

1. Historia y Evolución de los SGBD

Antes de comenzar a abordar los Sistemas de Gestión de Base de Datos (SGBD), los invitamos a recorrer una breve lectura en la que se aborda la historia de los mismos. Comprender el contexto ayudará a comprender de mejor manera los conceptos que se desarrollan a continuación.

1. Introducción

Llama la atención el desarrollo sobresaliente de cualquier conjunto de datos organizados para su almacenamiento en la memoria de un ordenador o computadora en nuestros días, y logre un diseño para facilitar su mantenimiento y acceso de una forma estándar.

Hay dos buenas razones para la siguiente investigación. En primer lugar, el conocer los acontecimientos que dieron lugar al sistema gestor de base de datos nos proporciona cobertura detallada y comprensiva de su origen. En segundo lugar, si en algún momento fuera necesario convertir un sistema de gestión de base de datos, alcanzar cómo trabaja este sistema puede ser una ayuda esencial en el ámbito de los negocios, diseño e implementación de estrategias para la relación cliente/servidor.

El término bases de datos fue escuchado por primera vez en un simposio celebrado en California en 1963.

En una primera aproximación, se puede decir que una base de datos es un conjunto de información relacionada que se encuentra agrupada o estructurada.

Desde el punto de vista informático, una base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Por su parte, un sistema de Gestión de Bases de datos es un tipo de software muy específico dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan; o lo que es lo mismo, una agrupación de programas que sirven para definir, construir y manipular una base de datos, permitiendo así almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

Actualmente, las bases de datos están teniendo un impacto decisivo sobre el creciente uso de las computadoras.

2. Orígenes

Los orígenes de las bases de datos se remontan a la Antigüedad donde ya existían bibliotecas y toda clase de registros. Además también se utilizaban para recoger información sobre las cosechas y censos. Sin embargo, su búsqueda era lenta y poco eficaz y no se contaba con la ayuda de máquinas que pudiesen reemplazar el trabajo manual.

Posteriormente, el uso de las bases de datos se desarrolló a partir de las necesidades de almacenar grandes cantidades de información o datos. Sobre todo, desde la aparición de las primeras computadoras, el concepto de bases de datos ha estado siempre ligado a la informática.

En 1884 Herman Hollerith creó la máquina automática de tarjetas perforadas, siendo nombrado así el primer ingeniero estadístico de la historia.

En esta época, los censos se realizaban de forma manual.

Ante esta situación, Hollerith comenzó a trabajar en el diseño de una maquina tabuladora o censadora, basada en tarjetas perforadas.

Posteriormente, en la década de los cincuenta se da origen a las cintas magnéticas, para automatizar la información y hacer respaldos. Esto sirvió para suplir las necesidades de información de las nuevas industrias. Y a través de este mecanismo se empezaron a automatizar información, con la desventaja de que solo se podía hacer de forma secuencial.

Las aplicaciones informáticas de los años sesenta acostumbraban a darse totalmente por lotes (batch) y estaban pensadas para una tarea muy específica relacionada con muy pocas entidades tipo.

Cada aplicación (una o varias cadenas de programas) utilizaba ficheros de movimientos para actualizar (creando una copia nueva) y/o para consultar uno o dos ficheros maestros o, excepcionalmente, más de dos.

Cada programa trataba como máximo un fichero maestro, que solía estar sobre cinta magnética y, en consecuencia, se trabajaba con acceso secuencial.

Cada vez que se le quería añadir una aplicación que requería el uso de algunos de los datos que ya existían y de otros nuevos, se diseñaba un fichero nuevo con todos los datos necesarios (algo que provocaba redundancia). Para evitar que los programas tuviesen que leer muchos ficheros.

A medida que se fueron introduciendo las líneas de comunicación, los terminales y los discos, se fueron escribiendo programas que permitían a varios usuarios consultar los mismos ficheros on-line y de forma simultánea. Más adelante fue surgiendo la

necesidad de hacer las actualizaciones también on-line.

A medida que se integraban las aplicaciones, se tuvieron que interrelacionar sus ficheros y fue necesario eliminar la redundancia.

El nuevo conjunto de ficheros se debía diseñar de modo que estuviesen interrelacionados; al mismo tiempo, las informaciones redundantes (como por ejemplo, el nombre y la dirección de los clientes o el nombre y el precio de los productos), que figuraban en los ficheros de más de una de las aplicaciones, debían estar ahora en un solo lugar. El acceso on-line y la utilización eficiente de las interrelaciones exigían estructuras físicas que diesen un acceso rápido, como por ejemplo los índices, las multilistas, etc.

Estos conjuntos de ficheros interrelacionados, con estructuras complejas y compartidas por varios procesos de forma simultánea (unos on-line y otros por lotes), recibieron al principio el nombre de Data Banks, y después, a inicios de los años setenta, el de Data Bases. El software de gestión de ficheros era demasiado elemental para dar satisfacción a todas estas necesidades.

Por ejemplo, el tratamiento de las interrelaciones no estaba previsto, no era posible que varios usuarios actualizaran datos simultáneamente, etc.

La utilización de estos conjuntos de ficheros por parte de los programas de aplicación era excesivamente compleja, de modo que, especialmente durante la segunda mitad de los años setenta, fue saliendo al mercado software más sofisticado: los **Data Base Management Systems**, que aquí denominamos **Sistemas de Gestión de BD (SGBD)**. En otras palabras, una base de datos es un conjunto estructurado de datos que representa entidades y sus interrelaciones. La representación será única e integrada, a pesar de que debe permitir utilizaciones varias y simultáneas.

Con todo lo que hemos dicho hasta ahora, podríamos definir el término BD. Una base de datos de un **SI (Sistema de Información)** es la representación integrada de los conjuntos de entidades instancia correspondientes a las diferentes entidades tipo del SI y de sus interrelaciones.

Esta representación informática (o conjunto estructurado de datos) debe poder ser utilizada de forma compartida por muchos usuarios de distintos tipos.

3. Década de los 60.

Posteriormente en la época de los sesenta, las computadoras bajaron los precios para que las compañías privadas las pudiesen adquirir; dando paso a que se popularizara el uso de los discos, cosa que fue un adelanto muy efectivo en la época, debido a que a partir de este soporte se podía consultar la información directamente, sin tener que saber la ubicación exacta de los datos.

En esta misma época se dio inicio a las primeras generaciones de bases de datos de red y las bases de datos jerárquicas, ya que era posible guardar estructuras de datos en listas y arboles.

Otro de los principales logros de los años sesenta fue la alianza de IBM y American Airlines para desarrollar **SABRE**, un sistema operativo que manejaba las reservas de vuelos, transacciones e informaciones sobre los pasajeros de la compañía American Airlines.

Y, posteriormente, en esta misma década, se llevó a cabo el desarrollo del IDS desarrollado por Charles Bachman (**que formaba parte de la CODASYL**) supuso la creación de un nuevo tipo de sistema de bases de datos conocido como modelo en red que permitió la creación de un estándar en los sistemas de bases de datos gracias a la creación de nuevos lenguajes de sistemas de información.

CODASYL (**Conference on Data Systems Languages**) era un consorcio de industrias informáticas que tenían como objetivo la regularización de un lenguaje de programación estándar que pudiera ser utilizado en multitud de ordenadores.

Los miembros de este consorcio pertenecían a industrias e instituciones gubernamentales relacionadas con el proceso de datos, cuya principal meta era promover un análisis, diseño e implementación de los sistemas de datos más efectivos; y aunque trabajaron en varios lenguajes de programación como COBOL, nunca llegaron a establecer un estándar fijo, proceso que se llevó a cabo por ANSI.

