación de un tipo de objeto.

Las interacciones entre los objetos pueden ser:

- Estáticas:
  - Destaca la generalización, en la cual los tipos y las clases se organizan en jerarquías o retículos de super/subtipos (super/subclases) que presentan herencia y se corresponden con el concepto “es-un”, donde los subtipos (o subclases) heredan los servicios o atributos de sus ancestros.
  - Y la agregación, donde se permiten construir objetos complejos o compuestos correspondientes a la noción “parte-de”.
- Dinámicas: Unos objetos solicitan la prestación de servicios a otros, los cuales si llega el caso, envían otro mensaje de respuesta.

Un aspecto a considerar, y que es importante en el modelado orientado a objetos es el polimorfismo (muchas formas), es decir, la capacidad de que un mensaje sea interpretado de formas distintas según el objeto que lo recibe. Encontramos dos formas de polimorfismo:

- De subclase: cuando un servicio definido en una clase se redefine en alguna de sus subclases manteniendo el mismo nombre. Entonces un mensaje enviado a un objeto que pertenece a una cierta clase de la jerarquía puede invocar cualquiera de estos servicios, según sea la clase a la que pertenezca el objetos que lo recibe.
- De sobrecarga: utilizando el mismo nombre para servicios distintos no situados en una jerarquía de generalización.

Para hacer del polimorfismo una característica eficiente es necesario que se produzca una vinculación entre el nombre de un servicio y su implementación en tiempo de ejecución (vinculación dinámica).

Para terminar, podemos decir que otra de las grandes ventajas de este modelo es la genericidad, pudiendo considerar a un tipo genérico como una plantilla que nos permite construir tipos.

3.3. MODELO BINARIO

Este modelo utiliza el concepto de relaciones binarias entre conjuntos de datos. El modelo binario de datos es importante en el desarrollo de los sistemas de bases de datos, ya que encontramos sus principios en todos los sistemas de bases de datos que utilizan el modelo de red. Es más; es posible transformar el modelo binario en un modelo de red. Los lenguajes de manipulación asociados a estos sistemas son bastante complejos en la medida en que ofrecen numerosas primitivas de acceso específicas de cada organización de datos.

Para dar una idea del modelo, haremos referencia al modelo Z de Abrial, el cual está dividido en dos partes. Una de ellas se corresponde con el nivel conceptual y denota clases de informaciones llamadas conjuntos de entidades, pudiéndose definir entre dos de estos conjuntos una relación binaria llamada asociación, y que se caracteriza mediante dos funciones mono o multivalorada, inversas entre si, y que son interpretaciones de esta relación.

Ejemplo: Sean los conjuntos de entidades ESTUDIANTE, PROFESOR, ESTUDIOS podemos construir las siguientes asociaciones:

Matriculado(1)
ESTUDIANTE ----------ESTUDIOS
Estudiantematriculado(0)

Esta asociación expresa el hecho de que un estudiante está matriculado en una única carrera(estudios), y que a una carrera dada le corresponde una lista de estudiantes.

Enseña
ESTUDIOS----------PROFESOR
Profesoresdelosestudios

Nos dice que un profesor enseña en uno o varios estudios

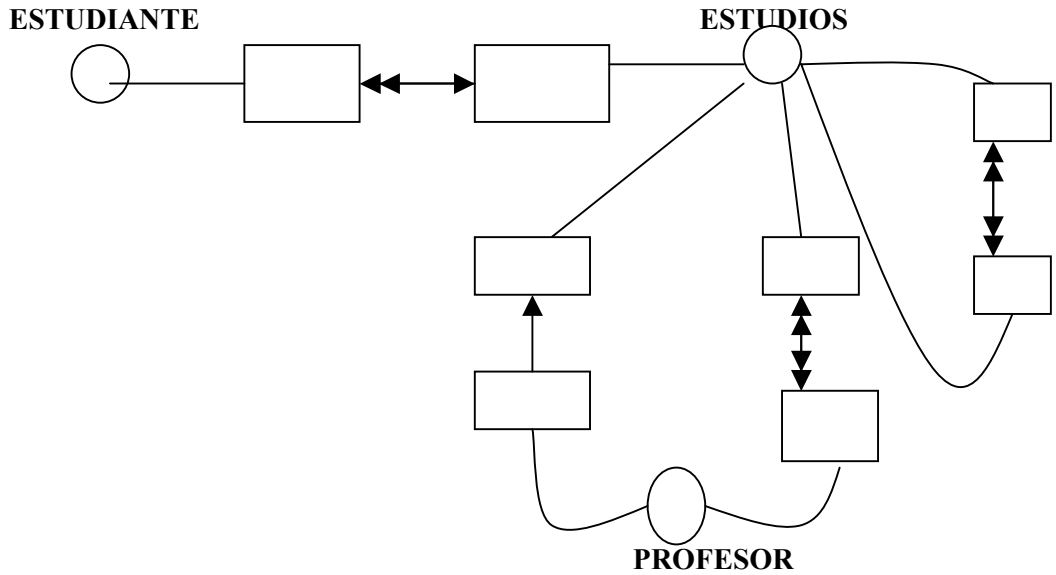
ESTUDIOS----------PROFESOR
Responsable(1)

Expresa el hecho de que cada uno de los estudios tiene un responsable

ESTUDIOS----------ESTUDIOS
Prerrequisito(0)

Dice cuáles son los prerrequisitos necesarios para cada uno de los estudios

Un esquema conceptual binario del anterior ejemplo sería:



Representaremos una asociación binaria en el esquema conceptual como un diagrama convencional en el que los círculos representarán los nombres de los conjuntos de entidades, los cuadrados representarán los nombres de las funciones, una flecha sencilla indicará una función monovalorada, y una flecha doble una función multivalorada. No se dará un nombre a una función si no es de ninguna utilidad en las expresiones de manipulación de datos.

Como ya hemos hecho referencia a una de las partes del modelo Z (referida al nivel conceptual), necesitaremos ahora hacer referencia a la otra parte, que se corresponde con el lenguaje

asociado y que nos es necesario para expresar correctamente las restricciones de integridad correspondientes.

3.4. MODELO SEMANTICO DE DATOS

El modelo de datos semántico (o SMD:semantic data model) introdujo los conceptos de clases y subclases en el modelado de datos; así pues, fue el origen de muchos de los conceptos que desde entonces se han incorporado en los modelos de datos conceptuales, como los modelos orientados a objetos. También clasificó las clases cuyos objetos representaban información con diversos tipos de semánticas.

El principal concepto de modelado de SMD es la clase, que es una colección de objetos del mismo tipo. Las propiedades (atributos) de una clase especifican el tipo de objetos que contiene. Las propiedades se clasifican como opcionales (se permiten nulos) u obligatorios (no se permiten nulos); simples (atómicas) o compuestas; monovaluadas o multivaluadas; derivables o almacenadas, y únicas o no únicas. Los objetos existen independientemente de cualquier valor de sus atributos. Cada propiedad está asociada a un dominio (conjunto de valores) del cual se pueden escoger sus valores para objetos individuales. Si el dominio de una propiedad es otra clase, los valores de la propiedad se refieren a objetos de la otra clase.

Las clase de objetos también se clasifican en varios tipos. Una clase de objetos concretos representa objetos con una existencia concreta en el minimundo. Una clase de objetos abstractos representa grupos de objetos de otras clases con propiedades idénticas. Por ejemplo, una clase AVIONES que contiene un objeto por cada avión individual propiedad de una aerolínea es una clase de objetos concretos, en tanto que una clase TIPOS_DE_AVIONES que contiene un objeto por cada tipo de avión(B737, B747, MD-11, DC-9, ...) es una clase de objetos abstractos. Una clase agregada contiene objetos que son agregados de otros objetos; por ejemplo, cada objeto de una clase agregada CONVOYES_DE_BUQUES consiste en un agregado de objetos de la clase de objetos concretos BUQUES. Una clase de sucesos incluye objetos temporales, como VIAJES o LLEGADAS.

Una subclase es un subconjunto de objetos de una clase base. Por ejemplo, BUQUES_TANQUE y BUQUES_CRUCERO son subclases de una clase base BUQUES. Una subclase de restricción está definida por predicado, lo que no sucede con las subclases sin restricción. Además de las subclases definidas por los usuarios, las clases de objetos abstractos y de agregación por lo regular definen ciertas subclases. Por ejemplo, cada objeto de la clase de objetos abstractos TIPOS_DE_AVIONES define una subclase de objetos de AVIONES que pertenecen a un cierto tipo.

3.5. MODELO INFOLOGICO

Las metodologías del desarrollo sobre el acercamiento infológico y el modelar conceptual (los datos que modelan) tienen su origen en ambientes estadísticos.

A comienzos de los años 70, Langefors y algunos de sus estudiantes, desviaron su atención al desarrollo de una teoría infológica, una teoría de conceptos y contenidos informativos. Sundgren (1973) desarrolló el framework OPR(t), también llamados modelos infológicos, o modelos conceptuales, como pasaron a llamarse más tarde en la literatura internacional. El framework OPR(t) está basado en rigurosos análisis de cuatro conceptos fundamentales: objetos, propiedades, relaciones (entre objetos) y tiempo. Langefors los llamó mensajes elementales, bloques de información. Podemos considerar al modelo E-R o al modelo E-A-R presentado por Chen en 1976 como un subconjunto de este modelo infológico cuya primera mejora había sido publicada por Sundgren tres años antes(1973). Las sucesivas mejoras corrieron a cargo de Sundgren en los dos años siguientes.

La teoría infológica, a principios de los 70, se utilizó como proyecto para llevar a cabo estadísticas económicas. Como proyecto paralelo también se desarrolló un catálogo de variables, y que era usado para el proyecto estadístico ya mencionado.

Pero, aún otro proyecto más importante iba a ser desarrollado, como fruto de la inspiración en las ideas infológicas. Se trataba de una base de datos que traía consigo tres grandes partes bien diferenciadas:

- Microbase de datos:
- Macrobase de datos: Contenía datos estadísticos agregados multidimensionalmente.

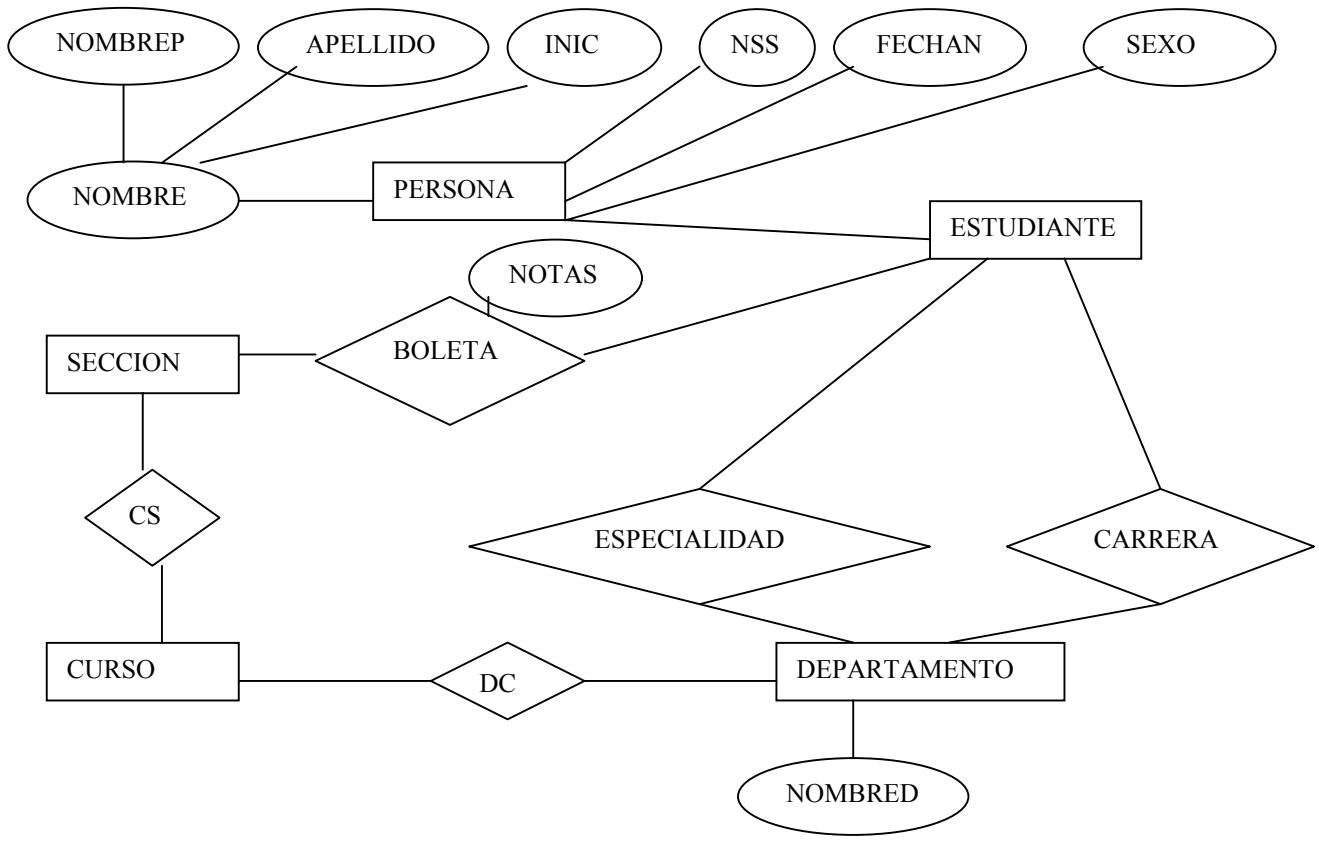
- Metabase de datos: Contenía descripciones técnicas y orientadas a contenidos de las microbases y de las macrobases.

