Unidad 1:
Gestión de datos

2º de ASI - Curso 2006-07

Autor: Jorge Sánchez Asenjo (año 2006)

http://www.jorgesanchez.net

email:info@jorgesanchez.net

[1.1] introducción

[1.1.1] sistemas gestores de bases de datos

la necesidad de gestionar datos

En el mundo actual existe una cada vez mayor demanda de datos. Esta demanda siempre ha sido patente en empresas y sociedades, pero en estos años la demanda todavía de ha disparado más debido al acceso multitudinario a las redes integradas en Internet y a la aparición de pequeños dispositivos (móviles y PDA) que también requieren esa información.

En informática se conoce como **dato** a cualquier elemento informativo que tenga relevancia para un usuario. Desde su nacimiento, la informática se ha encargado de proporcionar herramientas que faciliten la gestión de los datos.

Antes de la aparición de las aplicaciones informáticas, las empresas tenían como únicas herramientas de gestión de datos a los cajones, carpetas y fichas en las que se almacenaban los datos. En este proceso manual, el tipo requerido para manipular estos datos era enorme. Sin embargo el proceso de aprendizaje era relativamente sencillo ya que se usaban elementos que el usuario reconocía perfectamente.

Por esa razón, la informática ha adaptado sus herramientas para que los elementos que el usuario maneja en el ordenador se parezcan a los que utilizaba manualmente. Por eso en informática se sigue hablado de ficheros, formularios, carpetas, directorios,....

componentes de un sistema de información electrónico

En el caso de una gestión electrónica de la información (lo que actualmente se considera un sistema de información), los componentes son:

- **Datos.** Se trata de la información relevante que almacena y gestiona el sistema de información
- **Hardware.** Equipamiento físico que se utiliza para gestionar los datos
- **Software.** Aplicaciones que permiten el funcionamiento adecuado del sistema
- **Recursos humanos.** Personal que maneja el sistema de información

[1.1.2] tipos de sistemas de información

En la evolución de los sistemas de información ha habido dos puntos determinantes, que han formado los dos tipos fundamentales de sistemas de información.

sistemas de información orientados al proceso

En estos sistemas de información se crean diversas aplicaciones (software) para gestionar diferentes aspectos del sistema. Cada aplicación realiza unas determinadas operaciones. Los datos de dichas aplicaciones se almacenan en archivos digitales dentro de las unidades de almacenamiento del ordenador (a veces en archivos binarios, o en hojas de cálculo, o incluso en archivos de texto).

Cada programa almacena y utiliza sus propios datos de forma un tanto caótica. La ventaja de este sistema (la única ventaja), es que los procesos eran independientes por lo que la modificación de uno no afectaba al resto. Pero tiene grandes inconvenientes:

- **Datos redundantes.** Ya que se repiten continuamente
- **Datos inconsistentes.** Ya que un proceso cambia sus datos y no el resto. Por lo que el mismo dato puede tener valores distintos según qué aplicación acceda a él.
- **Coste de almacenamiento elevado.** Al almacenarse varias veces el mismo dato, se requiere más espacio en los discos. Luego se agotarán antes.
- **Difícil acceso a los datos.** Cada vez que se requiera una consulta no prevista inicialmente, hay que modificar el código de las aplicaciones o incluso crear una nueva aplicación.
- **Dependencia de los datos a nivel físico.** Para poder saber cómo se almacenan los datos, es decir qué estructura se utiliza de los mismos, necesitamos ver el código de la aplicación; es decir el código y los datos no son independientes.
- **Tiempos de procesamiento elevados**. Al no poder optimizar el espacio de almacenamiento.
- **Dificultad para el acceso simultáneo a los datos.** Es casi imposible de conseguir ya que se utilizan archivos que no admiten esta posibilidad. Dos usuarios no pueden acceder a los datos de forma concurrente.
- **Dificultad para administrar seguridad.** Ya que cada aplicación se crea independientemente; es por tanto muy difícil establecer criterios de seguridad uniformes.

A estos sistemas se les llama sistemas de gestión de ficheros. Se consideran también así a los sistemas que utilizan programas ofimáticos (como **Word** y **Excel**) para gestionar sus datos. De hecho estos sistemas producen los mismos (si no más) problemas.

sistemas de información orientados a los datos. bases de datos

En este tipo de sistemas los datos se centralizan en una **base de datos** común a todas las aplicaciones. Estos serán los sistemas que estudiaremos en este curso.

En esos sistemas los datos se almacenan en una única estructura lógica que es utilizable por las aplicaciones. A través de esa estructura se accede a los datos que son comunes a todas las aplicaciones.

ventajas

- **Independencia de los datos y los programas y procesos.** Esto permite modificar los datos sin modificar el código de las aplicaciones.
- **Menor redundancia**. No hace falta tanta repetición de datos. Sólo se indica la forma en la que se relacionan los datos**.**
- **Integridad de los datos**. Mayor dificultad de perder los datos o de realizar incoherencias con ellos.
- **Mayor seguridad en los datos.** Al permitir limitar el acceso a los usuarios. Cada tipo de usuario podrá acceder a unas cosas..

- **Datos más documentados.** Gracias a los **metadatos** que permiten describir la información de la base de datos.
- **Acceso a los datos más eficiente.** La organización de los datos produce un resultado más óptimo en rendimiento.
- **Menor espacio de almacenamiento.** Gracias a una mejor estructuración de los datos.
- **Acceso simultáneo a los datos.** Es más fácil controlar el acceso de usuarios de forma concurrente.

desventajas

- **Instalación costosa.** El control y administración de bases de datos requiere de un software y hardware poderoso
- **Requiere personal cualificado.** Debido a la dificultad de manejo de este tipo de sistemas.
- **Implantación larga y difícil.** Debido a los puntos anteriores. La adaptación del personal es mucho más complicada y lleva bastante tiempo.
- **Ausencia de estándares reales.** Lo cual significa una excesiva dependencia hacia los sistemas comerciales del mercado. Aunque, hoy en día, una buena parte de esta tecnología está aceptada como estándar de hecho.

[1.1.3] objetivo de los sistemas gestores de bases de datos

Un sistema gestor de bases de datos o **SGBD** (aunque se suele utilizar más a menudo las siglas **DBMS** procedentes del inglés, *Data Base Management System*) es el software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos.

En estos Sistemas se proporciona un conjunto coordinado de programas, procedimientos y lenguajes que permiten a los distintos usuarios realizar sus tareas habituales con los datos, garantizando además la seguridad de los mismos.

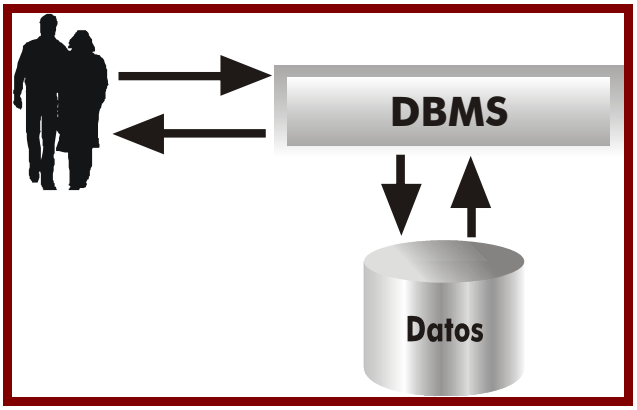


Ilustración 1, Esquema del funcionamiento y utilidad de un sistema gestor de bases de datos

El éxito del SGBD reside en mantener la seguridad e integridad de los datos. Lógicamente tiene que proporcionar herramientas a los distintos usuarios. Entre las herramientas que proporciona están:

- **Herramientas para la creación y especificación de los datos**. Así como la estructura de la base de datos.
- **Herramientas para administrar y crear la estructura física** requerida en las unidades de almacenamiento.
- **Herramientas para la manipulación de los datos** de las bases de datos, para añadir, modificar, suprimir o consultar datos.
- **Herramientas de recuperación** en caso de desastre
- **Herramientas para la creación de copias de seguridad**
- **Herramientas para la gestión de la comunicación** de la base de datos
- **Herramientas para la creación de aplicaciones** que utilicen esquemas externos de los datos
- **Herramientas de instalación** de la base de datos
- **Herramientas para la exportación e importación** de datos

[1.1.4] niveles de abstracción de una base de datos

introducción

En cualquier sistema de información se considera que se pueden observar los datos desde dos puntos vista:

- **Vista externa**. Esta es la visión de los datos que poseen los usuarios del Sistema de Información.
- **Vista física**. Esta es la forma en la que realmente están almacenados los datos.

En un Sistema de ficheros, los usuarios ven los datos desde las aplicaciones creadas por los programadores. Esa vista pueden ser formularios, informes visuales o en papel,... Pero la realidad física de esos datos, tal cual se almacenan en los discos, no la ven. Esa visión está reservada a los administrados.

En el caso de los Sistemas de Base de datos, se añade una tercera vista, que es la vista conceptual. Esa vista se sitúa entre la física y la externa. Se habla pues en Bases de datos de la utilización de tres esquemas para representar los datos.

esquema físico

Representa la forma en la que están almacenados los datos. Esta visión sólo la requiere el **administrador**. El administrador la necesita para poder gestionar más eficientemente la base de datos.