4. Década de los 70 – Sistemas Centralizados.

Los primeros SGBD de los años sesenta todavía no se les denominaba así. Estaban orientados a facilitar la utilización de grandes conjuntos de datos en los que las interrelaciones eran complejas.

El arquetipo de aplicación era el “**Bill of materials o parts explosión**”, típica en las industrias del automóvil o de la construcción, de naves espaciales y en campos similares, de aquellos momentos..

Estos sistemas trabajaban exclusivamente por lotes (batch).

Al aparecer los terminales de teclado, conectados al ordenador central (**Mainframes**) mediante una línea telefónica, se empiezan a construir grandes aplicaciones on-line transaccionales (**OLTP**).

Los SGBD estaban íntimamente ligados al software de comunicaciones y de gestión de transacciones.

Aunque para escribir los programas de aplicación se utilizaban lenguajes de alto nivel como Cobol o PL/I, se disponía también de instrucciones y de subrutinas especializadas para tratar las BD que

requerían que el programador conociese muchos detalles del diseño físico, y que hacían que la programación fuese muy compleja.

Puesto que los programas estaban relacionados con el nivel físico, se debían modificar continuamente cuando se hacían cambios en el diseño y la organización de la BD. La preocupación básica era maximizar el rendimiento: el tiempo de respuesta y las transacciones por segundo.

Por lo que respecta a la década de los setenta, Edgar Frank Codd, científico informático inglés conocido por sus aportaciones a la teoría de bases de datos relacionales, definió el modelo relacional a la par que publicó una serie de reglas para los sistemas de datos relacionales a través de su artículo “**Un modelo relacional de datos para grandes bancos de datos compartidos**”.

Este hecho dio paso al nacimiento de la segunda generación de los Sistemas Gestores de Bases de Datos.

Como consecuencia de esto, durante la década de 1970, Lawrence J. Ellison, más conocido como Larry Ellison, a partir del trabajo de Edgar F. Codd sobre los

sistemas de bases de datos relacionales, desarrolló el Relational Software System, o lo que es lo mismo, lo que actualmente se conoce como Oracle Corporation, desarrollando así un sistema de gestión de bases de datos relacional con el mismo nombre que dicha compañía.

Inicialmente no se usó porque tuvo inconvenientes con el rendimiento, no podía competir con las bases de datos jerárquicas y de redes. Finalmente IBM desarrolló unas técnicas para construir un sistema de bases de datos relacionales eficientes, las cuales llamó System R; por otro lado Ingres se desarrolló en la UBC en los años de 1974 a 1977.

Ingres utilizaba un lenguaje de consulta, llamado QUEL, dando pie a la creación de sistemas como

Ingres Corporación, MS SQL Server, Sybase, PACE Wang, y Britton Lee-. Por su parte, el Sistema R utilizó el lenguaje de consulta Secuela, el cual ha contribuido al desarrollo de SQL / DS, DB2, Allbase, Oracle y SQL Non-Stop.

Posteriormente en la época de los ochenta también se desarrollará el **SQL (Structured Query Language)** o lo que es lo mismo un lenguaje de consultas o lenguaje declarativo de acceso a bases de datos relacionales que permite efectuar consultas con el fin de recuperar información de interés de una base de datos y hacer cambios sobre la base de datos de forma sencilla; además de analizar grandes cantidades de información y permitir especificar diversos tipos de operaciones frente a la misma información, a diferencia de las bases de datos de los años ochenta que se diseñaron para aplicaciones de procesamiento de transacciones.

Pero cabe destacar que ORACLE es considerado como uno de los sistemas de bases de datos más completos que existen en el mundo, y aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace relativamente poco, actualmente sufre la competencia del SQL Server de la compañía Microsoft y de la oferta de otros Sistemas Administradores de Bases de Datos Relacionales con licencia libre como es el caso de PostgreSQL, MySQL o Firebird que aparecerían posteriormente en la década de 1990.

2. Historia y evolución de los SGBD II

5. Década de los 80 – SGBD Relacionales.

Por su parte, a principios de los años ochenta comenzó el auge de la comercialización de los sistemas relacionales, y SQL comenzó a ser el estándar de la industria, ya que las bases de datos relacionales con su sistema de tablas (compuesta por filas y columnas) pudieron competir con las bases jerárquicas y de red, como consecuencia de que su nivel de programación era sencillo y su nivel de programación era relativamente bajo.

Los ordenadores minis, en primer lugar, y después los ordenadores micros, extendieron la informática a prácticamente todas las empresas e instituciones.

Los SGBD de los años sesenta y setenta (IMS de IBM, IDS de Bull, DMS de Univac, etc.) eran sistemas totalmente centralizados, como corresponde a los sistemas operativos de

aquellos años, y al hardware para el que estaban hechos: un gran ordenador para toda la empresa y una red de terminales sin inteligencia ni memoria.

Esto exigía que el desarrollo de aplicaciones fuese más sencillo.

Los SGBD de los años setenta eran demasiado complejos e inflexibles, y sólo los podía utilizar un personal muy cualificado.

En la década de los años 80', se desarrolló el SQL (Structured Query Language), un lenguaje de consultas que permite consultar, valga la redundancia, con el fin de recuperar información de una base de datos y a su vez, hacer cambios sobre esa misma base, de forma sencilla. Permitía analizar gran cantidad de información y especificar varios tipos de operaciones con la misma

información, a diferencia de los años anteriores, cuando se diseñaron aplicaciones de procesamientos de transacciones.

SQL comenzó a ser el modelo estándar de las industrias, con su base de datos bajo un sistema de tablas (filas y columnas), pudo competir con las bases jerárquicas y de redes, ya que su nivel de programación era sencillo y el nivel era relativamente bajo.

Estos sistemas de bases de datos relacionales fueron un éxito comercial, así como el aumento en la venta de ordenadores, estimulando el mercado de bases de datos, lo que provocó una caída importante en la popularidad de las redes y los modelos jerárquicos de bases de datos.

El ORACLE está considerado como uno de los sistemas de bases de datos más completos del mundo, su dominio en el mercado fue casi total hasta muchos años después, pero esto cambió con la aparición del SQL Server de Microsoft. La oferta de otros Sistemas Administradores de Bases de Datos Relacionales, como PostgreSQL, MySQL o Firebird aparecieron posteriormente en la década de 1990. Igualmente se da inicio a las bases de datos que se orientaban a los objetos.

6. Década de los 90: Distribución, C/S y 4GL

Al acabar la década de los ochenta, los SGBD relacionales ya se utilizaban prácticamente en todas las empresas. A pesar de todo, hasta la mitad de los noventa, cuando se ha necesitado un rendimiento elevado se han seguido utilizando los SGBD pre-relacionales.

A finales de los ochenta y principios de los noventa, las empresas se han encontrado con el hecho de que sus departamentos han ido comprando ordenadores departamentales y personales, y han ido haciendo aplicaciones con BD. El resultado ha

sido que en el seno de la empresa hay numerosas BD y varios SGBD de diferentes tipos o proveedores.

Este fenómeno de multiplicación de las BD y de los SGBD se ha visto incrementado por la fiebre de las fusiones de empresas.

Esta distribución ideal se consigue cuando las diferentes BD son soportadas por una misma marca de SGBD, es decir, cuando hay homogeneidad.