Quizá, ya podíamos decir que este iba a ser uno de los primeros "datawarehouse" del mundo.

3.6. MODELO FUNCIONAL DE DATOS

Los modelos de datos funcionales (MDF) emplean el concepto de función matemática como elemento de modelado fundamental. Cualquier solicitud de información se puede visualizar como una llamada a una función con ciertos argumentos, y la función devuelve la información requerida. Existen varias propuestas de modelos de datos y lenguajes de consulta funcionales (el modelo DAPLEX y su lenguaje son el ejemplo más conocido.

Las principales primitivas del modelado de un MDF son entidades y vínculos funcionales. Hay varios tipos de entidades estándar en el nivel más básico, como "string", "integer", "character", y "real" entre otras, y se denominan tipos de entidades imprimibles. Los tipos de entidades abstractos que corresponden a objetos del mundo real tienen el tipo "entity". Así por ejemplo, considerando el siguiente diagrama EER:



Podremos mostrar cómo se pueden especificar algunos de los tipos de entidades y de vínculos de ese esquema en una notación funcional similar a la de DAPLEX. Los tipos de entidades PERSONA, ESTUDIANTE, CURSO, SECCION Y DEPARTAMENTO se declaran como sigue:

PERSONA() -> ENTITY
ESTUDIANTE()-> ENTITY
CURSO()-> ENTITY
SECCION()-> ENTITY
DEPARTAMENTO()-> ENTITY

Estos enunciados especifican que las funciones PERSONA, ESTUDIANTE, CURSO, SECCIÓN, y DEPARTAMENTO devuelven entidades abstractas; por tanto, estos enunciados sirven para definir los tipos de entidades correspondientes. Un atributo de un tipo de entidades también se especifica como una función cuyo argumento (dominio) es el tipo de entidades y cuyo resultado (intervalo) es una entidad imprimible. Las siguientes declaraciones de funciones especifican los atributos de PERSONA:

NSS(PERSONA)-> STRING
FECHAN(PERSONA)-> STRING
SEXO(PERSONA)-> STRING

Al aplicar la función NSS a una entidad de tipo PERSONA se devuelve el número de seguro social de la persona, que es un valor imprimible de tipo "string" (cadena). Si queremos declarar atributos compuestos como NOMBRE, deberemos declararlos como entidades y luego declarar sus atributos componentes como funciones:

NOMBRE()-> ENTITY
NOMBRE(PERSONA)-> NOMBRE
NOMBREP(NOMBRE)-> STRING
INIC(NOMBRE)-> CHAR
APELLIDO(NOMBRE)-> STRING

Para declarar un tipo de vínculos, le damos un nombre de papel en una dirección y lo definimos también como función:

CARRERA(ESTUDIANTE)-> DEPARTAMENTO
ESPECIALIDAD(ESTUDIANTE)-> DEPARTAMENTO

Estas declaraciones especifican que la aplicación de una función CARRERA o ESPECIALIDAD a una entidad ESTUDIANTE deberá devolver como resultado una entidad de tipo DEPARTAMENTO. Para declarar un nombre de papel inverso para los mismos tipos de vínculos:

CARRERA_EN(DEPARTAMENTO)->-> ESTUDIANTE INVERSE OF CARRERA
ESPECIALIDAD_EN(DEPARTAMENTO)->-> ESTUDIANTE INVERSE OF ESPECIALIDAD

La cláusula INVERSE OF declara que estas funciones son los inversos de las dos funciones previamente declaradas y que por ello especifican los mismos tipos de vínculos que las otras dos, pero en la dirección opuesta. Observamos las flechas dobles (->->), que especifican que aplicar la función CARRERA_EN o ESPECIALIDAD_EN a una sola entidad DEPARTAMENTO puede devolver un conjunto de entidades de tipo ESTUDIANTE. Esto especifica que los tipos de vínculos son 1:N. Esta notación también puede servir para especificar atributos multivaluados. Si queremos especificar un tipo de vínculos M:N, usamos flechas dobles en ambas direcciones:

CURSOS_COMPLETADOS(ESTUDIANTE)->-> SECCION
ESTUDIANTES_QUE_ASISTIERON(SECCION) ->->ESTUDIANTE INVERSE OR CURSOS_COMPLETADOS

Esto declara el vínculo BOLETA en ambas direcciones. Algunas funciones pueden tener más de un argumento:

NOTAS(ESTUDIANTE, SECCION)-> CHAR

Se observa que no es preciso declarar un inverso para cada vínculo. Por ejemplo, los vínculos DC y CS, que relacionan un curso con el departamento que lo ofrece y un curso con sus secciones, se pueden declarar en una sola dirección:

DEPARTAMENTO_QUE_OFRECE(CURSO)-> DEPARTAMENTO
SECCIONES_DE(CURSO)->-> SECCION

Un concepto fundamental en MDF es la composición de funciones. Al escribir NOMBRED(DEPARTAMENTO_QUE_OFRECE(CURSO)), componemos las dos funciones NOMBRED y DEPARTAMENTO_QUE_OFRECE. Esto se denomina expresión de camino y se puede escribir con una notación de punto alternativa como CURSO.DEPARTAMENTO_QUE_OFRECE.NOMBRED. La composición de funciones puede servir para declarar funciones derivadas; por ejemplo, las funciones NOMBREP(PERSONA) o NOMBREP(ESTUDIANTE) se declaran como composiciones de funciones previamente declaradas. Este enfoque también puede usarse para especificar los atributos heredados explícitamente.

La composición de funciones es también el principal concepto empleado en los lenguajes de consulta funcionales. Suponiendo que deseamos obtener los apellidos de todos los estudiantes que estudian la carrera “matemáticas”, haríamos:

RETRIEVE APELLIDO(NOMBRE(ES_UNA_PERSONA(CARRERA_EN(DEPARTAMENTO)))) WHERE NOMBRED(DEPARTAMENTO)=’Matematicas’

Aquí usamos la función CARRERA_EN, que es el inverso de CARRERA. Como alternativa podríamos usar la función derivada APELLIDO(ESTUDIANTE) para acortar la consulta anterior:

RETRIEVE APELLIDO(CARRERA_EN(DEPARTAMENTO)) WHERE NOMBRED(DEPARTAMENTO)=’Matematicas’

La función inversa CARRERA_EN(DEPARTAMENTO) devuelve entidades ESTUDIANTE, así que podemos aplicar la función derivada APELLIDO(ESTUDIANTE) a esas entidades.

CAPITULO 4:MODELOS LOGICOS BASADOS EN REGISTROS- MODELOS CLASICOS

Operan sobre niveles conceptual y de visión. Sus características principales son que permiten una descripción más amplia de la implantación, pero no son capaces de especificar con claridad las fronteras de los datos. Los modelos clásicos presentan una serie de características comunes:

- Uniformidad: Una gran cantidad de datos se estructuran de modo similar.
- Orientación en registros: Los datos básicos se almacenan en registros de longitud fija.
- Datos pequeños: Los registros son cortos, normalmente de pocos cientos de bytes.
- Campos atómicos: Los campos de los registros son cortos, de longitud fija y no poseen ningún tipo de estructura interna.
- Transacciones cortas: En las transacciones no tienen interacción con el usuario y además su duración es de alguna fracción de segundo.
- Esquema case estático: No suelen realizarse cambios en los esquemas, y si aparecen son de poca importancia.

Para el diseño lógico se recibe como entrada el esquema conceptual y da como resultado un esquema lógico, que es una descripción de la estructura de la base de datos que puede procesar el software DBMS.

4.1. MODELO RELACIONAL

Codd, (1970), propone un modelo de datos basado en la Teoría de las relaciones, donde los datos se estructuran lógicamente en forma de relaciones (TABLAS), siendo un objetivo fundamental mantener la independencia de la estructura lógica respecto al modelo de almacenamiento y a otras características del tipo físico.

Los objetivos para este modelo serán conseguir:

- Independencia física .- Que el modo en que se almacenan los datos no influya en su manipulación lógica, y por tanto, no sea necesario modificar los programas por cambios en el almacenamiento físico. (Codd concede mucha importancia a este aspecto → Independencia de ordenación , independencia de indexación e independencia en criterios de acceso).
- Independencia Lógica .- Que la modificación de objetos en la base de datos no repercuta en los programas y/o usuarios que estén accediendo al subconjunto parcial de la base de datos.
- Flexibilidad .- Poder presentar a cada usuario los datos de la forma que prefiera.
- Uniformidad.- Las estructuras logicas de datos presentan un estado uniforme.
- Sencillez,- siendo al mismo tiempo la principal característica -, en la estructura puesta de manifiesto sobre todo en las operaciones de recuperación y actualización. Esto se debe a que trata de igual forma las entidades y las relaciones, lo que provoca inconvenientes.

Este modelo, no permite distinguir entre objetos y asociaciones de objetos (ambos se representan por una única estructura llamada relación), por lo que su contenido semántico es menor que en el modelo Codasyl.

Como ya hemos adelantado en los objetivos, el acceso a los datos se realiza en función a las propiedades de éstos, no conociendo el usuario el camino de acceso y ocupándose así el sistema de seleccionar el camino físico y el tiempo de respuesta. Por tanto, nos encontramos con una mayor independencia físico-lógica.

Nos encontramos ante un modelo no muy eficiente, ya que ésto depende del equipo.

El modelo relacional es una forma de ver los datos, es decir, una receta para representar los datos mediante tablas y para manipular su representación. El modelo relacional se ocupa de 3 puntos importantes que son la estructura, la integridad y la manipulación de los datos.

Los componentes de la estructura son básicamente Relación, Tupla, Cardinalidad, Atributo, Grado, Dominio y Clave Primaria.

Una relación corresponde a lo que se conoce como tabla o entidad; una tupla corresponde a una fila de esa tabla, un atributo corresponde a una columna de la misma tupla, el número de tuplas se denomina cardinalidad, el número de atributos se llama grado, la clave primaria es un identificador

único para la tabla, es decir, una columna o combinación de columnas con la siguiente propiedad: "nunca existirán dos filas de la tabla con el mismo valor de una columna o combinación de columnas". El Dominio es una combinación de valores de los cuales uno o mas atributos obtienen sus valores reales.

La estructura de un enfoque relacional es en sí donde las asociaciones entre tuplas se representan únicamente por valores de datos en las columnas sacadas de un dominio común.

TERMINO RELACIONAL	**TERMINO INFORMAL**
RELACION	TABLA
TUPLA	FILA O REGISTRO
CARDINALIDAD	Nº DE FILAS
ATRIBUTO	CAMPO O COLUMNA
GRADO	Nº DE COLUMNA
CLAVE PRIMARIA	IDENTIFICADOR UNICO
DOMINIO	VALORES LEGALES

DOMINIO: Corresponde a un conjunto de valores que están entre algún tipo de rango, por ejemplo: El Dominio del código producto; es el conjunto de todos los productos posibles, el dominio del valor unitario es el conjunto de los reales mayores que 0, es decir, los dominios son fondos de valores de los cuales se extraen los valores reales, que aparecen en los atributos. Cada atributo debe estar definido sobre un dominio lo que significa que los valores de ese atributo deben proceder de ese dominio especifico.

RELACIONES: Están compuestas por dos partes una cabecera y un cuerpo, se define la relación sobre un conjunto de dominio, la cabecera esta formada por un conjunto fijo de atributos tal que cada atributo A(j) corresponde a cada uno de los Dominios de J, con J variando hasta N.

El cuerpo está formado por un conjunto de tuplas, el cual varía con el tiempo, cada tupla esta formada por un conjunto de pares (atributo, valor) Donde (i) varia de 1 hasta N y N es el número de tuplas del conjunto (A1, V1) (A2, V2), (A3,V3).....(An, Vn) N va a ser el grado de esa relaciòn.

Para este modelo tendremos varios tipos de relaciones:

- Relaciones base o reales: Corresponde al concepto de Tabla es decir una relación autónoma cuya importancia esta dada por el diseñador para un uso especifico dentro de una aplicación.
- Relaciones virtuales: (Relaciones de Vistas) Una vista es una relación derivada con nombre, representada dentro del sistema exclusivamente mediante su definición en término de otras relaciones, no posee datos almacenados propios, separados y distinguibles a diferencia de las relaciones bases, en si una vista.
- Relaciones instantaneas: (Snap Shop) Es también una relación derivada con nombre como una vista, pero a diferencia de esta última, las instantáneas son reales y no virtuales, es decir, están representadas no solo por su definición, en término de otras relaciones con nombre, sino, también por sus propios datos almacenados. (Snap Shop= consulta rápida, corta).

La mayor parte de las bases de datos están sujetas a un gran número de reglas de integridad. Por ejemplo, los códigos de ciertos objetos deben tener la forma "XXX"; los valores posibles de un atributo podrían variar entre 1 y 9.999; las ciudades deberían provenir de cierta tabla previamente definida; el atributo stock físico debería ser mayor que 0; etc.