En este esquema es donde aparecen las unidades de disco, archivos y carpetas del sistema.

esquema conceptual

Se trata de un esquema teórico de los datos en la que figuran organizados en estructuras reconocibles del mundo real y en el que también aparece la forma de relacionarse los datos.

Esta estructura es utilizada por el **desarrollador**, que necesita conocerla para al crear aplicaciones saber como acceder. Todas las aplicaciones que se crean sobre la base de datos utilizan este esquema.

Realmente cuando se habla del diseño de la base de datos, se suele referir a este esquema. El esquema conceptual lo realiza el **diseñador** de la base de datos.

esquema externo

Se trata de la visión de los datos que poseen los **usuarios finales.** Esa visión es la que obtienen a través de las aplicaciones. Las aplicaciones creadas por los desarrolladores abstraen la realidad conceptual de modo que el usuario no conoce las relaciones entre los datos, como tampoco conoce todos los datos que realmente se almacenan.

Realmente cada aplicación produce un esquema externo diferente (aunque algunos pueden coincidir) o **vistas de usuario**. El conjunto de todas las vistas de usuario es lo que se denomina **esquema externo global**.

[1.2] componentes de los SGBD

[1.2.1] funciones. lenguajes de los SGBD

Los SGBD tienen que realizar tres tipos de funciones para ser considerados válidos.

función de descripción o definición

Permite al diseñador de la base de datos crear las estructuras apropiadas para integrar adecuadamente los datos. Este función es la que permite definir las tres estructuras de la base de datos (relacionadas con sus tres esquemas).

- **Estructura interna**
- **Estructura conceptual**
- **Estructura externa**

Esta función se realiza mediante el **lenguaje de descripción de datos** o **DDL**. Mediante ese lenguaje:

- **Se definen las estructuras de datos**
- **Se definen las relaciones entre los datos**
- **Se definen las reglas que han de cumplir los datos**

función de manipulación

Permite modificar y utilizar los datos de la base de datos. Se realiza mediante el **lenguaje de modificación de datos** o **DML.** Mediante ese lenguaje se puede:

- **Añadir datos**
- **Eliminar datos**
- **Modificar datos**
- **Buscar datos**

Actualmente se suele distinguir la función de buscar datos respecto del resto. Para lo cual se proporciona un **lenguaje de consulta de datos** o **DQL.**

función de control

Mediante esta función los administradores poseen mecanismos para determinar las visiones de los datos permitidas a cada usuario, además de proporcionar elementos de creación y modificación de esos usuarios.

Se suelen incluir aquí las tareas de copia de seguridad, carga de ficheros, auditoria, protección ante ataques externos, configuración del sistema,...

El lenguaje que implementa esta función es el **lenguaje de control de datos** o **DCL.**

[1.2.2] recursos humanos de las bases de datos

Intervienen (como ya se ha comentado) muchas personas en el desarrollo y manipulación de una base de datos. Habíamos seleccionado cuatro tipos de usuarios (administradores/as, desarrolladores, diseñadores/as y usuarios/as). Ahora vamos a desglosar aún más esta clasificación.

informáticos

Lógicamente son los profesionales que definen y preparan la base de datos. Pueden ser:

- **Directivos/as.** Son los organizadores y coordinadores del proyecto a desarrollar. Esto significa que son los encargados de decidir los recursos que se pueden utilizar, planificar el tiempo y las tareas, la atención al usuario y de dirigir las entrevistas y reuniones pertinentes.
- **Analistas**. Son los encargados de controlar el desarrollo de la base de datos aprobada por la dirección. Son además los diseñadores de la base de datos (especialmente de los esquemas interno y conceptual) y los coordinadores de la programación de la misma.
- **Administradores/as de las bases de datos.** Definen la seguridad de la base de datos y gestionan las copias de seguridad y la gestión física de la base de datos. Los analistas suelen tener esta funcionalidad cuando la base de datos está creada.
- **Desarrolladores/as o programadores/as.** Encargados de la realización de las aplicaciones de usuario de la base de datos.
- **Equipo de mantenimiento.** Encargados de dar soporte a los usuarios en el trabajo diario (suelen incorporar además tareas administrativas).

usuarios

- **Expertos/as.** Utilizan el lenguaje de manipulación de datos (DML) para acceder a la base de datos. Son usuarios que utilizan la base de datos para gestión avanzada de decisiones.
- **Habituales.** Utilizan las aplicaciones creadas por los desarrolladores para consultar y actualizar los datos. Son los que trabajan en la empresa a diario con estas herramientas y el objetivo fundamental de todo el desarrollo de la base de datos.
- **Ocasionales**. Son usuarios que utilizan un acceso mínimo a la base de datos a través de una aplicación que permite consultar ciertos datos. Serían por ejemplo los usuarios que consultan el horario de trenes a través de Internet.

[1.2.3] estructura multicapa

Un SGBD está en realidad formado por varias capas que actúan como interfaces entre el usuario y los datos. El propio ANSI/X3/SPARC introdujo una mejora de su modelo en 1988 a través de un grupo de trabajo llamado **UFTG** (*User Facilities Task Group,* grupo de trabajo para las facilidades de usuario). Este modelo toma como objeto principal, al usuario habitual de la base de datos y orienta el funcionamiento de la base de datos de modo que este usuario ignora el funcionamiento externo.

Desde esta óptica para llegar a los datos hay que pasar una serie de capas que poco a poco van entrando más en la realidad física de la base de datos. Esa estructura se muestra en la siguiente figura:

Ilustración 2, Modelo de referencia de las facilidades de usuario

[1.2.4] núcleo

El núcleo de la base de datos es el encargado de traducir las operaciones que le llegan a instrucciones ejecutables por el sistema operativo en el lenguaje que éste último requiera.

[1.2.5] diccionario de datos

Se trata del elemento que posee todos los metadatos. Gracias a esta capa las solicitudes de los clientes se traducen en instrucciones que hacen referencia al esquema interno de la base de datos. La capa de acceso a datos es la que permite comunicar a las aplicaciones de usuario con el diccionario de datos a través de las herramientas de gestión de datos que incorpore el SGBD.

[1.2.6] facilidades de usuario

Son las herramientas que proporciona el SGBD a los usuarios para permitir un acceso más sencillo a los datos. Actúan de interfaz entre el usuario y la base de datos, y son el único elemento que maneja el usuario.

[1.2.7] funcionamiento del SGBD

El esquema siguiente presenta el funcionamiento típico de un SGBD:

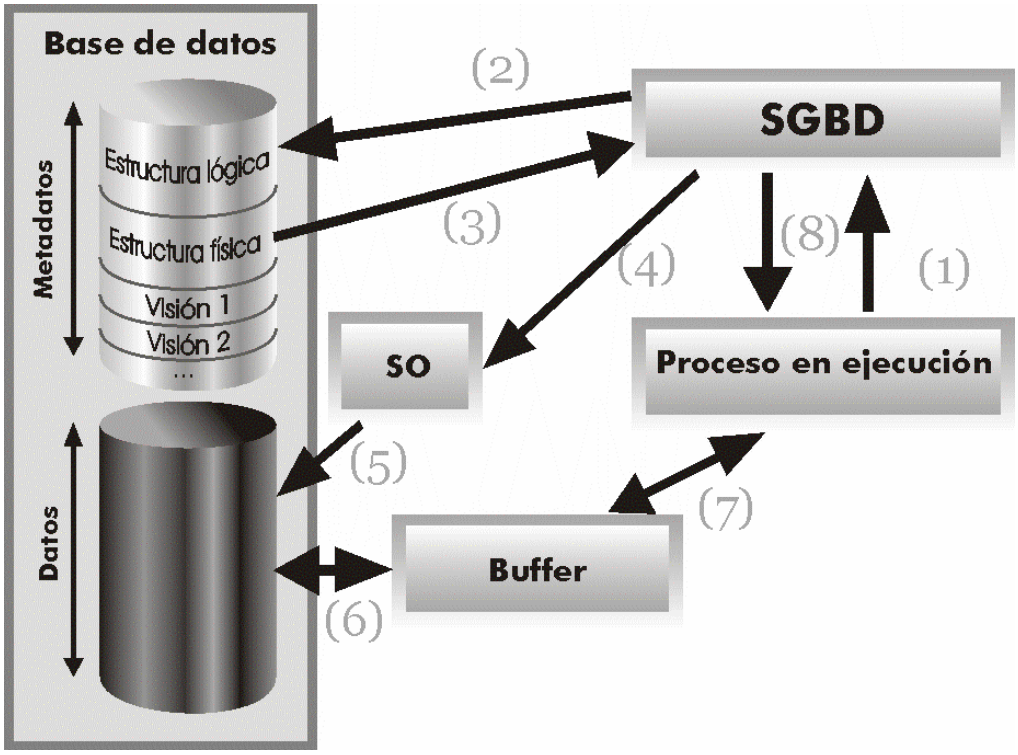


Ilustración 3, Esquema del funcionamiento de un SGBD

El esquema anterior reproduce la comunicación entre un proceso de usuario que desea acceder a los datos y el SGBD:

[1] El proceso lanzado por el usuario llama al SGBD indicando la porción de la base de datos que se desea tratar

[2] El SGBD traduce la llamada a términos del esquema lógico de la base de datos. Accede al esquema lógico comprobando derechos de acceso y la traducción física

[3] El SGBD obtiene el esquema físico

[4] El SGBD traduce la llamada a los métodos de acceso del Sistema Operativo que permiten acceder a los datos requeridos

[5] El Sistema Operativo accede a los datos tras traducir las órdenes dadas por el SGBD

[6] Los datos pasan del disco a una memoria intermedia o buffer. En ese buffer se almacenarán los datos según se vayan recibiendo

[7] Los datos pasan del buffer al área de trabajo del usuario (ATU) del proceso del usuario.