Sin embargo, esto no es tan sencillo si los SGBD son heterogéneos. En la actualidad, gracias principalmente a la estandarización del lenguaje SQL, los SGBD de marcas diferentes pueden darse servicio unos a otros y colaborar para dar servicio a un programa de aplicación. No obstante, en general, en los casos de heterogeneidad no se llega a poder dar en el programa que los utiliza la apariencia de que se trata de una única BD.

Además de esta distribución "impuesta", al querer tratar de forma integrada distintas BD preexistentes, también se puede hacer una distribución "deseada", diseñando una BD distribuida físicamente, y con ciertas partes replicadas en diferentes sistemas.

Las razones básicas por las que interesa esta distribución son las siguientes:

Disponibilidad

La disponibilidad de un sistema con una BD distribuida puede ser más alta, porque si queda fuera de servicio uno de los sistemas, los demás seguirán funcionando. Si los datos residentes en el sistema no disponible están replicados en otro sistema, continuarán estando disponibles. En caso contrario sólo estarán disponibles los datos de los demás sistemas.

Coste

Una BD distribuida puede reducir el coste. En el caso de un sistema centralizado, todos los equipos usuarios, que pueden estar distribuidos por distintas y lejanas áreas geográficas, están conectados al sistema central por medio de líneas de comunicación. El coste total de las comunicaciones se puede reducir haciendo que un usuario tenga más cerca los datos que utiliza con mayor frecuencia; por ejemplo, en un ordenador de su propia oficina o, incluso, en su ordenador personal.

Por ejemplo, un programa de aplicación que un usuario ejecuta en su PC (que está conectado a una red) pide ciertos datos de una BD que reside en un equipo UNIX donde, a su vez, se ejecuta el SGBD relacional que la gestiona. El programa de aplicación es el cliente y el SGBD es el servidor.

Un proceso cliente puede pedir servicios a varios servidores. Un servidor puede recibir peticiones de muchos clientes. En general, un proceso A que hace de cliente, pidiendo un servicio a otro proceso B puede hacer también de servidor de un servicio que le pida otro proceso C (o incluso el B, que en esta petición sería el cliente). Incluso el cliente y el servidor pueden residir en un mismo sistema.

La facilidad para disponer de distribución de datos no es la única razón, ni siquiera la básica, del gran éxito de los entornos C/S en los años noventa.

Tal vez el motivo fundamental ha sido la flexibilidad para construir y hacer crecer la configuración

informática global de la empresa, así como de hacer modificaciones en ella, mediante hardware y

software muy estándar y barato.

El éxito de las BD, incluso en sistemas personales, ha llevado a la aparición de los Fourth Generation Languages (4GL), lenguajes muy fáciles y potentes, especializados en el desarrollo de aplicaciones fundamentadas en BD.

Proporcionan muchas facilidades en el momento de definir, generalmente de forma visual, diálogos para introducir, modificar y consultar datos en entornos C/S.

7. Tendencias actuales

Los tipos de datos que se pueden definir en los SGBD relacionales de los años ochenta y noventa son muy limitados. La incorporación de tecnologías multimedia –imagen y sonido – en los SI hace necesario que los SGBD relacionales acepten atributos de estos tipos.

Sin embargo, algunas aplicaciones no tienen suficiente con la incorporación de tipos especializados en multimedia. Necesitan tipos complejos que el desarrollador pueda definir a medida de la aplicación.

En definitiva, se necesitan tipos abstractos de datos: **TAD**.

Los SGBD más recientes ya incorporaban esta posibilidad, y abren un amplio mercado de TAD

predefinidos o librerías de clases.

Esto nos lleva a la orientación a objetos (OO). El éxito de la OO al final de los años ochenta, en el

desarrollo de software básico, en las aplicaciones de ingeniería industrial y en la construcción de

interfaces gráficas con los usuarios, ha hecho que durante la década de los noventa se extendiese en prácticamente todos los campos de la informática. En los SI se inicia también la adopción, tímida de momento, de la OO.

La utilización de lenguajes como C++ o Java requiere que los SGBD relacionales se adapten a ellos con interfaces adecuadas.

La rápida adopción de la web a los SI hace que los SGBD incorporen recursos para ser servidores de páginas web, como por ejemplo la inclusión de SQL en guiones HTML, SQL incorporado en Java, etc.

Durante estos últimos años se ha empezado a extender un tipo de aplicación de las BD denominado Data Warehouse, o almacén de datos, que también produce algunos cambios en los SGBD relacionales del mercado.

A lo largo de los años que han trabajado con BD de distintas aplicaciones, las empresas han ido

acumulando gran cantidad de datos de todo tipo. Si estos datos se analizan convenientemente pueden dar información muy valiosa.

Por lo tanto, se trata de mantener una gran BD con información proveniente de toda clase de

aplicaciones de la empresa (e, incluso, de fuera). Los datos de este gran almacén, el Data Warehouse, se obtienen por una replicación más o menos elaborada de las que hay en las BD que se utilizan en el trabajo cotidiano de la empresa. Estos almacenes de datos se utilizan exclusivamente para hacer consultas, de forma especial para que lleven a cabo estudios los analistas financieros, los analistas de mercado, etc.

Actualmente, los SGBD se adaptan a este tipo de aplicación, incorporando, por ejemplo, herramientas como las siguientes:

La creación y el mantenimiento de réplicas, con una cierta elaboración de los datos.

La consolidación de datos de orígenes diferentes.

La creación de estructuras físicas que soporten eficientemente el análisis multidimensional.

Gestores de Bases de Datos

Hoy día la mayoría de las bases de datos se presentan en formato digital, gracias a los avances

tecnológicos en la informática y la electrónica. Esto ofrece un amplio abanico de soluciones al problema de almacenamiento de datos.

Los gestores de bases de datos, Database Management System o DBMS (SGBD) son programas que permiten almacenar y luego acceder a los datos de forma estructurada y rápida. Las aplicaciones más usadas son para gestiones de empresas e instituciones públicas, así como en entornos científicos, para almacenar la información experimental.

Una base de datos es un sistema compuesto por un conjunto de datos, los cuales están almacenados en discos, a los que se accede directamente y un conjunto de programas que regulen o manejen ese conjunto de datos.

Mientras que un sistema de Gestión de Bases de Datos es un software que sirve de interfaz entre la base de datos, el usuario y las aplicaciones que se utilizan.

Los mejores gestores de base de datos

El principal lenguaje de base de datos y el más utilizado desde que se conoce la programación de

gestión, es el Structured Query Language (SQL). Este, de consulta estructurada, facilita el acceso a la gestión de las bases de datos relaciones, lo que permite realizar tareas en ellas y realizar consultas, que sirvan para obtener, agregar, eliminar o modificar información.

Para el desarrollo de este lenguaje hay que utilizar un gestor de base de datos, de los que hay muchos, unos de acceso libre y otros de pago. Veamos cuáles son, primeramente, los gestores de base de datos de pago:

Oracle

Es de los más confiables sistemas de gestión de base de datos relacional, además del más usado. Es propiedad de Oracle Corporation y fue desarrollado en 1977.

Se accede directamente a los objetos, a través del lenguaje de consulta SQL, es muy utilizado en las empresas, con un componente de red que permite la comunicación a través de las redes.

Su versatilidad le facilita ejecutarse en casi todas las plataformas existentes, Windows, Unix, Linux, MAC OS, entre otros.

SQL Server

En competencia directa a Oracle, está SQL Server de Microsoft. Los dos ocupan gran parte del mercado en el sector de base de datos. Son muy parecidos en algunas de sus características y funciones, aunque tienen sus marcadas diferencias.