La primera regla general de integridad se relaciona con la clave primaria, y la segunda, con las claves ajenas:

- Clave primaria: En términos informales la clave primaria de una relación es solo un identificador único para esa relación. La componen dos o más atributos de la misma relación o también puede ser un resumen de alguna porción de los atributos de la relación. Cabe la posibilidad de definir claves cuyo identificador no es único para una tupla especifica o para un conjunto de tuplas, en este caso la clave pasa a ser llamada clave secundaria, presentando una probabilidad del 60% de que sea duplicada, también en este tipo de clave se conoce como clave alternativa duplicada, para el caso de acceder una o un grupo de tuplas, dando ésta como una segunda alternativa. Para poder determinar cuál es

la clave especifica se deben primero examinar los atributos de la relación, es decir, claves candidatas y dentro de esta claves candidatas una de ellas sería la clave primaria y el resto claves secundarias si es posible. Las condiciones que debiera tener la clave candidata es si el atributo K de la relación R es una clave candidata de R si y solo si satisface las siguientes dos propiedades: unicidad (en cualquier momento no existen dos tuplas en R con el mismo valor de K), y minimalidad (si K es compuesto no será posible eliminar ningún componente de K sin destruir la propiedad de unicidad. Toda relación tiene por lo menos una clave candidata. El razonamiento para elegir la clave primaria cuando existen varias claves candidatas, queda al alcance del modelo relacional; en la práctica la elección es más sencilla. En sí, la clave primaria es la que tiene mayor importancia. Las claves candidatas y alternativas son solo conceptos que nacen durante el proceso de definición de la clave primaria. Para terminar, decir que ninguna clave primaria debe contener valores nulos y que ninguna base de datos relacional registrará información de algo que no se puede identificar plenamente.

- Claves ajenas: Es un atributo de una relación R2 cuyos valores deben concordar con las claves primarias de alguna relación R1, es decir, un valor de la clave ajena representa una referencia a la tupla donde se encuentra el valor correspondiente de la clave primaria (tupla referenciada o tubla objetivo).

Al igual que ocurre en los demás modelos, nos encontraremos con un conjunto de restricciones que dividiremos en dos tipos:

- Inherentes:
  - No hay dos tuplas iguales.
  - El orden de las tuplas no es significativo.
  - El orden de los atributos no es significativo.
  - Cada atributo solo puede tomar un valor del dominio, no admitiéndose por tanto dos grupos repetitivos.
  - Se debe cumplir la regla de Integridad de entidad → ‘ Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor desconocido o inexistente’.
- De usuario:

- Integridad referencial → Si la relación R2 tiene un descriptor que referencia a la clave primaria de la relación R1 todo valor de dicha clave (Clave ajena R2) debe concordar con los valores de la clave primaria R1 o ser NULO. R1 y R2 son claves necesariamente distintas. Además, la clave ajena puede formar parte de la relación R2.

También se deben definir las acciones a tomar en caso de acciones de modificación y borrado :

- Operación restringida .- Solo se puede borrar una fila de la tabla que tiene clave primaria referenciada si no existen filas con esa clave en la tabla referenciada.
- Operación con transmisión en cascada.- El borrado o la modificación de una fila de la tabla que contiene la clave primaria lleva consigo la modificación de las tablas cuya clave ajena coincida con la clave primaria modificada.
- Operación con puesta a nulos.- El borrado o la modificación de una fila de la tabla que contiene la tabla primaria lleva consigo la puesta nulos de los valores de la clave ajena de las filas de la tabla que referencia cuya clave coincida con el valor de la clave primaria de la tabla referenciada.

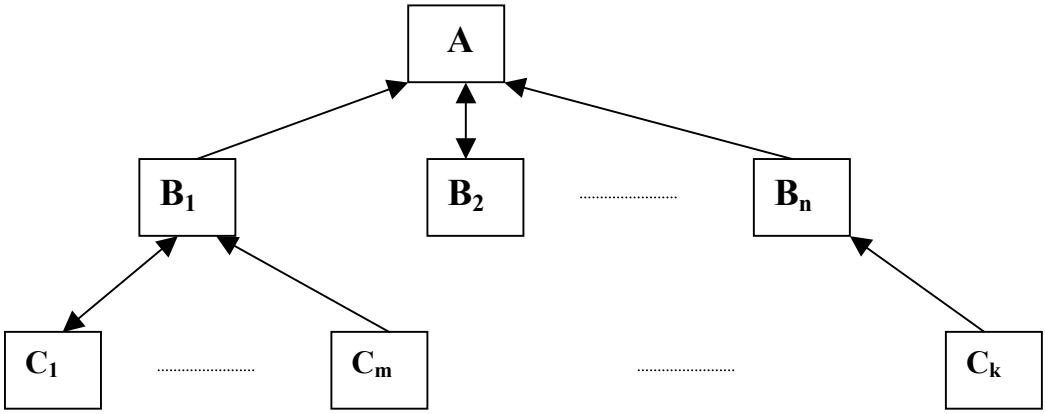
- Otra restricciones:
  - Definir predicados sobre :
    . Atributo de una relación.
    . Tuplas de una relación.
    . Atributos de varias relaciones.
    . Dominios.
  - Evaluación de restricciones:
    . Con cada operación.
    . Diferido.

4.2. MODELO JERARQUICO

Hace más acentuada que en el modelo Codasyl la difícil independencia entre el nivel lógico y el físico.

Este modelo fue el primero en aparecer y se caracteriza por organizar la información a través de una estructura de árboles, en la que las relaciones entre instancias o registros se expresan mediante una jerarquía. Dicha jerarquía distribuye y ordena los datos mediante un recorrido en Preorden.

La estructura general de un diagrama de estructura de árbol sería:

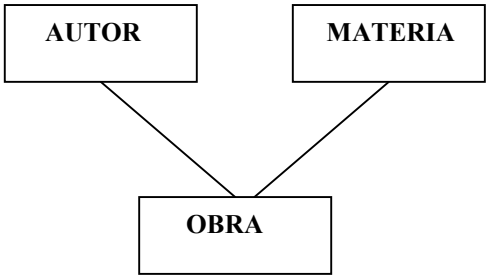


En este modelo contamos con:

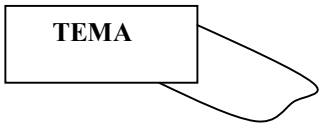
- Diagramas de ocurrencia(DO):
  - Cada uno de los rectángulos representa la instancia de un tipo de registro.
  - Los arcos definen la relación existente entre las instancias de dos niveles contiguos del árbol.
  - No se suelen especificar los campos de los registros.
  - Cuando la información almacenada en la base de datos es muy grande se convierte en inmanejable.
- Diagrama de Bachman(DA):
  - Describe gráficamente el esquema lógico de una base de datos jerárquica.
  - Los rectángulos representan los tipos de registro.
  - Los arcos representan la relación existente entre los tipos de registro.
  - Los arcos se etiquetan porque puede existir más de una relación entre dos tipos de registro (no en el modelo jerárquico).
  - La flecha simple/doble indica que existe un único padre para cada hijo, pero pueden existir cero o más hijos de un psdre.

Contamos también con una serie de situaciones no permitidas inherentes al modelo:

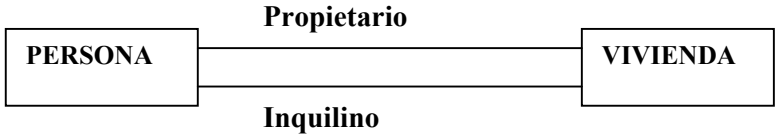
- Hijos con más de un padre. Ejemplo:



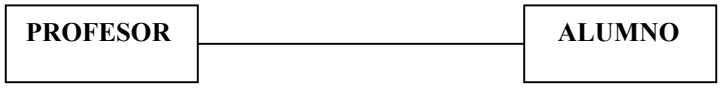
- Interrelaciones reflexivas. Ejemplo:



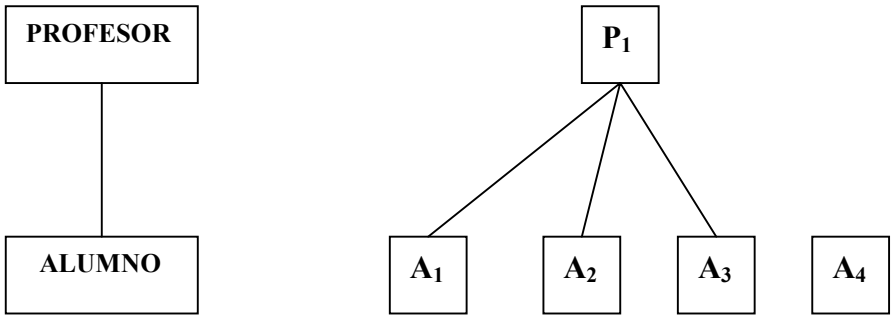
- Interrelaciones nominadas, y, por tanto, más de una asociación entre dos entidades. Ejemplo:



- Interrelaciones N:M.Ejemplo:



- Hijos sin padre. Ejemplo:



La estructura jerárquica cuenta con una gran cantidad de inconvenientes conocidos, por lo que su utilización en la actualidad es prácticamente nula. La flexibilidad y falta de estructuración de este modelo permite crear registros cuyos campos sean variables en número y tamaño; además cuando la información almacenada en la base de datos es muy grande, se convierte en inmanejable. En cuanto a limitaciones lógicas cabe destacar que este modelo sólo soporta relaciones del tipo 1 a N (ya que las de tipo N a N requieren un inaceptable nivel de redundancia). Además, si un registro tipo aparece como hijo en más de dos relaciones, se debe de replicar. Esto puede producir problemas de integridad y consistencia de los datos. Los lenguajes de manipulación asociados son fuertemente navegacionales y hace falta un lenguaje de programación anfitrión en el que se inserten los operadores.

4.3. MODELO DE RED

Modelo Codasyl: Difícil independencia entre el nivel lógico y el físico al presentar éste una correspondencia directa entre las relaciones lógicas y los caminos de acceso físicos.

Este modelo fue concebido como una ampliación del modelo jerárquico, cuya finalidad era solucionar las deficiencias lógicas de este último. Al igual que el anterior, también se emplea un árbol como estructura base, pero con la diferencia de que un mismo hijo puede tener diferentes padres, con lo que es posible representar relaciones N a N sin redundancia aparente. Por el contrario, desaparece la herencia de los campos. En algunas versiones modernas de éste modelo incluso encontramos la aparición de Registros Enlaces para establecer relaciones N a N. Como inconveniente presenta la complejidad que alcanza el entramado de enlaces entre las instancias cuando se almacenan gran cantidad de datos, así como la hostilidad de los lenguajes de programación y control de estas bases de datos.

Podemos distinguir dos tipos básicos de redes:

- Simples: Se dice que la red es simple si los padres de un hijo son instancias de registros tipo diferentes.
- Complejas: Se dice que la red es compleja si los padres pueden ser instancias del mismo registro tipo.

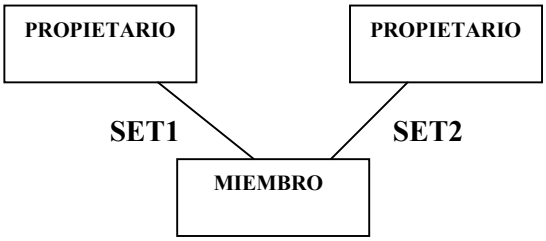
La conversión de compleja a simple nos permite reducir el problema de la pérdida de información asociado a las redes complejas. La idea es convertir una relación muchos a muchos en dos relaciones uno a muchos mediante la inserción de un nuevo tipo de registro. Este registro se denomina registro de intersección si contiene algún tipo de información, que se denomina datos de la intersección, en otro caso, se denomina registro de enlace.

Existen dos tipos de relaciones específicas; los ciclos y los lazos. En un ciclo, diferentes tipos de registros se relacionan de tipo circular. Los lazos representan la relación de un tipo de registro consigo mismo y éstos sólo pueden manejarse en redes complejas.

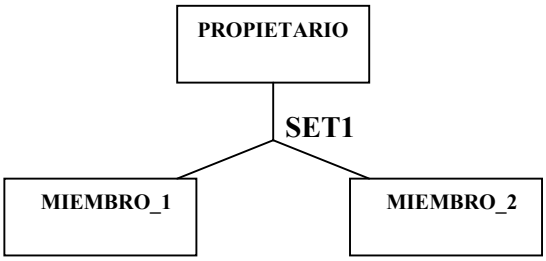
Una jerarquía es un caso particular de una estructura en red y, además no tiene ninguna de las restricciones que veíamos para el modelo jerárquico.