[8] El SGBD devuelve indicadores en los que manifiesta si ha habido errores o advertencias a tener en cuenta. Esto se indica al área de comunicaciones del proceso de usuario. Si las indicaciones son satisfactorias, los datos de la ATU serán utilizables por el proceso de usuario.

[1.3] arquitectura de los SGBD. estándares

Es uno de los aspectos que todavía sigue pendiente. Desde la aparición de los primeros gestores de base de datos se intentó llegar a un acuerdo para que hubiera una estructura común para todos ellos, a fin de que el aprendizaje y manejo de este software fuera más provechoso y eficiente.

El acuerdo nunca se ha conseguido del todo, no hay estándares aceptados del todo. Aunque sí hay unas cuentas propuestas de estándares que sí funcionan como tales.

[1.3.1] organismos de estandarización

Los intentos por conseguir una estandarización han estado promovidos por organismos de todo tipo. Algunos son estatales, otros privados y otros promovidos por los propios usuarios. Los tres que han tenido gran relevancia en el campo de las bases de datos son **ANSI/SPARC/X3**, **CODASYL** y **ODMG**. Los organismos grandes (que recogen grandes responsabilidades) dividen sus tareas en comités, y éstos en grupos de trabajo que se encargan de temas concretos.

[1.3.2] SC21

- **ISO** (*International Organization for Standardization).* Es un organismo internacional de definición de estándares de gran prestigio.
- **IEC** (*International Electrotechnical Commission*)**.** Organismo de definición de normas en ambientes electrónicos
- **JTC1** (*Joint Technical Committee*)**.** Comité formado por los dos organismos anteriores encargado de diversos proyectos. En el campo de las bases de datos, el subcomité **SC21** (en el que participan otros organismos nacionales, como el español AENOR) posee un grupo de trabajo llamado **WG 3** que se dedica a las bases de datos. Este grupo de trabajo es el que define la estandarización del lenguaje SQL entre otras cuestiones.

[1.3.3] DBTG Codasyl

Codasyl (*Conference on Data System Languages*) es el nombre de una conferencia de finales de los años 60 en la que participaron organismos privados y públicos del gobierno de Estados Unidos con la finalidad de definir estándares (Codasyl definió el lenguaje COBOL) para la informática de gestión.

De ahí salió DBTG *(Data Base Task Group*, grupo de tareas para bases de datos) grupo que definió el modelo en red de bases de datos que desde entonces se llama Codasyl o DBTG y que fue aceptado por la ANSI

[1.3.4] ANSI/X3/SPARC

ANSI (*American National Science Institute*) es un organismo científico de Estados Unidos que ha definido diversos estándares en el campo de las bases de datos. **X3** es la parte de ANSI encargada de los estándares en el mundo de la electrónica. Finalmente

SPARC ***System Planning and Repairments Committee,*** comité de planificación de sistemas y reparaciones es una subsección de X3 encargada de los estándares en Sistemas Informáticos en especial del campo de las bases de datos. Su logro fundamental ha sido definir un modelo de referencia para las bases de datos (que se estudiará posteriormente).

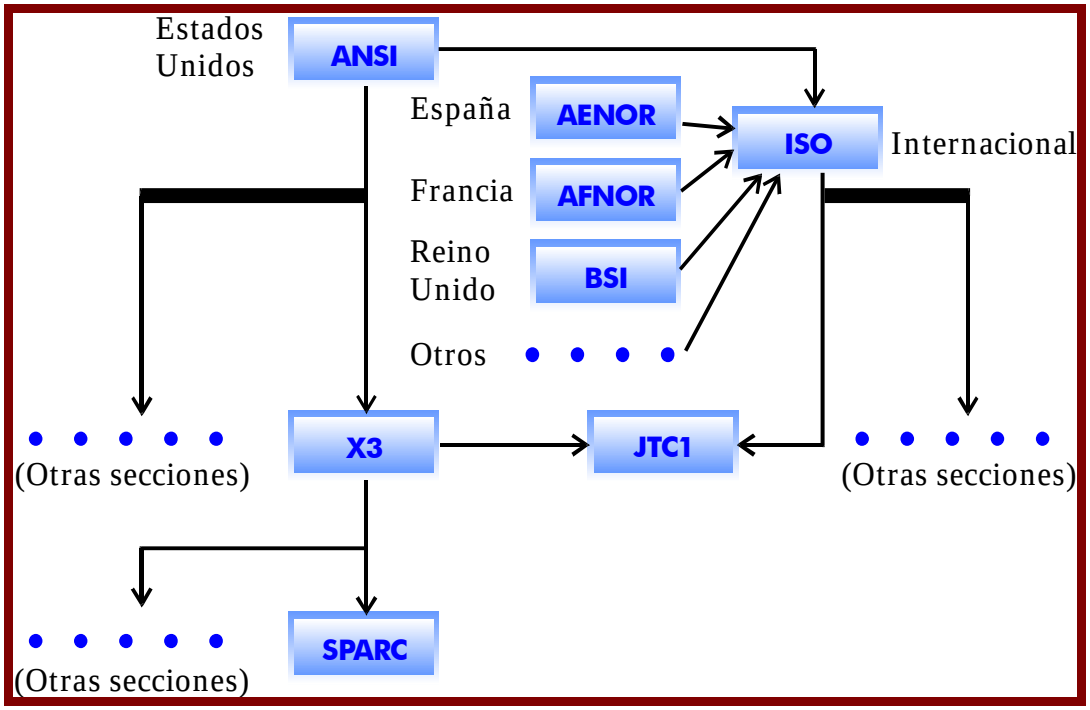


Ilustración 4, Relación entre los organismos de estandarización

En la actualidad ANSI para Estados Unidos e ISO para todo el mundo son nombres equivalentes en cuanto a estandarización de bases de datos, puesto que se habla ya de un único modelo de sistema de bases de datos.

[1.3.5] Modelo ANSI/X3/SPARC

El grupo ANSI ha marcado la referencia para la construcción de SGBD. Se basa en los estudios anteriores en los que se indicaban tres niveles de abstracción de la base de datos. ANSI profundiza más en esta idea y define cómo debe ser el proceso de creación y utilización de estos niveles.

En el modelo ANSI se indica que hay tres modelos (**externo**, **conceptual** e **interno**) entendiendo por modelo las normas que permiten crear esquemas (diseños de la base de datos). Los esquemas externos reflejan la información preparada para el usuario final, el esquema conceptual refleja los datos y relaciones de la base de datos y el esquema interno la preparación de los datos para ser almacenados.

El esquema conceptual contiene la información lógica de la base de datos. Su estructuración y las relaciones que hay entre los datos. Se trata de la propuesta teórica de los datos (es quizá la más importante).

El esquema interno contiene información sobre cómo están almacenados los datos en disco. Es el esquema más cercano a la organización real de los datos.