SQL Server se ejecuta en Transact-SQL, esto es un grupo de programas que pueden añadir características al programa, como tratamiento de errores y excepciones, extracción de datos de la web en forma directa, procesamiento de datos, uso de distintos lenguajes de programación y

otros más, que lo hacen un gestor muy completo y competitivo. Su carácter administrativo es otro valor agregado, tanto en sus funciones y seguridad, como en su flexibilidad.

Gestores de base de datos de acceso libre

Dos de los principales y más utilizados gestores de pago, que son de acceso libre (Open Source) son los siguientes:

MySQL

Este es de simple instalación y actúa de lado del cliente o servidor, es de código abierto y tiene licencia comercial disponible. Pertenece a Oracle Corporation y gestiona las bases de datos relacionales, con funciones multiusuario y es el más usado dentro del software libre. Requiere de poca memoria y procesador para su funcionamiento, lo que se traduce en mayor velocidad en sus operaciones. Se usa principalmente para el desarrollo Web.

Fire Bird

De gran potencia y muy sencillo a la vez, este sistema de gestión de base de datos relacional SQL, es uno de los mejores gestores Open Source (Código abierto) o libres. Es compatible con Windows y Linux.

Tiene buen soporte para los procedimientos almacenados, las transacciones compatibles con ACID y con los métodos de acceso múltiple como Nativo, Python, .NET, etc...

Como vemos, son múltiples las posibilidades que tenemos de acceso a gestores de base de datos, tanto adquiriendo licencias de pago como acudiendo a software libre. En función de los gustos, formas de trabajar y necesidades de cada uno, seguro

encontraremos distintos gestores de base de datos que pueden satisfacernos en pro de nuestro trabajo.

4 SISTEMAS GESTORES DE BASES DE DATOS

1.1 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 Internet. El propio nombre Informática hace referencia al hecho de ser una ciencia que trabaja con información. Desde los albores de la creación de ordenadores, la información se ha considerado como uno de los pilares de las computadoras digitales. Por ello las bases de datos son una de las aplicaciones más antiguas de la informática. En informática se conoce como dato a cualquier elemento informativo que tenga relevancia para el sistema. Desde el inicio de la informática se ha reconocido al dato como al elemento fundamental de trabajo en un ordenador. Por ello se han realizado numerosos estudios y aplicaciones para mejorar la gestión que desde las computadoras se realiza de los datos. Inicialmente los datos que se necesitaba almacenar y gestionar eran pocos, pero poco a poco han ido creciendo. En la actualidad las numerosas aplicaciones de Internet han producido enormes sistemas de información que incluso para poder gestionarlos requieren decenas de máquinas haciendo la información accesible desde cualquier parte del planeta y en un tiempo rápido. Eso ha requerido que la ciencia de las bases de datos esté en continua renovación para hacer frente a esas enormes necesidades. Pero incluso podemos remontarnos más al hablar de datos. El ser humano desde siempre ha necesitado gestionar datos; de esta forma se controlaban almacenes de alimentos, controles de inventario y otras muchos sistemas de datos. Como herramienta el ser humano al principio sólo poseía su memoria y cálculo y como mucho la ayuda de sus dedos. La escritura fue la herramienta que permitió al ser humano poder gestionar bases cada vez más grandes de datos. Además de permitir compartir esa información entre diferentes personas, también posibilitó que los datos se guardaran de manera continua e incluso estuvieran disponibles para las siguientes generaciones. Los problemas actuales con la privacidad ya aparecieron con la propia escritura y así el cifrado de datos es una técnica tan antigua como la propia escritura para conseguir uno de los todavía requisitos fundamentales de la gestión de datos, la seguridad. Para poder almacenar datos y cada vez más datos, el ser humano ideó nuevas herramientas archivos, cajones, carpetas y fichas en las que se almacenaban los datos. Sistemas Gestores de Bases de Datos Gestión y

diseño de bases de datos 1. Datos y Archivos. Tableta de arcilla del año 3000/3500 antes de Cristo. British Museum Imagen de BabelStone: http://commons.wikimedia.org/wiki/User:Babel_Stone
Antes de la aparición del ordenador, el tiempo 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 adaptó sus herramientas para que los elementos que el usuario maneja en el ordenador se parezcan a los que utilizaba manualmente. Así en informática se sigue hablando de ficheros, formularios, carpetas, directorios,....

2.1 La empresa como sistema Según la RAE, la definición de sistema es “Conjunto de cosas que ordenadamente relacionadas entre sí contribuyen a un determinado objeto”. La clientela fundamental del profesional de la informática es la empresa. La empresa se puede entender como un sistema formado por diversos objetos: el capital, los recursos humanos, los inmuebles, los servicios que presta, etc. El sistema completo que forma la empresa, por otra parte, se suele dividir en los siguientes subsistemas:

- 1) Subsistema productivo. También llamado subsistema real o físico. Representa la parte de la empresa encargada de gestionar la producción de la misma.
- 1) Subsistema financiero. Encargado de la gestión de los bienes económicos de la empresa
- 1) Subsistema directivo. Encargado de la gestión organizativa de la empresa

Hay que hacer notar que cada subsistema se asocia a un departamento concreto de la empresa.

2.2 Sistemas de información. Los sistemas que aglutinan los elementos que intervienen para gestionar la información que manejan los subsistemas empresariales es lo que se conoce como Sistemas de Información. Se suele utilizar las siglas SI o IS (de Information Server) para referirse a ello). Realmente un sistema de información sólo incluye la información que nos interesa de la empresa y los elementos necesarios para gestionar esa información. Un sistema de información genérico está formado por los siguientes elementos:

- 1) Recursos físicos. Carpetas, documentos, equipamiento, discos,....
- 1) Recursos humanos. Personal que maneja la información
- 1) Protocolo. Normas que debe cumplir la información para que sea manejada (formato de la información, modelo para los documentos,...)

Las empresas necesitan implantar estos sistemas de información debido a la competencia que las obliga a gestionar de la forma más eficiente sus datos para una mayor calidad en la organización de las actividades de los subsistemas empresariales.

2.3 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 electrónico), los componentes son:

-)] Datos. Se trata de la información relevante que almacena y gestiona el sistema de información. Ejemplos de datos son: Sánchez, 12764569F, Calle Mayo 5, Azul...
-)] Hardware. Equipamiento físico que se utiliza para gestionar los datos. cada uno de los dispositivos electrónicos que permiten el funcionamiento del sistema de información.
-)] Software. Aplicaciones informáticas que se encargan de la gestión de la base de datos y de las herramientas que facilitan su uso.
-)] Recursos humanos. Personal que maneja el sistema de información.

3. Los ficheros o archivos son la herramienta fundamental de trabajo en una computadora todavía a día de hoy. Las computadoras siguen almacenando la información en ficheros, eso sí de estructura cada vez más compleja. Los datos deben de ser almacenados en componentes de almacenamiento permanente, lo que se conoce como memoria secundaria (discos duros u otras unidades de disco). En esas memorias, los datos se estructuran en archivos (también llamados ficheros). Un fichero es una secuencia de números binarios que organiza información relacionada a un mismo aspecto.