Algunas de las situaciones no permitidas del modelo anterior, en el modelo CODASYL son tratadas de la siguiente manera:

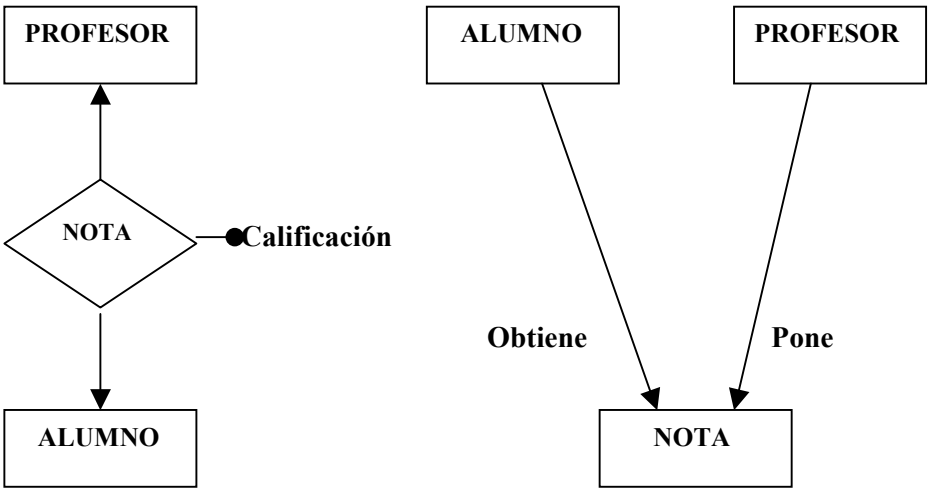
- Estructura en red: miembro con varios propietarios:

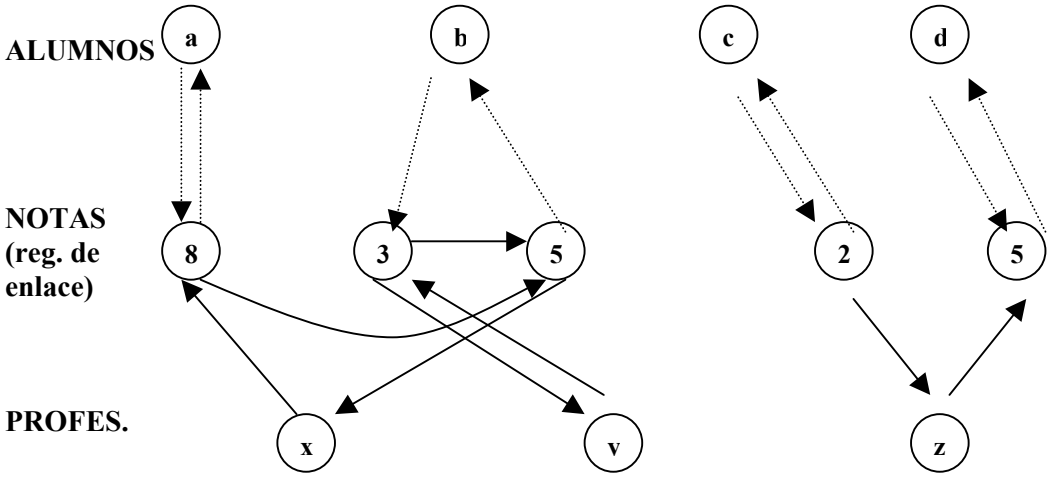


- Conjunto multimiembro:



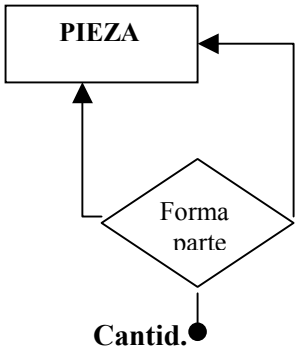
- Interrelación N:M:



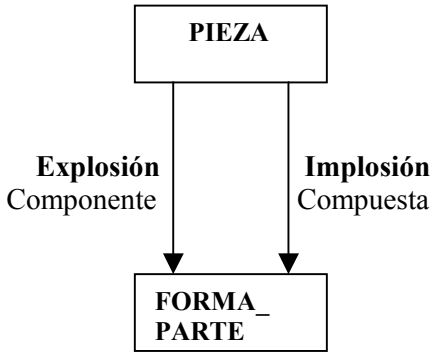


- Interrelación reflexiva:

TIPO DE INTERRELACION NO PERMITIDA



SOLUCION MEDIANTE DOS CONJUNTOS



Cantid. Es un elemento de datos de este registro

CAPITULO 5:MODELOS DE DATOS AVANZADOS

5.1. BASES DE DATOS ORIENTADAS A OBJETOS

5.1.1. Introducción

Como cualquier base de datos programable, una base de datos orientada a objetos (BDOO) proporciona un ambiente para el desarrollo de aplicaciones y un depósito persistente listo para su explotación. Una BDOO almacena y manipula información que puede ser digitalizada (presentada) como objetos, proporciona una estructura flexible y rica con acceso ágil y rápido y permite una gran capacidad de modificación. A pesar de que el mercado de las BDOO está en su infancia, las capacidades que se vislumbran en un manejador orientado a objetos para manipular relaciones complejas con un desempeño potencialmente espectacular son dignos de investigación.

5.1.2. Aspectos de la tecnología

Los objetos pueden estar compuestos o consistir de cualquier tipo de información que, eventualmente, puede almacenarse en forma digital; por ejemplo imágenes barridas (Scaned), voz y sonido, dibujos y planos arquitectónicos complejos, esquemas electrónicos y diagramas desarrollados por ingenieros, así como los tradicionales tipos de datos alfanuméricos. Las BDOO permiten que múltiples usuarios compartan objetos complejos y los manipulen en un ambiente seguro y estructurado. Las bases de datos convencionales fueron diseñadas para manejar tipos de datos alfanuméricos y por esto dificilmente pueden manipular objetos y métodos (los métodos son los comportamientos definidos de los objetos). Una base de datos en red o jerárquica puede almacenar objetos complejos, pero esta arquitectura no es flexible , lo cual motiva, por ejemplo, el uso del modelo relacional. Estos sistemas casi no permiten flexibilidad para modificaciones, y el sistema debe desactivarse cuando se requiere modificar estructuras de objetos y métodos. Los niveles en los cuales las bases de datos incorporan los conceptos alrededor de la metodología de objetos. La primera clase, puede denominarse BDOO pasivas o estructuralmente orientadas a objetos que permiten manejar objetos compuestos. Una base de datos pasiva puede almacenar objetos complejos pero no puede definir comportamientos. Una BDOO es activa u orientada a objetos por comportamiento si permite definir y ejecutar comportamientos de los objetos dentro de la base de datos, incorpora conceptos como herencia y permite el manejo de tipos definidos por el usuario.

5.1.3 Ventajas de las BDOOs

Entre las ventajas más ilustrativas de las BDOOs está su flexibilidad y soporte para el manejo de tipos de datos complejos. Una segunda ventaja es que en una BDOO manipula datos complejos rápida y ágilmente. No se requieren búsquedas en tablas o uniones para crear relaciones.

5.1.4. Posibles problemas

Al considerar la adopción de la tecnología orientada a objetos, la inmadurez del mercado de BDOOs constituye una posible fuente de problemas, y debe analizarse con detalle la presencia en el mercado del proveedor para adaptar su producto de forma sustantiva. El segundo problema es la falta de estándares en la industria orientada a objetos.

5.2. BASES DE DATOS DISTRIBUIDAS

En el presente apartado se mostrará la arquitectura general de un sistema de bases de datos distribuida, se introducirá el concepto de fragmentación de datos relacionado con el nivel de transparencia de distribución que un SBDD debe ofrecer. Se dará una descripción acerca de las componentes de las bases de datos distribuidas.

La arquitectura define la estructura de un sistema. Al definir la arquitectura se deben identificar las componentes de un sistema, las funciones que realiza cada una de las componentes y las interrelaciones e interacciones entre cada componente.

5.2.1. Niveles de transparencia en SBDD

El propósito de establecer una arquitectura de un sistema de bases de datos distribuidas es ofrecer un nivel de transparencia adecuado para el manejo de la información. La transparencia se puede entender como la separación de la semántica de alto nivel de un sistema de los aspectos de bajo nivel relacionados a la implementación del mismo. Un nivel de transparencia adecuado permite ocultar los detalles de implementación a las capas de alto nivel de un sistema y a otros usuarios.

En sistemas de bases de datos distribuidos el propósito fundamental de la transparencia es proporcionar independencia de datos en el ambiente distribuido. Se pueden encontrar diferentes aspectos relacionados con la transparencia. Por ejemplo, puede existir transparencia en el manejo de la red de comunicación, transparencia en el manejo de copias repetidas o transparencia en la distribución o fragmentación de la información.

La independencia de datos es la inmunidad de las aplicaciones de usuario a los cambios en la definición y/u organización de los datos y viceversa. La independencia de datos se puede dar en dos aspectos: lógica y física.

- Independencia lógica de datos. Se refiere a la inmunidad de las aplicaciones de usuario a los cambios en la estructura lógica de la base de datos. Esto permite que un cambio en la definición de un esquema no debe afectar a las aplicaciones de usuario. Por ejemplo, el agregar un nuevo atributo a una relación, la creación de una nueva relación, el reordenamiento lógico de algunos atributos.
- Independencia física de datos. Se refiere al ocultamiento de los detalles sobre las estructuras de almacenamiento a las aplicaciones de usuario. Esto es, la descripción física de datos puede cambiar sin afectar a las aplicaciones de usuario. Por ejemplo, los datos pueden ser movidos de un disco a otro, o la organización de los datos puede cambiar.

La transparencia al nivel de red se refiere a que los datos en un SBDD se accesan sobre una red de computadoras, sin embargo, las aplicaciones no deben notar su existencia. La transparencia al nivel de red conlleva a dos cosas:

- Transparencia sobre la localización de datos. Esto es, el comando que se usa es independiente de la ubicación de los datos en la red y del lugar en donde la operación se lleve a cabo. Por ejemplo, en Unix existen dos comandos para hacer una copia de archivo. Cp se utiliza para copias locales y rcp se utiliza para copias remotas. En este caso no existe transparencia sobre la localización.
- Transparencia sobre el esquema de nombramiento. Lo anterior se logra proporcionando un nombre único a cada objeto en el sistema distribuido. Así, no se debe mezclar la información de la localización con en el nombre de un objeto.

La transparencia sobre replicación de datos se refiere a que si existen réplicas de objetos de la base de datos, su existencia debe ser controlada por el sistema no por el usuario. Se debe tener en cuenta que al cuando el usuario se encarga de manejar las réplicas en un sistema, el trabajo de éste es mínimo por lo que se puede obtener una eficiencia mayor. Sin embargo, el usuario puede olvidarse de mantener la consistencia de las réplicas teniendo así datos diferentes.

La transparencia a nivel de fragmentación de datos permite que cuando los objetos de la bases de datos están fragmentados, el sistema tiene que manejar la conversión de consultas de usuario definidas sobre relaciones globales a consultas definidas sobre fragmentos. Así también, será necesario mezclar las respuestas a consultas fragmentadas para obtener una sola respuesta a una consulta global. El acceso a una base de datos distribuida debe hacerse en forma transparente.

En resumen, la transparencia tiene como punto central la independencia de datos. Los diferentes niveles de transparencia se puede organizar en capas como se muestra en la Figura 2.3. En

el primer nivel se soporta la transparencia de red. En el segundo nivel se permite la transparencia de replicación de datos. En el tercer nivel se permite la transparencia de la fragmentación. Finalmente, en el último nivel se permite la transparencia de acceso (por medio de lenguaje de manipulación de datos).

La responsabilidad sobre el manejo de transparencia debe estar compartida tanto por el sistema operativo, el sistema de manejo de bases de datos y el lenguaje de acceso a la base de datos distribuida. Entre estos tres módulos se deben resolver los aspectos sobre el procesamiento distribuido de consultas y sobre el manejo de nombres de objetos distribuidos.

5.2.2. Arquitectura de un sistema de bases de datos distribuidas

La mayoría de los sistemas de manejo de bases de datos disponibles actualmente están basadas en la arquitectura ANSI-SPARC la cual divide a un sistema en tres niveles: interno, conceptual y externo.

La vista conceptual, conocida también como vista lógica global, representa la visión de la comunidad de usuarios de los datos en la base de datos. No toma en cuenta la forma en que las aplicaciones individuales observan los datos o como éstos son almacenados. La vista conceptual está basada en el esquema conceptual y su construcción se hace en la primera fase del diseño de una base de datos.

Los usuarios, incluyendo a los programadores de aplicaciones, observan los datos a través de un esquema externo definido a nivel externo. La vista externa proporciona una ventana a la vista conceptual lo cual permite a los usuarios observar únicamente los datos de interés y los aísla de otros datos en la base de datos. Puede existir cualquier número de vistas externas y ellos pueden ser completamente independientes o traslaparse entre sí.

El esquema conceptual se mapea a un esquema interno a nivel interno, el cual es el nivel de descripción más bajo de los datos en una base de datos. Este proporciona una interfaz al sistema de archivos del sistema operativo el cual es el responsable del acceso a la base de datos. El nivel interno tiene que ver con la especificación de qué elementos serán indexados, qué técnica de organización de archivos utilizar y como los datos se agrupan en el disco mediante "clusters" para mejorar su acceso.

Desafortunadamente, no existe un equivalente de una arquitectura estándar para sistemas de manejo de bases de datos distribuidas. La tecnología y prototipos de SMBDD se han desarrollado más o menos en forma independiente uno de otro y cada sistema ha adoptado su propia arquitectura.