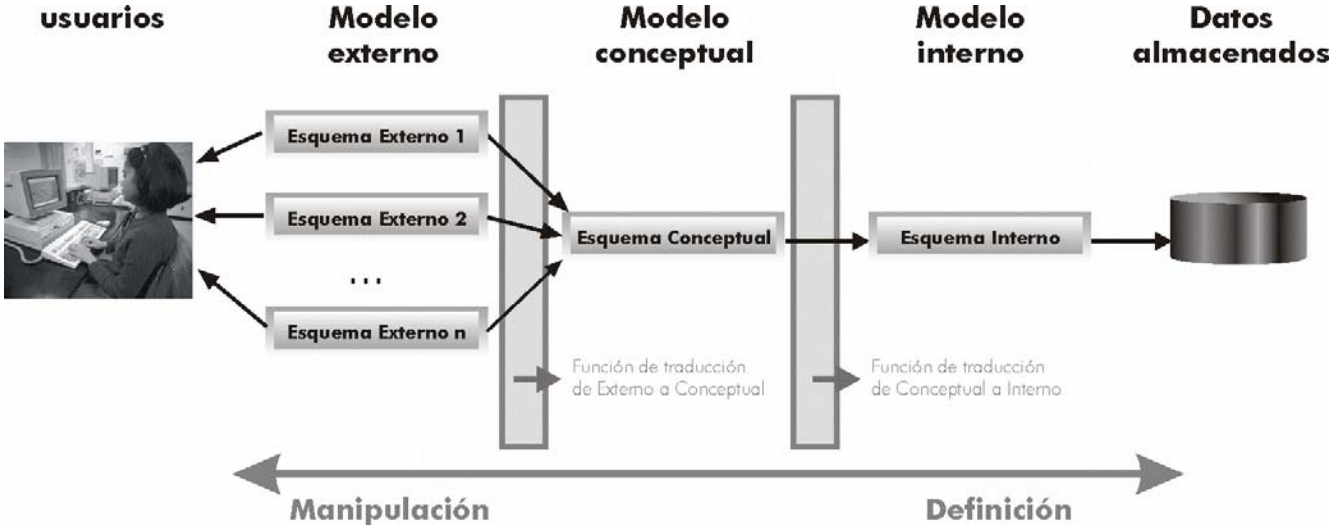


Ilustración 5, Niveles en el modelo ANSI

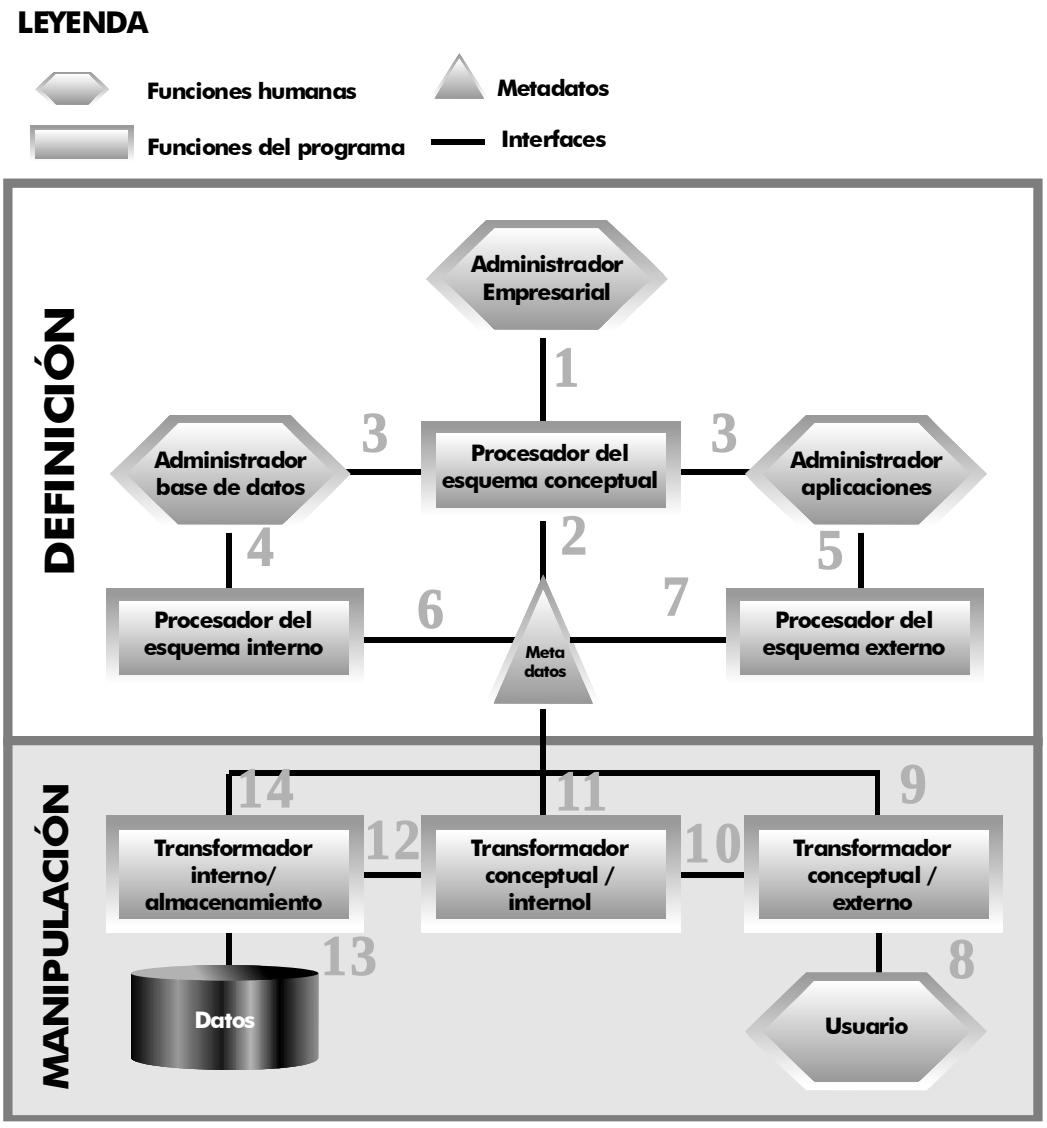


Ilustración 6, Arquitectura ANSI

El paso de un esquema a otro se realiza utilizando un interfaz o función de traducción. En su modelo, la ANSI no indica cómo se debe realizar esta función, sólo que debe existir.

La arquitectura completa (Ilustración 6) esta dividida en dos secciones, la zona de definición de datos y la de manipulación. Esa arquitectura muestra las funciones realizadas por humanos y las realizadas por programas.

En la fase de **definición**, una serie de interfaces permiten la creación de los **metadatos** que se convierten en el eje de esta arquitectura. La creación de la base de datos comienza con la elaboración del esquema conceptual realizándola el administrador de la empresa (actualmente es el diseñador, pero ANSI no lo llamó así). Ese esquema se procesa utilizando un procesador del esquema conceptual (normalmente una herramienta CASE, *interfaz 1* del dibujo anterior) que lo convierte en los metadatos (*interfaz 2*).

La *interfaz 3* permite mostrar los datos del esquema conceptual a los otros dos administradores: el administrador de la base de datos y el de aplicaciones (el desarrollador). Mediante esta información construyen los esquemas internos y externos mediante las *interfaces 4 y 5* respectivamente, los procesadores de estos esquemas almacenan la información correspondiente a estos esquemas en los metadatos (*interfaces 6 y 7*).

En la fase de **manipulación** el usuario puede realizar operaciones sobre la base de datos usando la *interfaz 8* (normalmente una aplicación) esta petición es transformada por el transformador externo/conceptual que obtiene el esquema correspondiente ayudándose también de los metadatos (*interfaz 9*). El resultado lo convierte otro transformador en el esquema interno (*interfaz 10*) usando también la información de los metadatos (*interfaz 11*). Finalmente del esquema interno se pasa a los datos usando el último transformador (*interfaz 12*) que también accede a los metadatos (*interfaz 13*) y de ahí se accede a los datos (*interfaz 14*). Para que los datos se devuelvan al usuario en formato adecuado para él se tiene que hacer el proceso contrario (observar dibujo).

[1.3.6] estructuras operacionales

Actualmente casi todos los sistemas gestores de base de datos poseen también la misma idea operacional en la que se entiende que la base de datos se almacena en un servidor y hay una serie de clientes que pueden acceder a los datos del mismo. Las posibilidades son:

- **Estructura Cliente-Servidor**. Estructura clásica, la base de datos y su SGBD están en un servidor al cual acceden los clientes. El cliente posee software que permite al usuario enviar instrucciones al SGBD en el servidor y recibir los resultados de estas instrucciones. Para ello el software cliente y el servidor deben utilizar software de comunicaciones en red.

- **Cliente multi-servidor**. Ocurre cuando los clientes acceden a datos situados en más de un servidor. También se conoce esta estructura como **base de datos distribuida.** El cliente no sabe si los datos están en uno o más servidores, ya que el resultado es el mismo independientemente de dónde se almacenan los datos. En esta estructura hay un servidor de aplicaciones que es el que recibe las peticiones y el encargado de traducirlas a los distintos servidores de datos para obtener los resultados. Una posibilidad muy extendida hoy en día es la posibilidad **Cliente/Servidor Web/Servidor de datos**, el cliente se conecta a un

servidor mediante un navegador web y desde las páginas de este ejecuta las consultas. El servidor web traduce esta consulta al servidor (o servidores) de datos.

[1.4] tipos de SGBD

[1.4.1] introducción

Como se ha visto en los apartados anteriores, resulta que cada SGBD puede utilizar un modelo diferente para los datos. Por lo que hay modelos conceptuales diferentes según que SGBD utilicemos.