En general sobre los archivos se pueden realizar las siguientes operaciones:

-)] Abrir (open). Prepara el fichero para su proceso.
-)] Cerrar (close). Cierra el fichero impidiendo su proceso inmediato.
-)] Leer (read). Obtiene información del fichero.
-)] Escribir (write). Graba información en el fichero.
-)] Posicionarse (seek). Coloca el puntero de lectura en una posición concreta del mismo (no se puede realizar en todos los tipos de ficheros).
-)] Fin de fichero (eof). Indica si hemos llegado al final del fichero. Cuando los ficheros almacenan datos, se dice que constan de registros. Cada registro contiene datos relativos a un mismo elemento u objeto. Por ejemplo en un fichero de personas, cada registro contiene datos de una persona. Si el archivo contiene datos de 1000 personas, constará de 1000 registros. A continuación se explican los tipos más habituales de ficheros.

3.1 Ficheros secuenciales. En estos ficheros, los datos se organizan secuencialmente en el orden en el que fueron grabados. Para leer los últimos datos hay que leer los anteriores. Es decir leer el registro número nueve, implica leer previamente los ocho anteriores. Ventajas

-)] Rápidos para obtener registros contiguos de una base de datos
-)] No hay huecos en el archivo al grabarse los datos seguidos, datos más compactos.

Desventajas

-)] Consultas muy lentas al tener que leer todos los datos anteriores al dato que queremos leer
-)] Algoritmos de lectura y escritura

más complejos] No se pueden eliminar registros del fichero (se pueden marcar de manera especial para que no sean tenidos en cuenta, pero no se pueden borrar)] El borrado provoca archivos que no son compactos] La ordenación de los datos requiere volver a crearle de nuevo

3.2 Ficheros de acceso directo o aleatorio. Se puede leer una posición concreta del fichero, con saber la posición (normalmente en bytes) del dato a leer. Cuando se almacenan registros, posicionarlos en el quinto registro se haría de golpe, lo único necesitamos saber es el tamaño del registro, que en este tipo de ficheros debe de ser el mismo. Suponiendo que cada registro ocupa 100 bytes, el quinto registro comienza en la posición 400. Lo que se hace es colocar el llamado puntero de archivo en esa posición y después leer. Acceso directo al registro N° 5 Ventajas] Acceso rápido al no tener que leer los datos anteriores] La modificación de datos es más sencilla] Permiten acceso secuencial] Permiten leer y escribir a la vez] Aptos para organizaciones relativas directas, en las que la clave del registro se relaciona con su posición en el archivo Desventajas] Salvo en archivos relativos directos, no es apto por sí mismo para usar en bases de datos, ya que los datos se organizan en base a una clave] No se pueden borrar datos (sí marcar para borrado, pero generarán huecos)] Las consultas sobre multitud de registros son más lentas que en el caso anterior.

3.3 Ficheros secuenciales encadenados. Son ficheros secuenciales gestionados mediante punteros, datos especiales que contienen la dirección de cada registro del fichero. Cada registro posee ese puntero que indica la dirección del siguiente registro y que se puede modificar en cualquier momento. El puntero permite recorrer los datos en un orden concreto. Cuando aparece un nuevo registro, se añade al final del archivo, pero los punteros se reordenan para que se mantenga el orden. Ventajas] El fichero mantiene el orden en el que se añadieron los registros y un segundo orden en base a una clave] La ordenación no requiere reorganizar todo el fichero, sino sólo modificar los punteros] Las mismas ventajas que el acceso secuencial] En esta caso sí se borran los registros y al reorganizar, se perderán definitivamente Desventajas] No se borran los registros, sino que se marcan para ser ignorados. Por lo que se malgasta espacio] Añadir registros o modificar las claves son operaciones que requieren recalcular los punteros

3.4 Ficheros secuenciales indexados. Se utilizan dos ficheros para los datos, uno posee los registros almacenados de forma secuencial, pero que permite su acceso aleatorio. El otro posee una tabla con punteros a la posición ordenada de los registros. Ese segundo fichero es el índice, una tabla con la ordenación deseada para los registros y la posición que

ocupan en el archivo. El archivo de índices posee unas cuantas entradas sólo en las que se indica la posición de ciertos valores claves en el archivo (cada 10, 15 ,20,... registros del archivo principal se añade una entrada en el de índices). El archivo principal tiene que estar siempre ordenado y así cuando se busca un registro, se busca su valor clave en la tabla de índices, la cual poseerá la posición del registro buscado. Desde esa posición se busca secuencialmente el registro hasta encontrarlo. Existe un archivo llamado de desbordamiento u overflow en el que se colocan los nuevos registros que se van añadiendo (para no tener que ordenar el archivo principal cada vez que se añade un nuevo registro) este archivo está desordenado. Se utiliza sólo si se busca un registro y no se encuentra en el archivo principal. En ese caso se recorre todo el archivo de overflow hasta encontrarlo. Para no tener demasiados archivos en overflow (lo que restaría velocidad), cada cierto tiempo se reorganiza el archivo principal.

Ejemplo: Ficheros secuenciales indexado

Ventajas] El archivo está siempre ordenado en base a una clave] La búsqueda de datos es rapidísima] Permite la lectura secuencial (que además será en el orden de la clave)] El borrado de registros es posible (aunque más problemático que en el caso anterior) Desventajas] Para un uso óptimo hay que reorganizar el archivo principal y esta operación es muy costosa ya que hay que reescribir de nuevo y de forma ordenada todo el archivo.] La adición de registros requiere más tiempo que en los casos anteriores al tener que reordenar los índices 3.5 Ficheros indexado – encadenados. Utiliza punteros e índices, es una variante encadenada del caso anterior. Hay un fichero de índices equivalente al comentado en el caso anterior y otro fichero de tipo encadenado con punteros a los siguientes registros. Cuando se añaden registros se añaden en un tercer registro llamado de desbordamiento u overflow. En ese archivo los datos se almacenan secuencialmente, se accede a ellos si se busca un dato y no se encuentra en la tabla de índices. Ficheros indexados – encadenados

Ventajas] Posee las mismas ventajas que los archivos secuenciales indexados, además de una mayor rapidez al reorganizar el fichero (sólo se modifican los punteros) Desventajas] Requieren compactar los datos a menudo para reorganizar índices y quitar el fichero de desbordamiento.

4. Operaciones relacionadas con uso de ficheros en bases de datos

Borrado y recuperación de registros Algunos de los tipos de ficheros vistos anteriormente no admiten el borrado real de datos, sino que sólo permiten añadir un dato que indica si el registro está borrado o no. Esto es interesante ya que permite anular una operación de borrado. Por ello esta técnica de marcar registros, se utiliza casi siempre en todos los tipos de archivos. En otros casos los datos antes de ser eliminados del todo

pasan a un fichero especial (conocido como papelera) en el que se mantienen durante cierto tiempo para su posible recuperación.

Fragmentación y compactación de datos La fragmentación en un archivo hace referencia a la posibilidad de que éste tenga huecos interiores debido a borrado de datos u a otras causas. Causa los siguientes problemas:

- 1) Mayor espacio de almacenamiento
- 2) Lentitud en las operaciones de lectura y escritura del fichero Por ello se requiere compactar los datos.

Esta técnica permite eliminar los huecos interiores a un archivo. Las formas de realizarla son:

- 1) Reescribir el archivo para eliminar los huecos. Es la mejor, pero lógicamente es la más lenta al requerir releer y reorganizar todo el contenido del fichero.
- 2) Aprovechar huecos. De forma que los nuevos registros se inserten en esos huecos. Esta técnica suele requerir un paso previo para reorganizar esos huecos.

Compresión de datos En muchos casos para ahorrar espacio de almacenamiento, se utilizan técnicas de compresión de datos. La ventaja es que los datos ocupan menos espacio y la desventaja es que al manipular los datos hay que descomprimirlos lo que hace que la manipulación de los datos sea lenta.