Para definir un esquema de estandarización en bases de datos distribuidas se debe definir un modelo de referencia el cual sería un marco de trabajo conceptual cuyo propósito es dividir el trabajo de estandarización en piezas manejables y mostrar a un nivel general como esas piezas se relacionan unas con otras. Para definir ese modelo de referencia se puede seguir uno de los siguientes tres enfoques:

- Basado en componentes. Se definen las componentes del sistema junto con las relaciones entre ellas. Así, un SMBD consiste de un número de componentes, cada uno de los cuales proporciona alguna funcionalidad.
- Basado en funciones. Se identifican las diferentes clases de usuarios junto con la funcionalidad que el sistema ofrecerá para cada clase. La especificación del sistema en esta categoría típicamente determina una estructura jerárquica para las clases de usuarios. La ventaja de este enfoque funcional es la claridad con la cual se especifican los objetivos del sistema. Sin embargo, este enfoque no proporciona una forma de alcanzar los objetivos.
- Basado en datos. Se identifican los diferentes tipos de descripción de datos y se especifica un marco de trabajo arquitectural el cual define las unidades funcionales que realizarán y/o usarán los datos de acuerdo con las diferentes vistas. La ventaja de este enfoque es la importancia que asigna al manejo de datos. Este es un enfoque significativo para los SMBD dado que su propósito principal es manejar datos. Sin embargo, la desventaja de este enfoque es que es prácticamente imposible especificar un modelo arquitectural sin especificar los modelos para cada una de sus unidades funcionales. Este es el enfoque seguido por el modelo ANSI/SPARC.

5.2.3. Alternativas para la implementación de SMBD

Las diferentes dimensiones (factores) que se deben considerar para la implementación de un sistema manejador de base de datos son tres:

- Distribución. Determina si las componentes del sistema están localizadas en la misma computadora o no.
- Heterogeneidad. La heterogeneidad se puede presentar a varios niveles: hardware, sistema de comunicaciones, sistema operativo o SMBD. Para el caso de SMBD heterogéneos ésta se puede presentar debido al modelo de datos, al lenguaje de consultas o a los algoritmos para manejo de transacciones.
- Autonomía. La autonomía se puede presentar a diferentes niveles:
  - Autonomía de diseño. La habilidad de un componente del SMBD para decidir cuestiones relacionadas a su propio diseño.
  - Autonomía de comunicación. La habilidad de un componente del SMBD para decidir como y cuando comunicarse con otros SMBD.
  - Autonomía de ejecución. La habilidad de un componente del SMBD para ejecutar operaciones locales de la manera que él quiera.

Desde el punto de vista funcional y de organización de datos, los sistemas de datos distribuidos están divididos en dos clases separadas, basados en dos filosofías totalmente diferentes y diseñados para satisfacer necesidades diferentes:

- Sistemas de manejo de bases de datos distribuidos homogéneos
- Sistemas de manejo de bases de datos distribuidos heterogéneos

Un SMBDD homogéneo tiene múltiples colecciones de datos; integra múltiples recursos de datos. Los sistemas homogéneos se parecen a un sistema centralizado, pero en lugar de almacenar todos los datos en un solo lugar, los datos se distribuyen en varios sitios comunicados por la red. No existen usuarios locales y todos ellos accesan la base de datos a través de una interfaz global. El esquema global es la unión de todas las descripciones de datos locales y las vistas de los usuarios se definen sobre el esquema global.

Para manejar los aspectos de la distribución, se deben agregar dos niveles a la arquitectura estándar ANSI-SPARC. El esquema de fragmentación describe la forma en que las relaciones globales se dividen entre las bases de datos locales. El esquema de asignación especifica el lugar en el cual cada fragmento es almacenado. De aquí, los fragmentos pueden migrar de un sitio a otro en respuesta a cambios en los patrones de acceso.

La clase de sistemas heterogéneos es aquella caracterizada por manejar diferentes SMBD en los nodos locales. Una subclase importante dentro de esta clase es la de los sistemas de manejo multi-bases de datos. Un sistema multi-bases de datos (Smulti-BD) tiene múltiples SMBDs, que pueden ser de tipos diferentes, y múltiples bases de datos existentes. La integración de todos ellos se realiza mediante subsistemas de software. En contraste con los sistemas homogéneos, existen usuarios locales y globales. Los usuarios locales accesan sus bases de datos locales sin verse afectados por la presencia del Smulti-BD.

En algunas ocasiones es importante caracterizar a los sistemas de bases de datos distribuidas por la forma en que se organizan sus componentes. La arquitectura basada en componentes de un SMBD distribuido consiste de dos partes fundamentales, el procesador de usuario y el procesador de datos. El procesador de usuario se encarga de procesar las solicitudes del usuario, por tanto, utiliza el esquema externo del usuario y el esquema conceptual global. Así también, utiliza un diccionario de datos global. El procesador de usuario consiste de cuatro partes: un manejador de la interfaz con el usuario, un controlador semántico de datos, un optimizador global de consultas y un supervisor de la ejecución global. El procesador de datos existe en cada nodo de la base de datos distribuida. Utiliza un esquema local conceptual y un esquema local interno. El procesador de datos consiste de tres partes: un procesador de consultas locales, un manejador de recuperación de fallas locales y un procesador de soporte para tiempo de ejecución.

La arquitectura basada en componentes de un sistema multi-bases de datos consiste en un sistema de manejo de bases datos para usuarios globales y de sistemas de manejo de bases de datos para usuarios locales. Las solicitudes globales pasan al procesador global el cual consiste de un procesador de transacciones, una interfaz de usuario, un procesador de consultas, un optimizador de

consultas, un esquema y un administrador de recuperación de fallas, todos ellos actuando de manera global.

En cada sitio existe un SMBD completo el cual consiste de la interfaz con el usuario, el procesador y optimizador de consultas, el manejador de transacciones, el despachador de operaciones, el manejador de recuperación de fallas y el sistema de soporte para tiempo de ejecución, todos ellos actuando de manera local. Para comunicar el sistema global con los sistemas locales se define una interfaz común entre componentes mediante la cual, las operaciones globales se convierten en una o varias acciones locales.

El manejo de directorio de datos es de una importancia mayor en bases de datos distribuidas. Por un lado, puede haber directorios locales o un solo directorio global. Por otra parte, su manejo puede ser local o distribuido. Finalmente, desde otro punto de vista el directorio puede ser replicado o no replicado. Existen combinaciones, en estas tres dimensiones, que no tienen mayor relevancia. Sin embargo, en varios de los vértices del cubo en tres dimensiones aparecen las combinaciones importantes para bases de datos distribuidas.

5.3. BASES DE DATOS DEDUCTIVAS

Las bases de datos deductivas forman un dominio de aplicación natural y avanzado para las técnicas que formalizan el razonamiento revisable. Cuando, desde un punto de vista lógico, las bases de datos deductivas simplemente extienden las bases de datos relacionales en una forma natural y elegante, la representación del conocimiento que fue restringida a hechos lógicos elementales (átomos terminales) es extendida a fórmulas lógicas más complejas.

Se ha mencionado que el razonamiento es comúnmente revisable cuando es tratado en el contexto de información disyuntiva, incompleta, negativa o implícita. La asunción del mundo cerrado es una postura satisfactoria en los casos de conocimiento consistente de hechos elementales solamente y cuando los procesos de consulta se restringen a hechos terminales positivos solamente, cuando uno quiere representar o inferir información disyuntiva, se debe de hacer uso de técnicas más elaboradas.

Presentaremos intuitivamente una extensión de la asunción mundo cerrado que ha sido propuesta para tratar con información disyuntiva, que ha sido llamada asunción mundo cerrado generalizada basada en la siguiente consideración semántica: En el campo de las bases de datos deductivas, solamente tomamos en cuenta Modelos Herbrand de las bases de datos. Podemos representar un modelo Herbrand M de una base de datos D por medio del conjunto de átomos terminales que son verificados en M . Por definición bases de datos Horn no contienen información disyuntiva. Vistas como un conjunto de fórmulas lógicas, esas bases de datos poseen la propiedad modelo intersección, esta propiedad puede ser establecida de la siguiente manera: La intersección de todos los Modelos Herbrand de una base de datos Horn consistente y finita es un modelo de esta base de datos, este modelo es llamado el modelo mínimo de la base de datos. Intuitivamente, un modelo mínimo denota los hechos positivos terminales seguros que pueden ser asociados con la base de datos, en este modelo, cualquier información incierta es interpretada como falsa. De hecho, una consulta a una base de datos Horn Δ bajo la asunción de mundo cerrado consiste en evaluar el query con respecto al modelo mínimo. El modelo mínimo es el modelo Herbrand canónico.

Bases de datos indefinidas, es decir, bases de datos que contienen información disyuntiva, no poseen la propiedad de modelo intersección, por ejemplo; consideremos la base de datos Δ={P(a) V P(b)}**¡Error! Marcador no definido.**. Esta base de datos admite tres modelos Herbrand, en el primero, la única literal terminal positiva que es verdadera es P(a), en el segundo solamente P(b) es verdadera y en el ultimo, ambos P(a) y P(b) son verdaderas. La intersección de esos tres modelos es el conjunto vacío, lo cual no es un modelo de Δ**¡Error! Marcador no definido.**.

Para competir con bases de datos indefinidas, Minker propone el siguiente método: Un modelo minimal de una base de datos Δ**¡Error! Marcador no definido.** se define ser un modelo Herbrand de Δ**¡Error! Marcador no definido.** que no contiene ningún modelo de Δ**¡Error! Marcador no definido.** como subconjunto propio. De acuerdo con esta definición, una base de datos indefinida puede admitir diversos modelos minimales. En el modelo anterior, {P(a)}**¡Error! Marcador no definido.** y {P(b)}**¡Error! Marcador no definido.** son modelos minimales de **¡Error! Marcador no definido.**Δ, mientras que **¡Error! Marcador no definido.**{P(a),P(b)} no lo es. Intuitivamente, una

fórmula que es verdadera en todos los modelos minimales de **¡Error! Marcador no definido.**Δ es verdadera con respecto a **¡Error! Marcador no definido.**Δ, y una fórmula que es verdadera en un modelo no minimal de **¡Error! Marcador no definido.**Δ es falsa con respecto a **¡Error! Marcador no definido.**Δ. El valor de verdad de una fórmula que es verdadera en alguno pero no en todos los modelos minimales de **¡Error! Marcador no definido.**Δ es desconocido.

Consideremos el conjunto de todos los modelos minimales de una base de datos de la intersección de esos modelos minimales es un conjunto de átomos terminales que denotamos **¡Error! Marcador no definido.**Γ(Δ). Por definición Γ(Δ) contienen los átomos terminales que son verdaderos en todos en todos los modelos minimales de **¡Error! Marcador no definido.**Δ. De una forma dual, para cada modelo minimal de **¡Error! Marcador no definido.**Δ, consideremos el conjunto de átomos terminales que no son verificados en ese modelo. La intersección de todos esos conjuntos es denotado Γ(Δ)**¡Error! Marcador no definido.**' y contiene, por definición los átomos terminales que son falsos en todos los modelos minimales de **¡Error! Marcador no definido.**Δ. Un átomo terminal será interpretado como verdadero, falso o indeterminado, de acuerdo a si pertenece a **¡Error! Marcador no definido.**Γ(Δ), a **¡Error! Marcador no definido.**Γ(Δ)' o a ninguno de esos conjuntos.

Bajo la asunción de mundo cerrado generalizada, un átomo terminal puede ser inferido de una base de datos **¡Error! Marcador no definido.**Δ, si y solo si pertenece a **¡Error! Marcador no definido.**Γ(Δ). Será permitido inferir la negación de un átomo terminal cuando este pertenece al conjunto **¡Error! Marcador no definido.**Γ(Δ)'.