No obstante existen modelos lógicos comunes, ya que hay SGBD de diferentes tipos. En la realidad el modelo ANSI se modifica para que existan dos modelos internos: el modelo lógico (referido a cualquier SGBD de ese tipo) y el modelo propiamente interno (aplicable sólo a un SGBD en particular). De hecho en la práctica al definir las bases de datos desde el mundo real hasta llegar a los datos físicos se pasa por los siguientes esquemas:

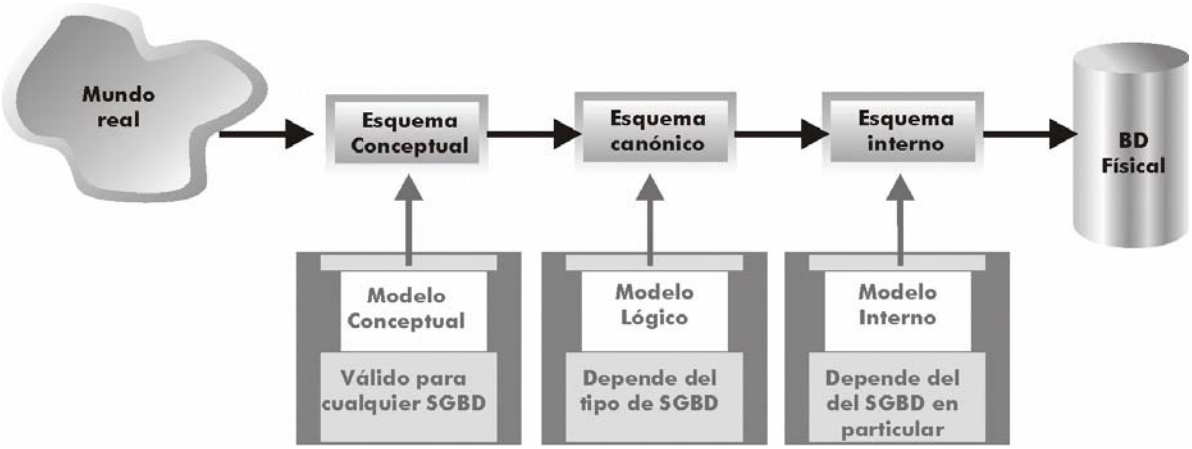


Ilustración 7, Modelos de datos utilizados en el desarrollo de una BD

Por lo tanto la diferencia entre los distintos SGBD está en que proporcionan diferentes modelos lógicos.

diferencias entre el modelo lógico y el conceptual

- El modelo conceptual es independiente del DBMS que se vaya a utilizar. El lógico depende de un **tipo** de SGBD en particular
- El modelo lógico es más cercano al ordenador
- El modelo conceptual es más cercano al usuario, el lógico es el encargado de establecer el paso entre el modelo informático y el modelo físico del sistema.

Algunos ejemplos de modelos conceptuales son:

- **Modelo E/R**

- **Modelo RM/T**
- **Modelos semánticos**

Ejemplos de modelos lógicos son:

- **Modelo relacional**
- **Modelo Codasyl**
- **Modelo Jerárquico**

A continuación se comentarán los modelos lógicos más importantes.

[1.4.2] modelo jerárquico

Era utilizado por los primeros SGBD, desde que IBM lo definió para su IMS (*Information Management System,* Sistema Administrador de Información) en 1970. Se le llama también modelo en árbol debido a que utiliza una estructura en árbol para organizar los datos.

La información se organiza con un jerarquía en la que la relación entre las entidades de este modelo siempre es del tipo **padre / hijo**. De esta forma hay una serie de nodos que contendrán atributos y que se relacionarán con nodos hijos de forma que puede haber más de un hijo para el mismo padre (pero un hijo sólo tiene un padre).

Los datos de este modelo se almacenan en estructuras lógicas llamadas **segmentos**. Los segmentos se relacionan entre sí utilizando **arcos.**

La forma visual de este modelo es de árbol invertido, en la parte superior están los padres y en la inferior los hijos.

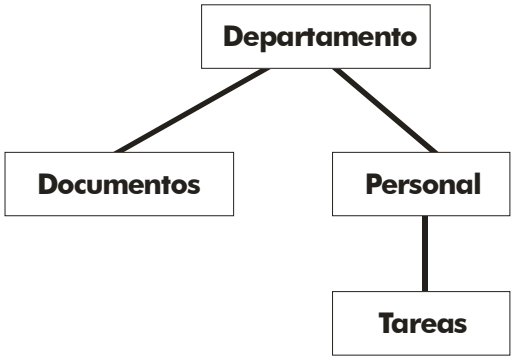


Ilustración 8, Ejemplo de esquema jerárquico

Este esquema está en absoluto desuso ya que no es válido para modelar la mayoría de problemas de bases de datos.

[1.4.3] modelo en red (Codasyl)

Es un modelo que ha tenido una gran aceptación (aunque apenas se utiliza actualmente). En especial se hizo popular la forma definida por Codasyl a principios de los 70 que se ha convertido en el modelo en red más utilizado.

El modelo en red organiza la información en **registros** (también llamados **nodos**) y **enlaces**. En los registros se almacenan los datos, mientras que los enlaces permiten

relacionar estos datos. Las bases de datos en red son parecidas a las jerárquicas sólo que en ellas puede haber más de un padre.

En este modelo se pueden representar perfectamente cualquier tipo de relación entre los datos (aunque el Codasyl restringía un poco las relaciones posibles), pero hace muy complicado su manejo.

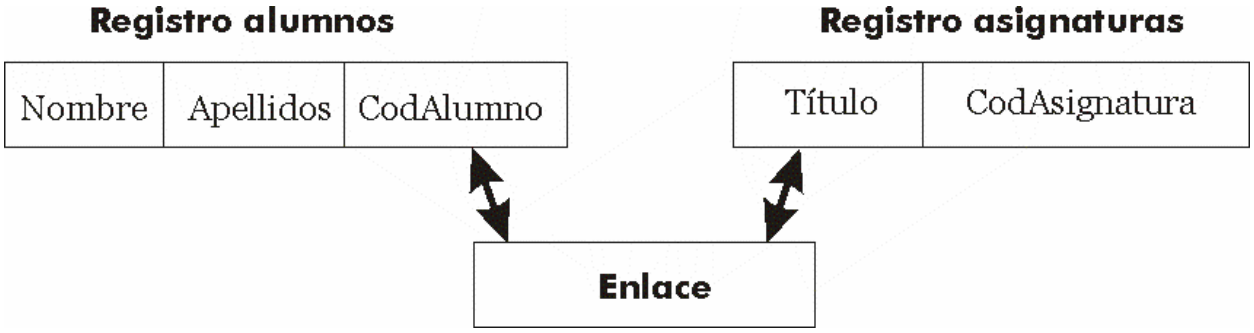


Ilustración 9, ejemplo de diagrama de estructura de datos Codasyl

[1.4.4] modelo relacional

En este modelo los datos se organizan en tablas cuyos datos se relacionan. Es el modelo más popular y se describe con más detalle en los temas siguientes.

[1.4.5] modelo de bases de datos orientadas a objetos

Desde la aparición de la programación orientada a objetos (**POO** u **OOP**) se empezó a pensar en bases de datos adaptadas a estos lenguajes. La programación orientada a objetos permite cohesionar datos y procedimientos, haciendo que se diseñen estructuras que poseen datos (**atributos**) en las que se definen los procedimientos (**operaciones**) que pueden realizar con los datos. En las bases orientadas a objetos se utiliza esta misma idea.

A través de este concepto se intenta que estas bases de datos consigan arreglar las limitaciones de las relacionales. Por ejemplo el problema de la herencia (el hecho de que no se puedan realizar relaciones de herencia entre las tablas), tipos definidos por el usuario, disparadores (triggers) almacenables en la base de datos, soporte multimedia...

Se supone que son las bases de datos de tercera generación (la primera fue las bases de datos en red y la segunda las relacionales), lo que significa que el futuro parece estar a favor de estas bases de datos. Pero siguen sin reemplazar a las relacionales, aunque son el tipo de base de datos que más está creciendo en los últimos años.

Su modelo conceptual se suele diseñar en **UML** y el lógico actualmente en **ODMG** (*Object Data Management Group,* grupo de administración de objetos de datos, organismo que intenta crear estándares para este modelo).

[1.4.6] bases de datos objeto relacionales

Tratan de ser un híbrido entre el modelo relacional y el orientado a objetos. El problema de las bases de datos orientadas a objetos es que requieren reinvertir capital y esfuerzos de nuevo para convertir las bases de datos relacionales en bases de datos orientadas a objetos. En las bases de datos objeto relacionales se intenta conseguir una compatibilidad relacional dando la posibilidad de integrar mejoras de la orientación a objetos.

Estas bases de datos se basan en el estándar **SQL 99**. En ese estándar se añade a las bases relacionales la posibilidad de almacenar procedimientos de usuario, triggers, tipos definidos por el usuario, consultas recursivas, bases de datos OLAP, tipos LOB,...

Las últimas versiones de la mayoría de las clásicas grandes bases de datos relacionales (**Oracle**, **SQL Server**, **Informix**, ...) son objeto relacionales.

[1.5] diseño conceptual de bases de datos. el modelo entidad - relación

[1.5.1] introducción

Ya hemos visto anteriormente que existen varios esquemas a realizar para poder representar en forma de base de datos informática un problema procedente del ordenador.

El primero de esos esquemas es el llamado **esquema conceptual**, que representa la información de forma absolutamente independiente al Sistema Gestor de Base de Datos. Los esquemas internos de las diferentes bases de datos no captan suficientemente bien la semántica del mundo real, de ahí que primero haya que pasar por uno o dos esquemas previos más cercanos al mundo real.

El hecho de saltarse el esquema conceptual conlleva un problema de pérdida con el problema real. El esquema conceptual debe reflejar todos los aspectos relevantes del mundo a real a modelar.

Peter P. Chen y el modelo entidad/relación

En 1976 y 1977 dos artículos de **Peter P. Chen** presentan un modelo para realizar esquemas que posean una visión unificada de los datos. Este modelo es el modelo entidad/interrelación (*entity/relationship* en inglés) que actualmente se conoce más con el nombre de entidad/relación (**Modelo E/R** o **ME/R**, en inglés *E/RM*).