Cifrado de datos

Otra de las opciones habituales sobre ficheros de datos es utilizar técnicas de cifrado para proteger los ficheros en caso de que alguien no autorizado se haga con el fichero. Para leer un fichero de datos, haría falta descifrar el fichero. Para descifrar necesitamos una clave o bien aplicar métodos de descifrado; lógicamente cuanto mejor sea la técnica de cifrado, más difícil será descifrar los datos mediante la fuerza bruta.

Sistemas Gestores de Bases de Datos Gestión y diseño de bases de datos 1. 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.

1.1 Sistemas de Información orientados a procesos. Antes de aparecer los SGBD (década de los setenta), la información se trataba y se gestionaba utilizando los típicos sistemas de gestión de archivos que iban soportados sobre un sistema operativo. Éstos consistían en un conjunto de programas que definían y trabajaban sus propios datos. Los datos se almacenaban en archivos y los programas manejan esos archivos para obtener la información. Si la estructura de los datos de los archivos cambia, todos los programas que los manejan se deben modificar; por ejemplo, un programa trabaja con un archivo de datos de alumnos, con una estructura o registro ya definido; si se incorporan elementos o campos a la estructura del archivo, los programas que utilizan ese archivo se tienen que modificar para tratar esos nuevos elementos.

estos sistemas de gestión de archivos, la definición de los datos se encuentra codificada dentro de los programas de aplicación en lugar de almacenarse de forma independiente, y además el control del acceso y la manipulación de los datos vienen impuesto por los programas de aplicación. 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 son independientes por lo que la modificación de uno no afectaba al resto. Esto supone un gran inconveniente a la hora de tratar grandes volúmenes de información. Surge así la idea de separar los datos contenidos en los archivos de los programas que los manipulan, es decir, que se pueda modificar la estructura de los datos de los archivos sin que por ello se tengan que modificar los programas con los que trabajan. Se trata de estructurar y organizar los datos de forma que se pueda acceder a ellos con independencia de los programas que los gestionan. Inconvenientes de un sistema de gestión de archivos:

-]) Redundancia e inconsistencia de los datos: se produce porque los archivos son creados por distintos programas y van cambiando a lo largo del tiempo, es decir, pueden tener distintos formatos y los datos pueden estar duplicados en varios sitios. Por ejemplo, el teléfono de un alumno puede aparecer en más de un archivo. La redundancia aumenta los costes de almacenamiento y acceso, y trae consigo la inconsistencia de los datos: las copias de los mismos datos no coinciden por aparecer en varios archivos.
-]) Dependencia de los datos física-lógica: o lo que es lo mismo, la estructura física de los datos (definición de archivos y registros) se encuentra codificada en los programas de aplicación. Sistemas Gestores de Bases de Datos Gestión y diseño de bases de datos 1. Tipos de Sistemas de Información Cualquier cambio en esa estructura implica al programador identificar, modificar y probar todos los programas que manipulan esos archivos.
-]) Dificultad para tener acceso a los datos: proliferación de programas, es decir, cada vez que se necesite una consulta que no fue prevista en el inicio implica la necesidad de codificar el programa de aplicación necesario. Lo que se trata de probar es que los entornos convencionales de procesamiento de archivos no permiten recuperar los datos necesarios de una forma conveniente y eficiente.
-]) Separación y aislamiento de los datos: es decir, al estar repartidos en varios archivos, y tener diferentes formatos, es difícil escribir nuevos programas que aseguren la manipulación de los datos correctos. Antes se deberían sincronizar todos los archivos para que los datos coincidiesen.
-]) Dificultad para el acceso concurrente: pues en un

sistema de gestión de archivos es complicado que los usuarios actualicen los datos simultáneamente. Las actualizaciones concurrentes pueden dar por resultado datos inconsistentes, ya que se puede acceder a los datos por medio de diversos programas de aplicación.) Dependencia de la estructura del archivo con el lenguaje de programación: pues la estructura se define dentro de los programas. Esto implica que los formatos de los archivos sean incompatibles. La incompatibilidad entre archivos generados por distintos lenguajes hace que los datos sean difíciles de procesar. 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.) Problemas en la seguridad de los datos: Resulta difícil implantar restricciones de seguridad pues las aplicaciones se van añadiendo al sistema según se van necesitando. Ya que cada aplicación se crea independientemente; es por tanto muy difícil establecer criterios de seguridad uniformes.) Problemas de integridad de datos (datos inconsistentes): es decir, los valores almacenados en los archivos deben cumplir con restricciones de consistencia. 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. Por ejemplo, no se puede insertar una nota de un alumno en una asignatura si previamente esa asignatura no está creada. Otro ejemplo, las unidades en almacén de un producto determinado no deben ser inferiores a una cantidad. Esto implica añadir gran número de líneas de código en los programas. El problema se complica cuando existen restricciones que implican varios datos en distintos archivos. Aplicación 1 Aplicación 2 Aplicación 3 Datos Datos Datos A estos sistemas se les llama sistemas de gestión de ficheros. Se consideran también así a los sistemas que utilizan programas ofimáticos (como Word o Excel por ejemplo) para gestionar sus datos (muchas pequeñas empresas utilizan esta forma de administrar sus datos). De hecho estos sistemas producen los mismos (si no más) problemas.

1.2 Sistemas de Información orientados a los datos. Bases de Datos.

Todos estos inconvenientes hacen posible el fomento y desarrollo de SGBD. El objetivo primordial de un gestor es proporcionar eficiencia y seguridad a la hora de extraer o almacenar información en las BD. Los sistemas gestores de BBDD están diseñados para gestionar grandes bloques de información, que implica tanto la definición de estructuras para el almacenamiento como de mecanismos para la gestión de la información. En este tipo de sistemas los datos se centralizan en una base de datos común a todas las aplicaciones. 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. Cuando una aplicación modifica un dato, dicho dato la modificación será visible para el resto de aplicaciones. Una BD es un gran

almacén de datos que se define una sola vez; los datos pueden ser accedidos de forma simultánea por varios usuarios; están relacionados y existe un número mínimo de duplicidad; además en las BBDD se almacenarán las descripciones de esos datos, lo que se llama metadatos en el diccionario de datos. 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. Aplicación 1 Aplicación 2 Aplicación 3 Sistemas Gestores de Base de Datos 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. 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. Especificación de la estructura, el tipo de los datos, las restricciones y relaciones entre ellos mediante lenguajes de definición de datos. Toda esta información se almacena en el diccionario de datos, el SGBD proporcionará mecanismos para la gestión del diccionario 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:** para restablecer la información en caso de fallos en el sistema.] **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.

2.1 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 de vista:]

Nivel externo: Esta es la visión de los datos que poseen los usuarios del Sistema de Información.

2. Arquitectura de los Sistemas Gestores de Bases de Datos]

Nivel físico: Esta es la forma en la que realmente están almacenados los datos. Realmente la base de datos es la misma, pero se la puede observar desde estos dos puntos de vista. Al igual que una casa se la pueda observar pensando en los materiales concretos con los que se construye o bien pensando en ella con el plano en papel. En todo sistema de información digital, los usuarios ven los datos desde las aplicaciones creadas por los programadores. A ese nivel se manejan formularios, informes en pantalla o en papel,... Pero la realidad física de esos datos, tal cual se almacenan en los discos queda oculta a los usuarios. Esa forma de ver la base de datos está reservada a los administradores. Es el nivel físico el que permite ver la base de datos en función de cómo realmente se están almacenando en el ordenador, en qué carpeta, qué archivos se usan... En el caso de los Sistemas de Base de datos, se añade un tercer nivel, un tercer punto de vista, es el nivel conceptual. Ese nivel se sitúa entre el físico y el externo. En cada nivel se manejan esquemas de la base de datos, al igual que al construir una casa, los distintos profesionales manejan distintos tipos de planos (eléctricos, de albañilería, de tuberías de agua,...). Con lo cual una base de datos requiere diseñar al menos tres esquemas (en realidad son más).