APENDICE A: REFERENCIAS BIBLIOGRAFICAS

[Abdellatif y Zeronal, 1992] A. Abdellatif y A. Zeronal "Bases de Datos y Modelos de Datos" Informix; La Base de Datos relacional para UNIX (1992), pgs. 9-11

[Batini, Ceri y Navathe, 1994] "Diseño conceptual de Bases de Datos" Conceptos sobre el modelado de datos (1994), pgs. 17-63

[Booch, Rumbaugh y Jacobson, 1997] G. Booch, J. Rumbaugh e I. Jacobson "La importancia de modelar" El lenguaje unificado de modelado (1997), pgs. 4-7

[Delobel y Adiba, 1987] C. Delobel y M. Adiba "Bases de datos y sistemas relacionales" Los diferentes niveles de representación de una base de datos (1987)

[Date, 1985] C. J. Date "An introduction to Database Systems" Data models (1985), pgs 181-236

[Date, 1986] C. J. Date "Introducciòn a los sistemas de Bases de Datos" (1986)

[Date, 1990] C. J. Date "Sistemas de bases de datos" Conceptos básicos (1990), pgs.1-103

[Date, 1990] C. J. Date "Sistemas de bases de datos" El modelo relacional (1990), pgs. 241-284

[Date, 1990] C. J. Date "Sistemas de bases de datos" Modelado semántico (1990), pgs 568-599

[Date, 1990] C. J. Date "Sistemas de bases de datos" Sistemas distribuidos (1990), pgs. 605-624

[Date, 1990] C. J. Date "Sistemas de bases de datos" Sistemas orientados a objetos (1990), pgs. 671-692

[De Miguel y Piattini, 1993] Adoraciòn de Miguel y Mario Piattini "Modelos de Datos" Concepciòn y diseño de Bases de Datos (1993), pgs. 159-514

[De Miguel y Piattini, 1997] Adoración de Miguel y Mario Piattini "La estandarización de la arquitectura de los SGBD" Fundamentos y Modelos de Bases de Datos (1997), pgs. 57-60

[De Miguel y Piattini, 1997] Adoración de Miguel y Mario Piattini "Modelos de Datos" Fundamentos y Modelos de Bases de Datos (1997), pgs. 81-376

[Elmasri y Navate,] Fundamentals of Database Systems ( )

[Gardarin,] Bases de datos ( )

[Korth y Silberschatz, 1993] H. F. Korth y A. Silberschatz, "Modelos de datos" Fundamentos de Bases de Datos (1993), pgs. 6-12

[Kroenke, 1995] David M. Kroenke , "Componentes de un sistema de bases de datos" Procesamiento de bases de datos. Fundamentos, diseño e instrumentación (1995),pgs. 26-52

[Kroenke, 1995] David M. Kroenke , "Modelado de datos" Procesamiento de bases de datos. Fundamentos, diseño e instrumentación (1995),pgs. 55-77

[Lucas, 1993] Angel Lucas Gómez "Diseño y gestión de sistemas de bases de datos" Diseños de sistemas de bases de datos (1993), pgs. 17-83

[Lucas, 1993] Angel Lucas Gómez "Diseño y gestión de sistemas de bases de datos" Diseños de sistemas de bases de datos (1993), pgs. 245-279

[Tsichritzis y Lochovsky] Dionysius C. Tsichritzis y Frederick H. Lochovsky "Data Models" (1982)

[Ullman,] Principles of Database Systems ( )

[Vossen, 1991] Gottfried Vossen, "Foundations of the Relational Data Model" Data models, database languages and database management systems (1991),pgs.95-188

[Vossen, 1991] Gottfried Vossen, "Traditional (graphical) data models for the conceptual schema" Data models, database languages and database management systems (1991),pgs.33-94

APENDICE B: VOCABULARIO

- **Administración de bases de datos:** La función que se relaciona con el uso y control eficientes de una base de datos particular y sus aplicaciones relacionadas.
- **Administrador de datos:** La función, a nivel de toda una empresa, que se relaciona con el uso y control eficientes de los activos de datos de una organización. Puede ser una persona, pero con más frecuencia es un grupo. Las funciones específicas incluyen el establecimiento de normas de datos, políticas de datos y proporcionar un foro para la solución de conflictos.
- **Afinidad:** Sinónimo de tabla. Un arreglo en dos dimensiones que contiene entradas de valor único y no tiene hileras duplicadas. El significado de las columnas es igual en cada hilera. El orden de las hileras y columnas no es significativo.
- **Aplicación:** Un sistema comercial de computadora que procesa una parte de una base de datos para cumplir con las necesidades de información de un usuario. Se compone de menús, formas, reportes y programas de aplicaciones.
- **Apuntador:** Una dirección a una instancia de una estructura de datos. Con frecuencia la dirección de un registro está en un archivo direccionado de manera directa.
- **Árbol:** Un conjunto de registros, entidades u otras estructuras de datos en las cuales cada elemento posee como máximo un padre, excepto el elemento superior, el cual no tiene padre.
- **Arquitectura de base de datos cliente servidor:** La estructura de un sistema de cómputo en red, en el cual una computadora (que casi siempre es una micro), ejecuta servicios a nombre de las otras computadoras (micros también). Para un sistema de base de datos, la computadora servidora se denomina servidor de base de datos y procesa el DBMS, así como las computadoras clientes procesan los programas de aplicaciones. Todas las aplicaciones de la base de datos las ejecuta el servidor de la base de datos.
- **Arquitectura de recursos compartidos:** La estructura de una red de área local en la cual una microcomputadora ejecuta servicios de procesamientos de archivos para otras microcomputadoras. En una aplicación de base de datos, cada computadora de usuario contiene una copia del DBMS que solicita entradas/salidas el servidor de archivo. El servidor de archivo sólo procesa E/S de archivos; todas las actividades de bases de datos las procesa el DBMS en la computadora del usuario.
- **Atómicas:** Un conjunto de acciones que se terminan como una unidad. Se concluyen todas o ninguna de las acciones.
- **Atributo (campo o columna):** Cada una de las propiedades que caracterizan a una entidad. Unidad de datos individual de menor tamaño y con significado pleno. Los campos pueden agruparse para constituir otros más complejos (ej. fecha de nacimiento). Pueden ser de naturaleza numérica o alfabética, en general. Tienen un tamaño asociado, medido físicamente en *bytes*.
- **Base de datos:** Es un conjunto integrado de datos junto con una serie de aplicaciones para su manejo accesibles simultáneamente por diferentes usuarios y programas.
- **Base de datos Abierta:** Si se ofrece comercial o gratuitamente al mercado o público en general que pueda estar interesado. Por ejemplo, una base de datos de legislación, estadísticas varias, productos comerciales, etc.
- **Base de datos Autónoma:** Si se encuentra en un soporte independiente, fácilmente manejable e intercambiable, y puede ser consultada en el ordenador del propio usuario.

Éste es el caso, por ejemplo, de las bases de datos que actualmente se están ofreciendo en soporte CD-ROM. Es interesante observar que el usuario no tiene la "propiedad", en el sentido jurídico, de la base de datos.

- **Base de datos Centralizada:** Todos los datos están físicamente almacenados en el mismo sistema informático y bajo un control unitario. Los datos pueden estar compartidos por múltiples aplicaciones y usuarios.
- **Base de datos Cerrada:** Si la base de datos es desarrollada por una persona física o jurídica, ya sea privada o pública, para su uso interno. Por ejemplo, una base de datos de clientes de un gran almacén, de contribuyentes, etc.
- **Base de datos Distribuida:** Los datos están almacenados en varios sistemas informáticos geográficamente repartidos y conectados mediante una red telemática. Problema de la localización de los datos. La administración de la base de datos puede realizarse en varios lugares distintos y por personas distintas. Toda esta problemática debe ser transparente a los usuarios, el cual no necesita saber dónde están realmentte almacenados los datos a los que accede.
- **Base de datos lógica(LBD):** En DL/I, el conjunto de todos los registros lógicos de base de datos que contiene la base de datos.
- **Base de datos Online:** *S*i su soporte físico es la memoria de un ordenador de servicios y es consultada a distancia, mediante comunicación telemática, desde un equipo informático terminal. De esta forma, el usuario se conecta al ordenador que contiene la información y realiza las operaciones que desee, desconectándose al final. Sólo utiliza el ordenador de la base de datos el tiempo que tarda en hacer la consulta, compartiendo tiempo y ordenador con otros múltiples usuarios que también pueden estar accediendo.
- **Base de datos relacional:** Una base de datos formada por afinidades, tal base de datos está estructurada de acuerdo con los principios de normalización, aunque en la práctica las bases de datos relacionales contienen afinidades con renglones duplicados. La mayoría de los productos DBMS incluyen una característica que quita las hileras duplicadas cuando es necesario y conveniente. Tal eliminación no se realiza como rutina porque es costosa y consume mucho tiempo.
- **Buffer:** Un área de memoria empleada para contener datos. Para una lectura, los datos se leen desde un dispositivo de almacenamiento hacia dentro del buffer. Para una escritura, los datos se escriben desde el buffer hacia el almacenamiento.
- **Búsqueda:** El proceso de obtener datos relacionados usando del valor de una clave ajena.
- **Cardinalidad (número de filas):** Número de instancias de una entidad en una relación.
- **Cardinalidad máxima:** La cantidad máxima de valores que puede poseer un atributo dentro de un objeto semántico. En una relación entre tablas, la cantidad máxima de hileras a las que puede hacer referencia una hilera de una tabla en otra tabla.
- **Cardinalidad mínima:** La cantidad mínima de valores que puede tener un atributo dentro de un objeto semántico. En una relación entre tablas, la cantidad de hileras con las cuales se puede relacionar una hilera de una tabla con las de otra tabla.
- **Celda:** Es el término genérico con que se refiere a una pista, un cilindro, un módulo, o cualquier área de almacenamiento. Es el valor de un atributo en determinada ocurrencia.
- **Clase de entidad:**Un conjunto de entidades del mismo tipo.
- **Clase de objeto:** En la programación orientada a objetos, un conjunto de objetos con una estructura común.
- **Clave:** Un grupo de uno o más atributos que identifican una hilera única en una afinidad. Como las afinidades no pueden tener hileras duplicadas, cada una debe tener al menos una clave, la cual es la combinación de todos los atributos en la afinidad. En ocasiones una clave se denomina clave lógica.
- **Clave ajena:** Un atributo que es una clave en una o más afinidades además de aquella en donde aparece.
- **Clave candidata:** Un atributo o grupo de atributos que identifican una clave única en una afinidad. Una de las claves candidatas se elige como la clave primaria.
- **Clave compuesta:** Una clave con más de un atributo.

- **Clave física:** Una columna que posee un índice u otra estructura de datos para ella. Sinónimo de índice. Tales estructuras se crean para mejorar la búsqueda y el ordenamiento de los valores de columnas.
- **Clave lógica:** Una o más columnas que determinan de manera singular una hilera de una tabla o un registro de un archivo. Sinónimo de clave.
- **Clave primaria (identificador único):** Conjunto de uno o más atributos que identifican de modo completo una instancia de una entidad.
- **Cliente/Servidor:** Método de distribución de información o de archivos en el cual la agrupación central, servidor, almacena los archivos y los hace disponibles para solicitudes de aplicaciones cliente.
- **Columna:** Un grupo lógico de bytes en una hilera de una afinidad o tabla. El significado de una columna es el mismo para cada hilera de la afinidad.
- **Constructor de objetos:** En la programación orientada a objetos, una función que crea un objeto.
- **Cursor:** Es el apuntador a un conjunto de registros u ocurrencias con determinado criterio en la base de datos.
- **Datos de base de datos:** La parte de una base de datos que contiene información de interés y utilidad para los usuarios finales de la aplicación.
- **Datos duplicados:** En una base de datos distribuida, la información que se almacena en dos o más computadoras.
- **DBMS(*Database Management System*):** Sistema de administración de base de datos. Un conjunto de programas empleado para definir, administrar y procesar la base de datos y sus aplicaciones.
- **DDBMS(*Distributed Database Management System*):** Sistema distribuido de administración de bases de datos.
- **Desarrollador/Programador:** Se encarga tanto del desarrollo de aplicaciones, como de las funciones de interface con los usuarios (menús, ayuda, etc).
- **Descripción de base de datos:** En DL/I, una estructura de datos que describe la estructura de un registro de base de datos.
- **Destructor de objetos:** En la programación orientada a objetos, una función que destruye un objeto.
- **Diagrama de estructura de datos:** Un despliegue gráfico de tablas(archivos) y sus relaciones. Las tablas se muestran en rectángulos y las relaciones con líneas. Una relación a muchos se señala con una ramificación al extremo de la línea; una relación opcional se señala mediante una elipse; una relación obligatoria se representa con marcas transversales.
- **Diagrama de flujo de datos:** Un despliegue gráfico que preparan quienes desarrollan el sistema para mostrar los procesos de negocios y las interfaces de datos. También presenta el flujo del sistema desde la perspectiva de los datos.
- **Diagrama entidad-relación:** Una representación gráfica de las entidades y sus relaciones. Por lo general, las entidades se muestran en cuadrados o rectángulos y las relaciones en diamantes. La cardinalidad de la relación se muestra dentro del diamante.
- **Diagrama de objeto:** Un rectángulo orientado de manera vertical y que representa la estructura de un objeto semámtico.
- **Diccionario de datos:** Es un lugar en donde se almacena la información relativa a la estructura de la base de datos. En otras palabras, es la colección de toda la información para mentener una base de datos como campos accesados, índices usados, usuarios activos, etc.
- **DL/I(*Data Language I*):** Lenguaje de datos I. Un modelo de datos desarrollado por IBM afines de los años 60 para definir y procesar bases de datos gráficas. El IMS producto de DBMS de IBM se basa en DL/I. Aunque IMS todavía se emplea mucho en macrocomputadoras, tales bases de datos están siendo sustituidas con modelos relacionales.
- **DNS:** conjunto de bases de datos distribuidos que mantiene la correlación entre las relaciones nombre de dominio y las direcciones numéricas.