Posteriormente otros autores han añadido mejoras a este modelo lo que ha producido una familia de modelos. La más aceptada actualmente es el modelo entidad/relación extendido (ERE) que complemente algunas carencias del modelo original. No obstante las diversas variantes del modelo hacen que la representación de este modelo no sea muy estándar, aunque hay ideas muy comunes a todas las variantes.

Hay que insistir en que este modelo no tiene nada que ver con las bases de datos relacionales, los esquemas entidad/relación se pueden utilizar con cualquier SGBD ya que son conceptuales. Confunde el uso de la palabra **relación**, pero el concepto de relación en este esquema no tiene nada que ver con la idea de relación expuesta por **Codd** en su modelo relacional.

[1.5.2] componentes del modelo

entidad

Se trata de cualquier objeto u elemento (real o abstracto) acerca del cual se pueda almacenar información en la base de datos. Es decir cualquier elemento informativo que tenga importancia para una base de datos.

Ejemplos de entidades son Pedro, la factura número 32456, el coche matrícula 3452BCW, etc. Una entidad no es un propiedad concreta sino un objeto que puede poseer múltiples propiedades (atributos). Es decir "Sánchez" es el contenido del atributo *Primer Apellido* de la entidad que representa a la persona Pedro Sánchez Crespo con DNI 12766374,...

Una entidad es un objeto concreto, no un simple dato: el coche que tenemos en el garaje es una entidad, "Mercedes" sin embargo es la marca de ese coche, es decir es un atributo de esa entidad.

conjuntos de entidades

Las entidades que poseen las mismas propiedades forman conjuntos de entidades. Ejemplos de conjuntos de entidades son los conjuntos: personas, facturas, coches,...

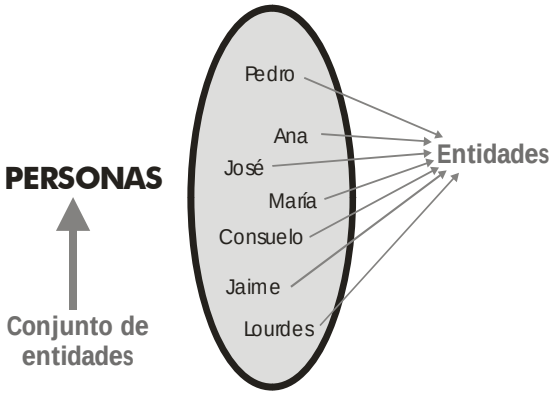


Ilustración 10, Ejemplos de entidad y conjunto de entidad

En la actualidad se suele llamar **entidad** a lo que anteriormente se ha definido como conjunto de entidades. De este modo hablaríamos de la entidad PERSONAS. Mientras que cada persona en concreto sería una **ocurrencia** o un **ejemplar** de la entidad **persona**.

Esa terminología es la que actualmente vamos a utilizar en este manual.

representación gráfica de las entidades

En el modelo entidad relación los conjuntos de entidades se representan con un rectángulo dentro del cual se escribe el nombre de la entidad:

Ilustración 11, Representación de la entidad persona

tipos de entidades

- **Regulares.** Son las entidades normales que tienen existencia por sí mismas sin depender de otras. Su representación gráfica es la indicada arriba
- **Débiles.** Su existencia depende de otras. Por ejemplo la entidad **tarea laboral** sólo podrá tener existencia si existe la entidad **trabajo.** Las entidades débiles se presentan de esta forma:

Ilustración 12, Entidad débil

[1.5.3] relaciones

qué es una relación

Representan **asociaciones** entre entidades. Es el elemento del modelo que permite relacionar en sí los datos del mismo. Por ejemplo, en el caso de que tengamos una entidad personas y otra entidad trabajos. Ambas se realizan ya que las personas trabajan y los trabajos son realizados por personas:

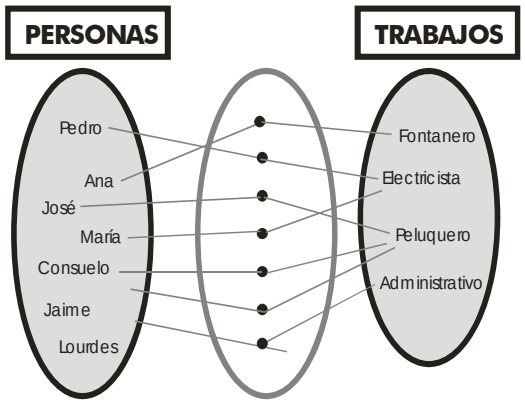


Ilustración 13, ejemplo de relación

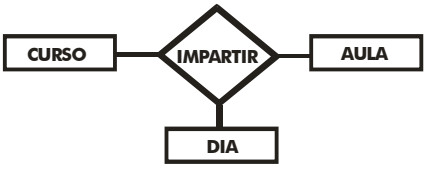
En una relación (Chen llamaba conjunto de relaciones a lo que ahora se llama relación a secas) cada ejemplar (relación en la terminología de Chen) asocia un elemento de una entidad con otro de la otra entidad. **En una relación no pueden aparecer dos veces relacionados los mismos ejemplares.** Es decir en el ejemplo anterior, en la relación no puede aparecer dos veces el mismo trabajador asociado al mismo trabajo.

representación gráfica

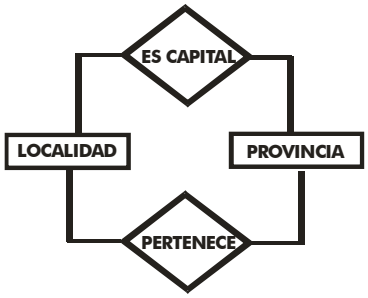
La representación gráfica de las entidades se realiza con un rombo al que se le unen líneas que se dirigen a las entidades, las relaciones tienen nombre (se suele usar un verbo). En el ejemplo anterior podría usarse como nombre de relación, trabajar:

ejemplos de relaciones

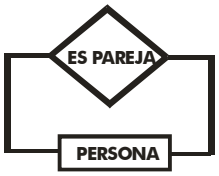
Relación binaria



Relación ternaria



Relación doble



Relación reflexiva

- **Relaciones Binarias.** Son las relaciones típicas. Se trata de relaciones que asocian dos entidades.
- **Relaciones Ternarias.** Relacionan tres entidades. A veces se pueden simplificar en relaciones binarias, pero no siempre es posible.
- **Relaciones *n*-arias.** Relacionan *n* entidades
- **Relaciones dobles.** Se llaman así a dos relaciones distintas que sirven para relacionar a las mismas relaciones. Son las más difíciles de manejar ya que al manipular las entidades hay que elegir muy bien la relacionan a utilizar para relacionar los datos.
- **Relación reflexiva.** Es una relación que sirve para relacionar ejemplares de la misma entidad (personas con personas, piezas con piezas, etc.)

cardinalidad

Indica el número de relaciones en las que una entidad puede aparecer. Se anota en términos de:

- **cardinalidad mínima.** Indica el número mínimo de asociaciones en las que aparecerá cada ejemplar de la entidad (el valor que se anota es de cero o uno, aunque tenga una cardinalidad mínima de más de uno, se indica sólo un uno)
- **cardinalidad máxima.** Indica el número máximo de relaciones en las que puede aparecer cada ejemplar de la entidad. Puede ser uno, otro valor concreto mayor que uno (tres por ejemplo) o muchos (se representa con *n*)

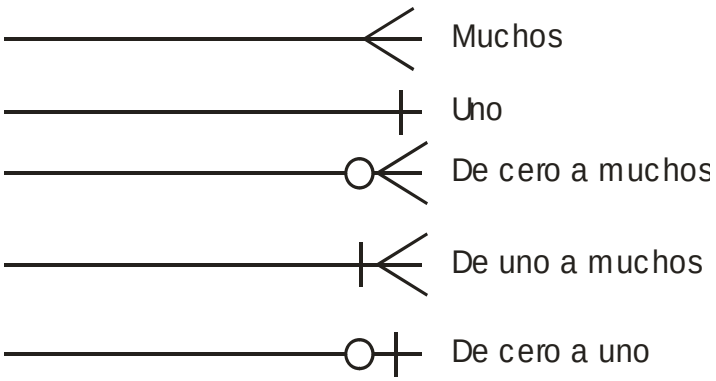
En los esquemas entidad / relación la cardinalidad se puede indicar de muchas formas. Quizá la más completa (y la que se utiliza en este documento es ésta) consiste en anotar en los extremos la cardinalidad máxima y mínima de cada entidad en la relación.

Ejemplo de uso de cardinalidad:

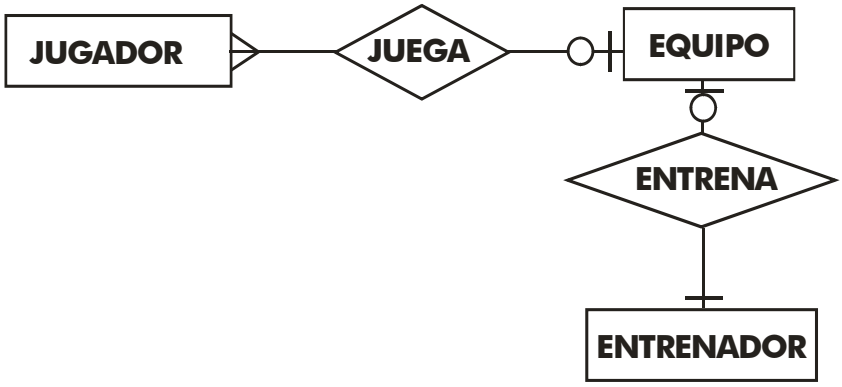
Ilustración 14, Cardinalidades.