En 1975, el comité ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee) propuso una arquitectura de tres niveles para los SGBD cuyo objetivo principal era el de separar los programas de aplicación de la BD física. En esta arquitectura el esquema de una BD se define en tres niveles de abstracción distintos:

Nivel interno o físico: el más cercano al almacenamiento físico, es decir, tal y como están almacenados en el ordenador. Describe la estructura física de la BD mediante un esquema interno. Este esquema se especifica con un modelo físico y describe los detalles de cómo se almacenan físicamente los datos: los archivos que contienen la información, su organización, los métodos de acceso a los registros, los tipos de registros, la longitud, los campos que

los componen, etcétera. Esta visión sólo la requiere el administrador/a. El administrador la necesita para poder gestionar más eficientemente la base de datos. Nivel externo o de visión: es el más cercano a los usuarios, es decir, es donde se describen varios esquemas externos o vistas de usuarios. Cada esquema describe la parte de la BD que interesa a un grupo de usuarios en este nivel se representa la visión individual de un usuario o de un grupo de usuarios. En realidad son varios. Se trata de la visión de los datos que poseen los usuarios y usuarias 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 dónde realmente se están almacenando los datos. Los esquemas externos los realizan las programadoras/es según las indicaciones formales de los y las analistas. Realmente cada aplicación produce un esquema externo diferente (aunque algunos pueden coincidir) o vista de usuario. El conjunto de todas las vistas de usuario es lo que se denomina esquema externo global. Nivel conceptual: describe la estructura de toda la BD para un grupo de usuarios mediante un esquema conceptual. Este esquema describe las entidades, atributos, relaciones, operaciones de los usuarios y restricciones, ocultando los detalles de las estructuras físicas de almacenamiento. Representa la información contenida en la BD. Se trata de un esquema teórico de los datos en el que figuran organizados en estructuras reconocibles del mundo real y en el que también aparece la forma de relacionarse los datos. Este esquema es el paso que permite modelar un problema real a su forma correspondiente en el ordenador. Este esquema es la base de datos de todos los demás. Como se verá más adelante, es el primer paso a realizar al crear una base de datos. En definitiva es el plano o modelo general de la base de datos. El esquema conceptual lo realizan diseñadores/as o analistas. Niveles de abstracción de la arquitectura ANSI.

3. Componentes de los Sistemas Gestores de Bases de Datos

Los SGBD son paquetes de software muy complejos que deben proporcionar una serie de servicios que van a permitir almacenar y explotar los datos de forma eficiente. Los componentes principales son los siguientes. Lenguajes de los SGBD Todos los SGBD ofrecen lenguajes e interfaces apropiadas para cada tipo de usuario: administradores, diseñadores, programadores de aplicaciones y usuarios finales. Los lenguajes van a permitir al administrador de la BD especificar los datos que componen la BD, su estructura, las relaciones que existen entre ellos, las reglas de integridad, los controles de acceso, las características de tipo físico y las vistas externas de los usuarios. Los lenguajes del SGBD se clasifican en: Lenguaje de definición de datos (LDD o DDL): se utiliza para especificar el esquema de la BD, las vistas de los usuarios y las

estructuras de almacenamiento. Es el que define el esquema conceptual y el esquema interno. Lo utilizan los diseñadores y los administradores de la BD. **3. Componentes de los Sistemas Gestores de Bases de Datos** Permite al diseñador de la base de datos crear las estructuras apropiadas para integrar adecuadamente los datos. Se dice que esta función es la que permite definir las tres estructuras de la base de datos (relacionadas con los tres niveles de abstracción).

- Estructura interna
- Estructura conceptual
- Estructura externa

Realmente esta función trabaja con metadatos. Los metadatos es la información de la base de datos que realmente sirve para describir a los datos. Es decir, Sánchez Rodríguez y Crespo son datos; pero Primer Apellido es un metadato. También son datos decir que la base de datos contiene Alumnos o que el dni lo forman 9 caracteres de los cuales los 8 primeros son números y el noveno un carácter en mayúsculas. La función de definición sirve pues para crear, eliminar o modificar metadatos. Para ello permite usar un 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

Lenguaje de manipulación de datos (LMD o DML): se utilizan para leer y actualizar los datos de la BD. Es el utilizado por los usuarios para realizar consultas, inserciones, eliminaciones y modificaciones. Los hay procedurales, en los que el usuario será normalmente un programador y especifica las operaciones de acceso a los datos llamando a los procedimientos necesarios. Estos lenguajes acceden a un registro y lo procesan. Las sentencias de un DML procedural están embebidas en un lenguaje de alto nivel llamado anfitrión. Las BD jerárquicas y en red utilizan estos DML procedurales. Mediante ese lenguaje se puede:

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

Actualmente se suele distinguir aparte la función de buscar datos en la base de datos (función de consulta). Para lo cual se proporciona un lenguaje de consulta de datos o DQL. Lenguaje de control de datos (LCD o DCL): Mediante esta función los administradores poseen mecanismos para proteger los datos; de modo que se permite a cada usuario ver ciertos datos y otros no; o bien usar ciertos recursos concretos de la base de datos y prohibir otros. Es decir simplemente permite controlar la seguridad de la base de datos. El lenguaje que implementa esta función es el lenguaje de control de datos o DCL. No procedurales son los lenguajes declarativos. En muchos SGBD se pueden introducir interactivamente instrucciones del LMD desde un terminal, también pueden ir embebidas en un lenguaje de programación de alto nivel. Estos lenguajes permiten especificar los datos a obtener en una consulta, o los datos a modificar, mediante sentencias sencillas. Las BD relacionales utilizan lenguajes no procedurales como SQL (Structured Query Language) o QBE (Query By Example). La mayoría de los SGBD comerciales incluyen lenguajes de cuarta generación (4GL) que permiten al usuario desarrollar aplicaciones de forma fácil y rápida,

también se les llama herramientas de desarrollo. Ejemplos de esto son las herramientas del SGBD ORACLE: SQL Forms para la generación de formularios de pantalla y para interactuar con los datos; SQL Reports para generar informes de los datos contenidos en la BD; PL/SQL lenguaje para crear procedimientos que interactúen con los datos de la BD.

3.1 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:

- 1) Directivos/as. Organizadores y coordinadores del proyecto a desarrollar y máximos responsables del mismo. 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.
- 2) Analistas. Son los encargados de controlar el desarrollo de la base de datos aprobada por la dirección. Normalmente son además los diseñadores de la base de datos (especialmente de los esquemas interno y conceptual) y los directores de la programación de la misma.
- 3) Administradores/as de las bases de datos. Encargados de crear el esquema interno de la base de datos, que incluye la planificación de copia de seguridad, gestión de usuarios y permisos y creación de los objetos de la base de datos.
- 4) Desarrolladores/as o programadores/as. Encargados de la realización de las aplicaciones de usuario de la base de datos.
- 5) Equipo de mantenimiento. Encargados de dar soporte a los usuarios en el trabajo diario (suelen incorporar además tareas administrativas como la creación de copias de seguridad por ejemplo o el arreglo de problemas de red por ejemplo).
- 6) Usuarios
- 7) 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.
- 8) 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.
- 9) 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.