- **Dominio (conjunto de valores legales):** Define la colección de valores de donde uno o más atributos obtienen sus valores reales.
- **Elemento de datos:** En el contexto del modelo relacional, un sinónimo de atributo.
- **Entidad:** Algo de importancia para un usuario que necesita representarse en una base de datos. En un modelo entidad-relación, las entidades están limitadas a cosas que pueden representarse por medio de una tabla única.
- **Entidad débil:** En un modelo entidad-relación, una entidad cuya existencia en la base de datos, depende de la existencia de otra.
- **Entidad dependiente de existencia:** Igual que una entidad débil. Una entidad que no puede aparecer en la base de datos a menos que también aparezca una ocurrencia de una o más de otras entidades. Una subclase de las entidades dependientes de existencia son las entidades dependientes ID.
- **Entidad dependiente de ID:** Una entidad que no puede existir lógicamente sin la existencia de otra entidad. La entidad dependiente de ID siempre contiene la clave de la entidad de la cual depende. Tales entidades son un subconjunto de una entidad débil.
- **Entidad fuerte:** En un modelo entidad-relación, cualquier entidad cuya existencia en la base de datos no dependa de la existencia de cualquier otra entidad.
- **Esquema:** Indica la visión lógica global de la BD. Útil para el diseñado del sistema y el administrador del sistema.
- **Esquema relacional:** Un conjunto de afinidades con limitantes de interafinidad.
- **Estructura encapsulada:** Una parte de un objeto que no es visible para otros objetos.
- **Fichero**: conjunto de ocurrencias de un mismo tipo de registro.
- **Forma**: Un documento en papel empleado en un sistema comercial para registrar datos, por lo general relacionados con una transacción. Las formas se analizan en el proceso de desarrollar un modelo de datos.
- **Gestor**: Componente *software* (programa) encargado de la interface entre las peticiones de los usuarios de la base de datos y los propios datos en sí. Interacción con el sistema operativo.
- **Gestor/Administrador (DBA: *Data Base Administrator*):** Se encarga de la creación, manipulación y modificación de la estructura de la base de datos, de las funciones relativas a la seguridad e integridad de los datos, del control sobre los usuarios y concesión de autorizaciones o permisos a los mismos, así como de la realización de estadísticas.
- **Grado (número de columnas):** Número de atributos asociados a una entidad.
- **Grupo compuesto:** Un grupo de atributos en un objeto semántico que posee valores múltiples y no contiene otros atributos de objeto.
- **Herencia:** Una característica de sistemas orientados a objetos cuyos atributos son obtenidos de objetos padres.
- **Hermano:** Un registro o nodo que tiene el mismo padre que otro registro o nodo.
- **Hijo:** Una hilera, registro o nodo en el lado muchos en una relación uno-a-muchos.
- **Hilera:** Un grupo de columnas en una tabla. Todas las columnas en una hilera pertenecen a la misma entidad. Una hilera es igual que un tuple y un registro.
- **Huérfana:** Cualquier hilera(registro) que no tenga padre en una relación uno-a-muchos obligatoria.
- **Identificador de grupo:** Un atributo que identifica una ocurrencia única de un grupo dentro de un objeto semántico u otro grupo.
- **Identificador de objeto:** Un atributo que se emplea para especificar la ocurrencia de un objeto. Los identificadores de objetos pueden ser únicos, lo que significa que identifican a una (y sólo una) ocurrencia o, una no única, lo que significa que determinan exactamente una ocurrencia del objeto.
- **IMS(*Information Management System*):** Sistema de administración de información: Un sistema de procesamiento de transacciones desarrollado y concesionado por IBM. Incluye IMS/DC, un programa de control de comunicaciones e IMS/DB, un DBMS que pone en práctica el modelo de datos DL/I.
- **Independencia de los datos**: inmunidad de las aplicaciones a los cambios de la estructura de almacenamiento de los datos y su estrategia de acceso.

- **Índice**: Datos significativos empleados para mejorar el acceso y ordenar el desempeño. Pueden esquematizarse índices para una columna o grupo de columnas. Son muy útiles en columnas usadas para saltos de control en los reportes y en las columnas usadas para especificar condiciones en uniones.
- **Integridad**: ¿son veraces y consistentes los datos? ¿verifican las restricciones que pudieran haberse definido sobre ellos? ¿son correctos los datos después de un fallo de *hardware* o *software*? Ejemplos:
  - NIF de una persona (comprobación de la letra).
  - Fecha de nacimiento de una persona (no todos los valores de días, meses y años son correctos).
  - Edad de una persona (valor no negativo).
  - "No puede existir en la base de datos ninguna persona cuyos ingresos medios en los últimos 5 años no excedan una determinada cantidad", etc.
- **Items:** Sinónimo de campo o atributo.
- **Jerarquía de generalización:** Un conjunto de objetos o entidades del mismo tipo lógico, ordenados en una jerarquía de subtipos lógicos. Los subtipos heredan características de sus supertipos.
- **Lenguaje de consulta/actualización:** Un lenguaje que pueden utilizar los usuarios finales para consultar la base de datos y para realizar cambios en la información que contiene.
- **Lenguaje de control de datos (DCL: *Data Control Language*):** encargado del control y seguridad de los datos (privilegios y modos de acceso, etc).
- **Lenguaje de definición de datos (DDL: *Data Definition Language*):** Sencillo lenguaje artificial para definir y describir los objetos de la base de datos, su estructura, relaciones y restricciones. En la práctica puede consistir en un subconjunto de instrucciones de otro lenguaje informático.
- **Lenguaje de definición del almacenamiento de los datos (DSDL: *Data Storage Definition Language*):** permite especificar características físicas de la base de datos (volúmenes y archivos donde van a ser almacenados los datos, etc).
- **Lenguaje de manipulación de datos (DML: *Data Manipulation Language):*** Lenguaje artificial de cierta complejidad que permite el manejo y procesamiento del contenido de la base de datos. En la práctica puede consistir en un subconjunto de instrucciones de otro lenguaje informático. Las aplicaciones que trabajan sobre la base de datos se programan en un lenguaje de programación (C, Cobol, ...) insertando en el código fuente sentencias del DML. Al utilizar un DML se deben especificar los datos que serán afectados por las sentencias del lenguaje. Un DML puede tener o no procedimientos, según sea necesario especificar además cónmo deben obtenerse esos datos. Los DML con procedimientos tienen sentencias de control de flujo como bucles o condicionales. Los DML sin procedimientos son conocidos también como declarativos.
- **Lenguaje orientado a transformaciones:** Un sublenguaje de datos, tal como SQL, que proporciona comandos y opciones para transformar un conjunto de afinidades en una afinidad nueva.
- **Limitación:** Una regla relacionada con los valores de atributos permitidos cuya veracidad puede evaluarse. Una limitación no incluye las reglas dinámicas.
- **Limitación de integridad referencial:** La condición en una base de datos en la cual se satisfacen todas las limitantes de interafinidad.
- **Limitación de interafinidad:** Una restricción que requiere que el valor de un atributo en una hilera de una afinidad sea igual al valor de un atributo encontrado en otra afinidad.
- **Lista numerada:** Una lista de valores permitidos para un dominio, atributo o columna.
- **Máquina de base de datos:** Una CPU de propósito especial diseñada específicamente para procesar una base de datos. En ocasiones se denomina una máquina de respaldo, porque reside entre el sistema operativo y los datos. Sin embargo, la máquina de base de datos no ha tenido éxito comercial y actualmente se usa muy pocas veces.
- **Máquina de DBMS:** Un subsistema de un DBMS que procesa solicitudes lógicas de E/S de otros subsistemas de DBMS y somete peticiones físicas de E/S el sistema operativo.
- **Máscara:** Un formato usado cuando se presentan datos en una forma o reporte.

- **Metadatos:** Información referente a la estructura de los datos en una base de datos, almacenada en el diccionario de datos. Se emplean metadatos para describir tablas, columnas, limitaciones, índices y demás.
- **Metadatos de aplicación:** La información de un diccionario de datos relacionada con la estructura y contenido de los menús, formas y reportes de una aplicación.
- **Método:** Un programa anexado a un objeto de programación orientada a objetos. Un método puede ser heredado por los objetos de programación orientada a objetos de nivel inferior.
- **Miembro:** En el modelo CODASYL DBTG, un tipo de registro que está en el lado muchos de una relación uno-a-muchos o una relación de conjunto.
- **Modelo de datos:** Un modelo de los requisitos de datos de los usuarios, expresado en términos del modelo entidad-relación o el modelo de objeto semántico. En ocasiones se lo conoce como modelo de datos de los usuarios.
- **Modelo de datos de red:** Un modelo de datos que soporta relaciones de red simples. El CODASYL DBTG, que soporta relaciones de red simples pero no complejas, es el modelo de red más importante.
- **Modelo de datos relacional:** Un modelo de datos en el cual la información se almacena en afinidades y relaciones entre hileras representadas por valores de datos.
- **Modelo entidad-relación:** Los esquemas y convenciones empleados para crear un modelo de los datos de los usuarios. Las cosas en el mundo de los usuarios se representan con entidades y las asociaciones entre estas cosas se representan con relaciones. Normalmente los resultados se documentan en un diagrama entidad-relación.
- **Modelo jerárquico de datos:** Un modelo de datos que representa todas las relaciones que emplean jerarquías o árboles. Las estructuras de red deben descomponerse en árboles antes de que pueda representarse mediante un modelo jerárquico de datos. DL/I es el único modelo de datos jerárquico que sobrevive.
- **Modelo semántico de objetos:** Los esquemas y convenciones empleados para crear un modelo de los datos del usuario. Las cosas en el mundo de los usuarios se representan con objetos semánticos (denominados en ocasiones objetos). Las relaciones se modelan en los objetos y los resultados se documentan en diagramas de objetos.
- **N-M:** Una abreviatura para una relación muchos-a-muchos entre las hileras de dos tablas.
- **Nodo:** Una entidad en un árbol. También se refiere a una computadora en un sistema de procesamiento distribuido.
- **Nodo de base de datos:** Una computadora que procesa un administrador de base de datos y su(s) base(s) de datos relacionada(s).
- **Objeto:** Un objeto semántico o una estructura en un programa orientado a objetos que contiene una estructura de datos encapsulados y métodos de datos. Tales objetos se ordenan en una jerarquía para que puedan heredar métodos de sus padres.
- **Objeto combinado:** Un objeto que contiene al menos otro objeto.
- **Objeto compuesto:** Un objeto con al menos un atributo o grupo de atributos de valores múltiples. Se denomina compuesto porque la clave de la afinidad que representa el atributo o grupo de valores múltiples es una clave compuesta.
- **Objeto de asociación:** Un objeto que representa la combinación de al menos otros dos objetos y que contiene datos sobre tal combinación. Con frecuencia se emplea en aplicaciones para contraer y para asignación.
- **Objeto de generalización:** Un objeto que contiene objetos subtipos. El objeto de generalización y sus subtipos tienen todos la misma clave. Los subtipos heredan atributos del objeto de generalización. También se denomina un objeto de supertipo.
- **Objeto simple:** Un objeto que no contiene atributos de repetición, ni atributos de objeto.
- **Ocurrencia:** Es el conjunto de registros que cumplen determinado criterio.
- **Ocurrencia de entidad:** Una ocurrencia particular de una entidad.
- **Ocurrencia de objeto:** El caso de un objeto semántico particular.
- **Padre:** Una hilera, registro o nodo que está en el lado uno de una relación uno-a-muchos.

- **Persistencia de objetos:** En la programación orientada a objetos, la característica de que un objeto puede guardarse en una memoria no volátil, tal como un disco. Los objetos persistentes existen entre las ejecuciones de un programa.
- **PL/I (*Programming Language I*):** Lenguaje de programación I. Un lenguaje de programación de tercera generación comercializado por IBM.
- **Polimorfismo:** La situación en la cual puede usarse un nombre para invocar diferentes funcionalidades, dependiendo del objeto que invoca el nombre.
- **Privilegios:** Son las cualidades que se le asignan a cada uno de los atributos de una base de datos para su acceso a él.
- **Programa:** En un sistema distribuido de base de datos, una secuencia ordenada de solicitudes de datos.
- **Programa de aplicación:** Un programa que dearrolla un usuario para procesar una base de datos. Puede escribirse en un lenguaje normal de procedimientos, tal como COBOL; C o BASIC o en un lenguaje único para el DBMS.
- **Propiedad:** Lo mismo que atributo.
- **Propietario:** En el modelo CODASYL DBTG, un tipo de registro que está en el lado uno en una relación uno-a-muchos o de conjunto. En la administración de datos, el departamento u otra unidad de la organización a cargo de la administración de un elemento de datos particular. Un propietario también puede denominarse un proponente de datos.
- **Propietario de datos:** Autor de datos.
- **Query:** Son consultas que se realizan a la base de datos.
- **Raíz:** El registro, hilera o nodo superior en un árbol. Una raíz no tiene padre.
- **Rama:** El subelemento de un árbol que puede consistir en uno o múltiples nodos.
- **Red compleja:** Un conjunto de entidades, objetos o afinidades y sus correspondencias, de las cuales al menos una es compleja (muchos-a-muchos).
- **Red simple:** Un conjunto de tres afinidades y dos relaciones en la cual una de las afinidades,R ,tiene una relación muchos-a-uno con las otras dos afinidades. Las hileras en R tienen dos padres y éstos son de tipos diferentes.
- **Redundancia**: Un mismo dato almacenado varias veces. Problemas: 1) Gasto de capacidad de almacenamiento 2) Posibilidad de información inconsistente. Ejemplo: edad de una persona. Las bases de datos pretenden reducir en lo posible la existencia de redundancia en los datos almacenados.
- **Registro (segmento o tupla):** Es una colección de items. Sinónimo de fila. Conjunto de campos intrínsecamente relacionados mediante una significación común a una entidad. Suele constituir la unidad básica de acceso a la base de datos (contenido que se puede almacenar o recuperar en un solo acceso).
- **Registro de base de datos:** En DL/I, un arreglo jerárquico de segmentos de datos.
- **Registro especial:** En el modelo CODASYL DBTG, un indicador de algún aspecto del estado de la base de datos.
- **Registro físico de base de datos(PDBR):** En DL/I, una jerarquía de segmentos almacenados en archivos de base de datos.
- **Registro lógico de base de datos(LDBR):** En DL/I, una jerarquía de segmentos según los percibe un programa de aplicación. Tal estructura puede o no existir de un modo físico y puede tomarse de otras estructuras usando apuntadores y otros datos significativos.
- **Relación (tabla):** Elemento básico del modelo relacional que se asocia a una entidad.
- **Relación binaria:** Una relación entre dos entidades o tablas.
- **Relación ES-UN:** Una relación entre dos entidades u objetos del mismo tipo lógico.
- **Relación recursiva:** Una relación entre entidades, objetos o hileras del mismo tipo.
- **Relación TIENE-UN:** Una relación entre dos entidades u objetos que son de diferentes tipos lógicos.
- **Reporte:** Una extracción de los datos de una base de datos. Los reportes pueden imprimirse, desplegarse en una pantalla de computadora o almacenarse como un archivo. Un reporte es parte de una aplicación de base de datos.