En el ejemplo un jugador tiene una cardinalidad mínima de 0 (puede no estar en ningún equipo) y una máxima de 1 (como mucho está en un equipo, no puede estar en dos a la vez). Cada equipo tiene una cardinalidad mínima de uno (en realidad sería una cardinalidad mínima más alta, pero se anota un uno) y una máxima de *n* (en cada equipo hay muchos jugadores)

En la página siguiente se indican otras notaciones para las cardinalidades.

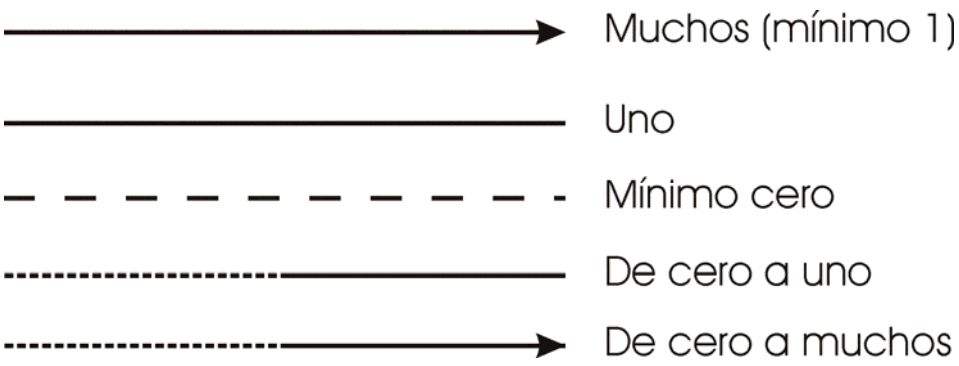


Ejemplo:



En el ejemplo, cada equipo cuenta con varios jugadores. Un jugador juega como mucho en un equipo y podría no jugar en ninguno. Cada entrenador entrena a un equipo (podría no entrenar a ninguno), el cual tiene un solo entrenador como mucho y como poco.

Otra notación es:



Y aún habría más pero nos quedaremos con la primera ya que es la más completa.

roles

A veces en las líneas de la relación se indican **roles.** Los roles representan el papel que juega una entidad en una determinada relación.

Ejemplo:

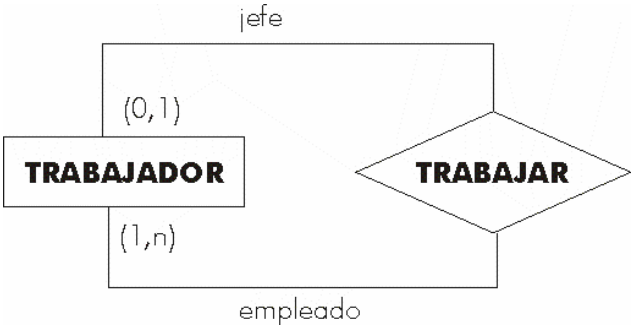


Ilustración 15, Ejemplo de rol. Un trabajador puede ser visto como jefe o como empleado según a qué lado de la relación esté

[1.5.4] atributos

Describen propiedades de las entidades y las relaciones. En este modelo se representan con un círculo, dentro del cual se coloca el nombre del atributo. Ejemplo:

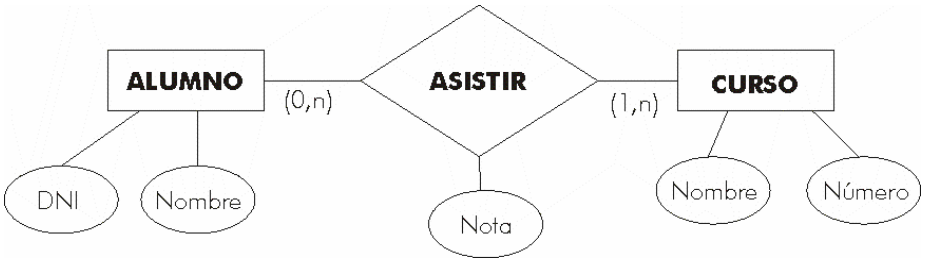
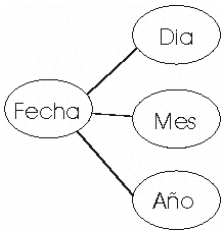


Ilustración 16, Atributos

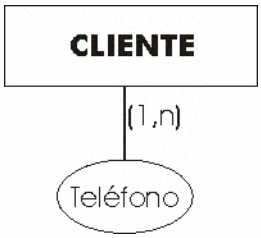
tipos de atributos

compuesto



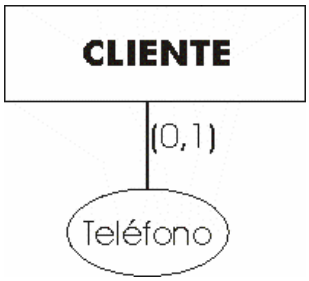
múltiples

Pueden tomar varios valores (varios teléfonos para el mismo cliente):



opcionales

Lo son si pueden tener valor nulo:



identificador o clave

Se trata de uno o más campos cuyos valores son únicos en cada ejemplar de una entidad. Se indican subrayando el nombre del identificador.

Para que un atributo sea considerado un buen identificador tiene que cumplir con los siguientes requisitos:

[1] Deben distinguir a cada ejemplar teniendo en cuenta las entidades que utiliza el modelo. No tiene que ser un identificador absoluto.

[2] Todos los ejemplares de una entidad deben tener el mismo identificador.

[3] Cuando un atributo es importante aun cuando no tenga una entidad concreta asociada, entonces se trata de una entidad y no de un atributo

identificador alternativo

Se trata de uno o más campos cuyos valores son únicos para cada ejemplar de una entidad, pero que no son identificadores ya que existen identificadores mejores en la entidad. En este caso los candidatos es aconsejable marcarlos con un subrayado discontinuo (ejemplo de subrayado discontinuo)

[1.5.5] modelo entidad relación extendido

En el modelo entidad relación extendido aparecen nuevos tipos de relaciones. Son las **relaciones ISA** (*es un*) y las **entidades débiles**

relaciones *is a* o relaciones de herencia

Se utilizan para unificar entidades agrupándolas en una entidad más general (**generalización**) o bien para dividir una entidad general en entidades más específicas (**especificación**).

Se habla de generalización si inicialmente partimos de una serie de entidades que al estudiarlas en detalle descubrimos que todas ellas pertenecen al mismo conjunto. En la generalización las entidades son totalmente heterogéneas, es decir, los atributos son diferentes. La entidad general se llama **superentidad** las otras se denominan **subentidades.** La superentidad normalmente tiene una clave principal distinta de las subentidades.

La especialización ocurre cuando partimos de una entidad que podemos dividir en subentidades para detallar atributos que varían en las mismas. Comparten clave con la superentidad y los atributos de la superclase se heredan en las subclases.

En la práctica se manejan casi igual ambas; de hecho la representación es la misma:

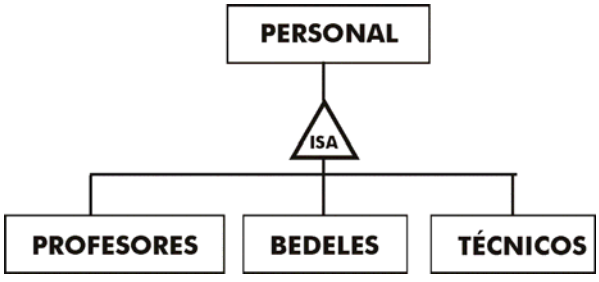


Ilustración 17, Relación ISA

La entidad general personal se ha dividido en tres pequeñas entidades. La cuestión de si es generalización o especialización no suele ser excesivamente importante salvo por el tema de la herencia de los atributos; hay que recordar que en la especialización, las subentidades heredan los atributos de la superentidad.

En el diseño la herencia de los atributos queda reflejado por las cardinalidades, si en la superentidad se indica una cardinalidad mínima de uno, se entiende entonces que se trata de de una especialización en las que las subentidades heredan los atributos de la superclase. Si la cardinalidad mínima es cero se entiende que las subclases no heredan los atributos.

Como se comentó antes la cuestión de si es una especialización o generalización se suele distinguir por las claves; si se comparte clave entre la superentidad y sus descendientes, se habla de especialización; de otro modo se habla de generalización (aunque esto es muy rebatible, en la práctica suele ser la única forma de distinguir ambos conceptos en el esquema).

De cualquier modo, la cuestión de si tenemos una generalización o una especialización no es tan importante como el hecho de no fallar con las cardinalidades, ya que al pasar el esquema al modelo relacional es lo que importa más.

La representación de relaciones ISA (independientemente de si es generalización o especialización) es esta:

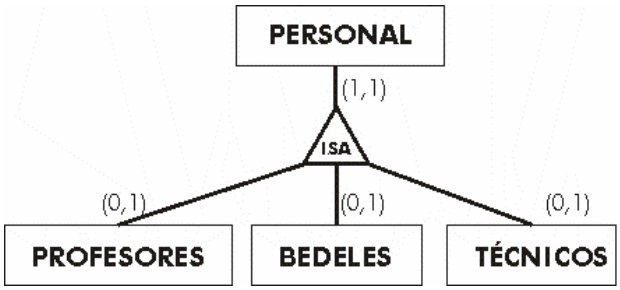


Ilustración 18, Relación ISA con cardinalidades

Con atributos el esquema sería:

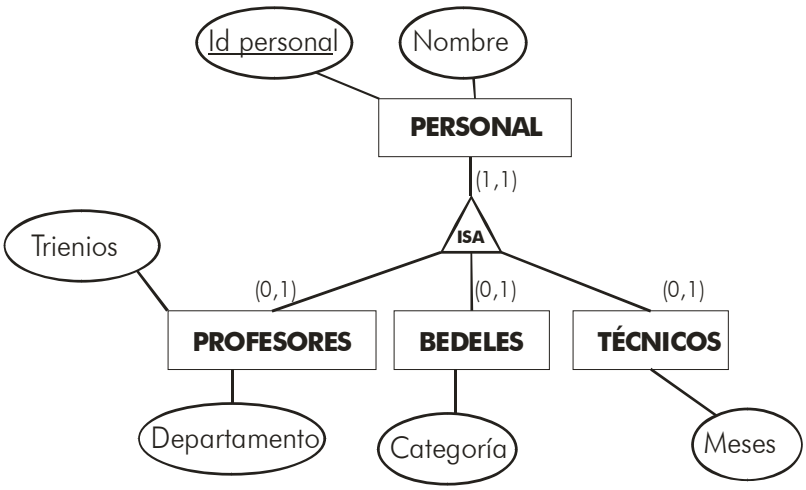


Ilustración 19, Especialización

En la especialización anterior (lo es porque la clave la tiene la superentidad) los profesores, bedeles y técnicos heredan el atributo **id personal** y el **nombre**, el resto son

atributos propios sólo de cada entidad (**trienios** pertenece sólo a los profesores, en este ejemplo)

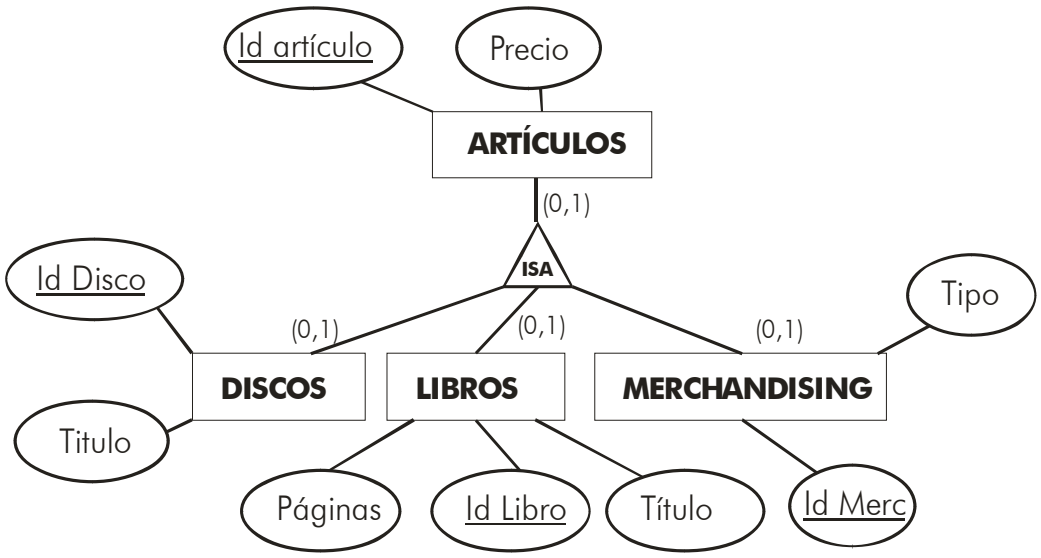


Ilustración 20, Generalización

En la ilustración anterior artículo es una generalización de los discos, libros y artículos de merchandising, se utiliza una clave distinta para esta entidad. Incluso en este caso podría haber discos o libros o merchandising que no están relacionados con los artículos (la cardinalidad de artículos es 0,1).

obligatoriariedad

En las relaciones ISA (y también en otros tipos de relaciones) se puede indicar el hecho de que cada ejemplar obligatoriamente tiene que participar en una de entre varias ramas de una relación. Este hecho se marca con un arco entre las distintas relaciones. En las relaciones ISA se usa mucho, por ejemplo:

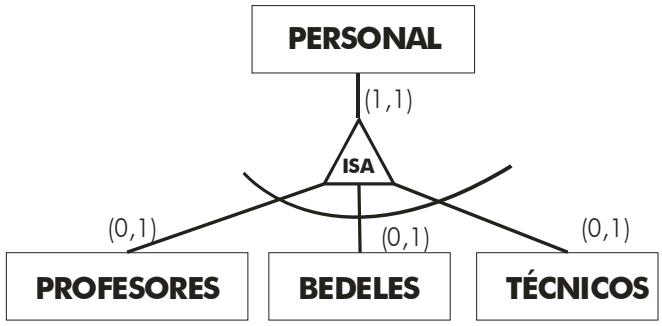


Ilustración 21, Relación ISA con obligatoriedad

En el ejemplo, el personal tiene que ser obligatoriamente un bedel, o un profesor o un técnico; una de las tres cosas (no puede haber personal que no sea una de estas tres cosas).

entidades débiles

Ya se ha comentado antes que una entidad débil es aquella cuya existencia depende de otra. Ahora vamos a clarificar más estas entidades. Efectivamente ocurren cuando hay

una entidad más fuerte de la que dependen. Lógicamente tienen relación con esa entidad. En la forma clásica se representaría de esta forma:

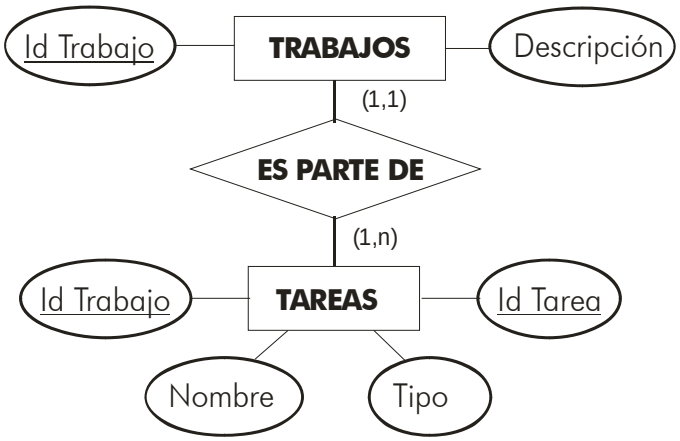


Ilustración 22, Relación candidata a entidad débil

En el diagrama la relación entre las tareas y los trabajos es 1 a *n* (cada trabajo se compone de *n* tareas). Una tarea obligatoriamente está asignada a un trabajo, es más no tiene sentido hablar de tareas sin hablar del trabajo del que forma parte.

Hay incluso (aunque no siempre) una **dependencia de identificación** ya que las tareas se identifican por un número de tarea y el número de trabajo al que se asignan. Esto es un síntoma definitivo de que se trata de una entidad débil.

Todas las entidades débiles tienen este tipo de relación 1 a *n* con respecto a la entidad fuerte de la que depende su existencia, por eso se representan de esta otra forma:

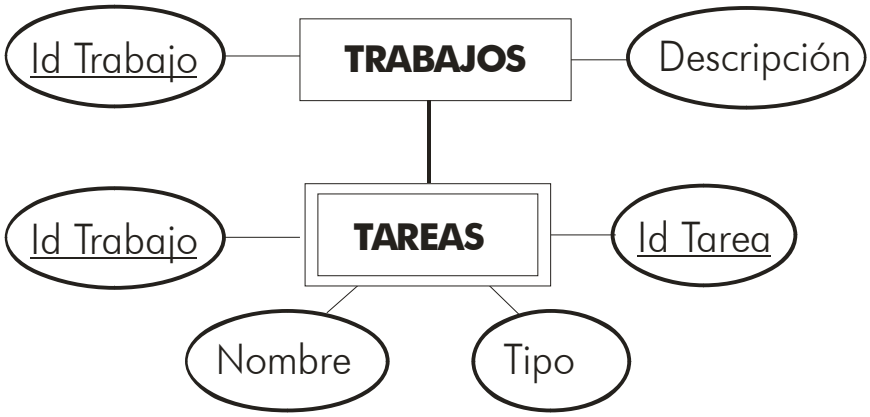


Ilustración 23, Entidad débil relacionada con su entidad fuerte

No hace falta dibujar el rombo de la relación ni la cardinalidad, se sobreentiende el tipo y cardinalidad (1 a n) que posee.