- **Segmento:** En DL/I, un conjunto de campos que forman un nodo en un registro de base de datos.
- **Servidor de archivos:** En una red de área local, una microcomputadora que contiene un archivo que procesa en representación de las otras microcomputadoras de la red. El término servidor de archivo se usa para la arquitectura de recursos compartidos.
- **Servidor de base de datos:** En una red de área local con arquitectura de base de datos cliente-servidor, la microcomputadora que ejecuta el DBMS y procesa acciones sobre la base de datos en nombre de sus computadoras clientes.
- **Set:** Es la relación definida entre dos miembros o entidades.
- **Sistema cliente servidor:** Un sistema de dos o más computadoras en el cual al menos una de ellas proporciona servicios para otra o para las demás. Los servicios pueden ser de base de datos, de comunicaciones, de impresora o algunas otras funciones.
- **Sistema de administración de base de datos distribuida(DDBMS):** En una base de datos distribuida, es el conjunto de transacciones y administradores de base de datos distribuidos en todas las computadoras.
- **Sistema de gestión de bases de datos (SGBD):** Programas (*software* de ordenador) que posibilitan la existencia y utilización de las bases de datos. En inglés: DBMS (*Data Base Management System*).
- **Sistema distribuido:** Un sistema en el cual los programas de aplicación de una base de datos se procesan en dos o más computadoras.
- **Sistema distribuido de base de datos:** Un sistema distribuido en el cual una base de datos, o parte de una base de datos se distribuye a través de dos o más computadoras.
- **SQL(Stuctured Query Language):** Lenguaje de consulta estructurada: Un lenguaje para definir la estructura y procesamiento de una base de datos relacional. Se emplea como un lenguaje de consulta único o puede incorporarse en programas de aplicación. El American National Standards Institute acepta un SQL como una norma en el país (E.E.U.U.). Fue desarrollado por IBM.
- **Subesquema o Vista:** Visión lógica de los datos relacionados con una aplicación. Útil para programadores de aplicaciones y usuarios finales.
- **Sublenguaje de datos:** Un lenguaje para definir y procesar una base de datos que se pretende incorporar en programas escritos en otro lenguaje- en la mayoría de los casos, un lenguaje de procedimientos, tal como COBOL, C o BASIC. Un sublenguaje de datos es un lenguaje de programación incompleto, dado que sólo contiene esquemas para acceder datos.
- **Subsistema de diccionario de datos y administración de base de datos:** Un conjunto de programas en el DBMS, empleado para acceder el diccionario de datos y llevar a cabo funciones de administración de base de datos, tales como mantener contraseñas y realizar respaldos y recuperaciones.
- **Subtipos:** En jerarquías de generalización, una entidad u objeto que es una subespecie o categoría de un tipo de nivel superior.
- **Supertipo:** En jerarquías de generalización, una entidad u objeto que contiene subtipos.
- **Tabla:** Estructura que permite almacenar una entidad.
- **Tabla base:** En puestas en práctica relacionales, la tabla a partir de la cual se definen las vistas relacionales.
- **Tupla (fila o registro):** Cada una de las instancias de una entidad.
- **1-N:** Una abreviatura para una relación uno-a-muchos entre las hileras de dos tablas.
- **Usuario:** Cualquiera que requiere los servicios de los productos de un sistema de computación.
- **Valor computado:** Una columna de una tabla que se calcula a partir de los valores de otra columna. Los valores no se almacenan pero sí se computan cuando se van a exhibir.
- **Valor nulo:** Un valor que se desconoce o no es aplicable. Un valor nulo no es igual que un cero o un espacio en blanco, aunque en la mayoría de los productos DBMS comerciales, los valores nulos se representan mediante ceros o espacios en blanco.
- **Variable anfitriona:** Una variable de un programa de aplicación dentro de la cual un DBMS coloca un valor de la base de datos.

- **Vista:** El subconjunto de una base de datos que puede procesar una aplicación o una vista de un objeto.
- **Vista de aplicación:** La parte de una base de datos procesada por un programa de aplicación particular.
- **Vista de objeto:** La parte de un objeto semántico visible para una aplicación particular. Una vista consiste en el nombre del objeto semántico más una lista de los atributos visibles en tal vista.
- **Vista de objeto semántico:** La parte de un objeto semántico visible en una forma o reporte.
- **Vista de usuario:** Una vista particular del usuario de una base de datos.

APENDICE C: TRANSFORMACION DEL MODELO E/R AL RELACIONAL

Tenemos tres principios :

1) Toda entidad se transforma en una tabla.
2) Toda interrrelación M :N se transforma en una tabla.
3) Toda interrrelación del tipo 1 :N se traduce en el fenómeno de propagación de clave o se crea una nueva tabla.

Reglas de transformación del Modelo Básico.

- Transformación de Dominios .- Hay sistemas relacionales que no lo permiten.
- Transformación de entidades .- Toda entidad se transforma en una tabla.
- Transformación de atributos .- Cada atributo de una entidad se transforma en una columna de la tabla. AIP → Clave primaria de la relación.
- Transformación interrelaciones .- Según tipo :

  - N :M → Se transforma en una nueva tabla cuyos atributos son la concatenación de los AIP de las entidades que asocian.
    - Cada uno de estos atributos es clave ajena de la tabla donde estos atributos son clave primaria.
    - CLAVE AJENA (<Nombre columna>)
      REFERENCIA TABLA (<Nombre tabla>)

TABLA ESCRIBE (Cod_Libro, Nombre)
CLAVE PRIMARIA (Cod_Libro, Nombre)
CLAVE AJENA (Cod_Libro)
REFERENCIA TABLA LIBRO
EN BORRADO : CASCADA
EN ACTUALIZACION : CASCADA
CLAVE AJENA (Nombre)
REFERENCIA TABLA AUTOR
EN BORRADO : NULOS
EN ACTUALIZACION : CASCADA

  - 1 :N → Dos soluciones :

(A) Propagación :

- Propaga AIP de entidad con cardinalidad Maxima 1
- Perdida de semantica

(B) Considerarla como una relación M :_N

¿Cuándo A y cuándo B ?

- Si la relación al final resultará una M :N mejor B
- Si hay atributos propios en la relación B
- Si el nº de ocurrencias interrlacionadas de la entidad que produce la clave es pequeña o bien la cardinalidad minima es 0 (Nulos) → B

- Perdida de semántica en cardinalidades :

a) Cardinalidad máxima (Predicados)

1 :N) → Máximo es conocido (En tablas se permite limitar el número de filas)

b) Cardinalidad mínima (Tabla distinta para la relación) → Predicado
c) Cardinalidad mínima → Propagando claves (Permitir NULOS en clave ajena)

- Interrelaciones 1 :1
  - Caso particular de las relaciones 1 : y M :N
  - No existe una regla fija para su transformación , pudiéndose crear una tabla nueva o propagar la clave. En este último caso, la propagación de la clave puede realizarse en ambas direcciones.
  - Criterios para la selección de una solución :
    . Cardinalidad mínima.
    . Semántica
    . Eficiencia (Nulos o no Nulos / Accesos más frecuentes)

En definitiva:

- Si la cardinalidad es (0,1) en ambas entidades la interrelación se transforma en una nueva tabla.
- Si la cardinalidad de una de ellas es (0,1) y la otra es (1,1) conviene propagarla la clave de la entidad con cardinalidad (1,1) a la tabla resultante de la entidad con cardinalidad (0,1)
- Si en ambas la cardinalidad es (1,1) se puede propagar clave en ambos sentidos o incluso propagar ambas claves.

La solución, con ejemplos, en el modelo Relacional sería:

Ej 1

EMPLEADO (Cod_emp, ...)

DEPARTAMENTO (Cod_Dep, ..., Cod_emp)

Ej 2.-

MATRIMONIO (Cod_H, Cod_ M)

HOMBRE (Cod_H , ...)
MUJER (Cod_M, ...)

Caso 1 : 1 → Cardinalidades en ambos (1,1) , (1,1)

- Propagar las dos entidades.
  - Accesos más frecuentes (Primar estos accesos)

Transformación de los atributos de interrelaciones.

- Si la interrelación se transforma en una relación , todos sus atributos pasan a ser columnas de la relación .
- Si se transforma mediante propagación de clave, sus atributos migran junto con la clave de la relación correspondiente, aunque suele ser mejor crear una nueva tabla para realizar esta migración de la interrelación con sus atributos.

Transformación de restricciones de usuario.

- Se añaden como predicados al definir las tablas.
- Tipos :
  - Comprobación del rango de valores de los atributos.
  - Especificar todos los valores posibles de un atributo
  - Comprobar un predicado. (Ej : sobre valor de un atributo).
- Notación :
  - <atributo> RANGO ENTRE < Intervalo (Valores)>
  - <Atributo> EN < conjunto de valores>
  - COMPROBAR (<condición>, <Atributo>)

TRANSFORMACION DE REGLAS DEL MODELO EXTENDIDO.

Transformación de dependencias en identificación y en existencia.

(*) No reflejado por el modelo realcional

- Utilizar el mecanismo de propagación de clave, creando una clave ajena con NULOS no permitidos en la tabla de la entidad dependiente, obligando a la modificación y borrado en cascada.
- si la dependencia es en identificación la clave primaria de la tabla de al entidad débil, debe estar formada por la concatenación de las claves de las dos entidades que participan en la interrelación.

SOLUCION (Ej)

TABLA LIBRO (Cod_ Libro, ...)
CLAVE RPIMARIA (Cod_Libro)

TABLA EJEMPLAR (Cod_Libro ; Cod _ Ejem, ...)
CLAVE PRIMARIA (Cod_Libro, Cod_ejem)
CLAVE AJENA (Cod_Libro)
REFERENCIA A TABLA LIBRO
EN BORRADO : CASCADA
EN MODIFICACION : CASCADA

Relaciones exclusivas

En cada caso según su semántica.

Edita 1 EXCLUYE Edita 2

Propagando claves :

TABLA LIBRO (Cod_Libro, CodEd, Cod_Univ.....)
CLAVE PRIMARIA (Cod_libro)
CLAVE AJENA (CodEd)
REFERNCIA A TABLA EDITORIAL
EN BORRADO : CASCADA
EN ACTUALIZACION : CASCADA

CLAVE AJENA(Cod_Univ)
REFERNCIA TABLA UNIVERSIDAD

EN ACTUALIZACION Y BORRADO : CASCADA

COMPROBACION [(CodEd == NULO) y (Cod_univ != NULO)] o
[(CodEd != NULO) y(COD_univ == NULO)]

Transformación de TIPOS y SUBTIPOS:

- No está recogido directamente en el modelo Relacional.
- Se pierde semántica.
- Opciones :

A) Una tabla que englobe los atributos del supertipo y de los subtipos.

- Cuando los subtipos se diferencian en muy pocos atributos
- Las interrelaciones que les asocian con otras entidades sean las mismas parta todos los subtipos.
- Se debe añadir a la tabla un atributo adicional. (DISCRIMINANTE).

B) Una sola tabla para el Supertipo y tantas tablas como subtipos haya con sus atributos correspondientes.

- Cuando existen atributos diferentes en los subtipos y se quieren mantener los atributos comunes en una tabla.

C) Tablas distintas para cada subtipo que tengan además los atributos comunes

- Cuando existan atributos distintos entre los subtipos
- Cuando los acceso realizados a los datos de los subtipos siempre afectan a los atributos comunes.

Evaluación de eficiencia.

A) Acceso a una fial que refleje todos los datos de una entidad es mucho más rapido.
B) Mejor desde punto de vista semantico.
C) Aumenta eficiencia en determinadas consultas. Se produce redundancia y perdida de semantica.

Dimensión Temporal.

OBS → Puede ocurrir.

- Atributo temporal forma parte de las claves. (Fecha diferencia una ocurrencia de otras)

Atributos derivados.

- No hay representación directa.
- Opciones :

a) Considerarlo una columna más :

- Procedimientos para recalcular valores cada vez que se modifique la tabla.

b) Disparadores .- Sin columnas en las tablas.→ Procedimientos que calculan los datos en el momento en que se piden.