El Administrador de la Base de Datos DBA El DBA tiene una gran responsabilidad ya que posee el máximo nivel de privilegios. Será el encargado de crear los usuarios que se conectarán a la BD. En la administración de una BD siempre hay que procurar que haya el menor número de administradores, a ser posible una sola persona. El objetivo principal de un DBA es garantizar que la BD cumple los fines previstos por la organización, lo que incluye una serie de tareas como:

Instalar SGBD en el sistema informático.] Crear las BBDD que se vayan a gestionar.] Crear y mantener el esquema de la BD.] Crear y mantener las cuentas de usuario de la BD.] Arrancar y parar SGBD, y cargar las BBDD con las que se ha de trabajar.] Colaborar con el administrador del S.O. en las tareas de ubicación, dimensionado y control de los archivos y espacios de disco ocupados por el SGBD.] Colaborar en las tareas de formación de usuarios.] Establecer estándares de uso, políticas de acceso y protocolos de trabajo diario para los usuarios de la BD.] Suministrar la información necesaria sobre la BD a los equipos de análisis y programación de aplicaciones. Efectuar tareas de explotación como: → Vigilar el trabajo diario colaborando en la información y resolución de las dudas de los usuarios de la BD. → Controlar en tiempo real los accesos, tasas de uso, cargas en los servidores, anomalías, etcétera. → Llegado el caso, reorganizar la BD. → Efectuar las copias de seguridad periódicas de la BD. → Restaurar la BD después de un incidente material a partir de las copias de seguridad. → Estudiar las auditorías del sistema para detectar anomalías, intentos de violación de la seguridad, etcétera. → Ajustar y optimizar la BD mediante el ajuste de sus parámetros, y con ayuda de las herramientas de monitorización y de las estadísticas del sistema. → En su gestión diaria, el DBA suele utilizar una serie de herramientas de administración de la BD. → Con el paso del tiempo, estas herramientas han adquirido sofisticadas prestaciones y facilitan en gran medida la realización de trabajos que, hasta no hace demasiado, requerían de arduos esfuerzos por parte de los administradores.

3.2 Estructura multicapa. El proceso que realiza un SGBD está en realidad formado por varias capas que actúan como interfaces entre el usuario y los datos. Fue el propio organismo ANSI (en su modelo X3/SPARC que luego se comenta) la que introdujo una mejora de su modelo de bases de datos 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 modela el funcionamiento de la base de datos en una sucesión de capas cuya finalidad es ocultar y proteger la parte interna de las bases de datos. Desde esta óptica para llegar a los datos hay que pasar una serie de capas que desde la parte más externa poco a poco van entrando más en la realidad física de la base de datos.

Modelo de referencia de las facilidades de usuario

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. Son, en definitiva, las páginas web y las aplicaciones con las que los usuarios manejan la base de datos. Permite abstraer la realidad de la base de datos a las usuarias y usuarios, mostrando la información de una forma más humana.

Capa de acceso a datos

La capa de acceso a datos es

la que permite comunicar a las aplicaciones de usuario con el diccionario de datos. Es un software (un driver o controlador en realidad) que se encarga traducir las peticiones del usuario para que lleguen de forma correcta a la base de datos y ésta pueda responder de forma adecuada.

Diccionario de datos Se trata del elemento que posee todos los metadatos. Gracias a esta capa las solicitudes de los clientes (que son conceptuales antes de llegar aquí) se traducen en instrucciones que hacen referencia al esquema interno de la base de datos. El diccionario de datos es el lugar donde se deposita información acerca de todos los datos que forman la BD. Es una guía en la que se describe la BD y los objetos que la forman. El diccionario contiene las características lógicas de los sitios donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenido y organización. Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información. En una BD relacional, el diccionario de datos proporciona información acerca de:

- La estructura lógica y física de la BD.
- Las definiciones de todos los objetos de la BD: tablas, vistas, índices, disparadores, procedimientos, funciones, etcétera.
- El espacio asignado y utilizado por los objetos.
- Los valores por defecto de las columnas de las tablas.
- Información acerca de las restricciones de integridad.
- Los privilegios y roles otorgados a los usuarios.
- Auditoría de información, como los accesos a los objetos.

Un diccionario de datos debe cumplir las siguientes características:

- Debe soportar las descripciones de los modelos conceptual, lógico, interno y externo de la BD.
- Debe estar integrado dentro del SGBD.
- Debe apoyar la transferencia eficiente de información al SGDB.
- La conexión entre los modelos interno y externo debe ser realizada en tiempo de ejecución.
- Debe comenzar con la reorganización de versiones de producción de la BD.
- Además debe reflejar los cambios en la descripción de la BD.
- Cualquier cambio a la descripción de programas ha de ser reflejado automáticamente en la librería de descripción de programas con la ayuda del diccionario de datos.
- Debe estar almacenado en un medio de almacenamiento con acceso directo para la fácil recuperación de información.

Núcleo El núcleo de la base de datos es la encargada de traducir todas las instrucciones requeridas y prepararlas para su correcta interpretación por parte del sistema. Realiza la traducción física de las peticiones.

Sistema operativo Es una capa externa al software SGBD pero es la única capa que realmente accede a los datos en sí. En realidad los SGBD no acceden directamente al disco, sino que piden al Sistema Operativo que lo haga.

3.3 Funcionamiento del SGBD

En esta arquitectura describe los datos a tres niveles de abstracción. En realidad los únicos datos que existen están a nivel físico almacenados en discos u otros dispositivos. Los SGBD basados en esta arquitectura permiten que cada grupo de usuarios haga referencia a su propio esquema externo. El SGBD debe de

transformar cualquier petición de usuario (esquema externo) a una petición expresada en términos de esquema conceptual, para finalmente ser una petición expresada en el esquema interno que se procesará sobre la BD almacenada. El proceso de transformar peticiones y resultados de un nivel a otro se denomina correspondencia o transformación, el SGBD es capaz de interpretar una solicitud de datos y realiza los siguientes pasos: 1) El usuario solicita unos datos y crea una consulta. 2) La Aplicación del usuario convierte esta consulta en un proceso realizado por el cliente del SGBD. 3) La consulta viaja a través de un medio (red). 4) El SGBD verifica y acepta el esquema externo para ese usuario. Se convierte el proceso del usuario en un “Proceso de Servidor” interno. 5) Transforma la solicitud al esquema conceptual. 6) Verifica y acepta el esquema conceptual. 7) El proceso lanzado por el usuario llama al SGBD indicando la porción de la base de datos que se desea tratar. 8) Transforma la solicitud al esquema físico o interno. El SGBD obtiene el esquema físico. 9) El SGBD traduce la llamada a los métodos de acceso del Sistema Operativo que permiten acceder realmente a los datos requeridos. 10) Selecciona la o las tablas implicadas en la consulta y ejecuta la consulta. El Sistema Operativo accede a los datos tras traducir las órdenes dadas por el SGBD. 11) Transforma del esquema interno al conceptual, y del conceptual al externo. Los datos pasan del disco a una memoria intermedia o buffer. En ese buffer se almacenarán los datos según se vayan recibiendo 12) Los datos pasan del buffer al área de trabajo del usuario (ATU) del proceso del usuario. Los pasos se repiten hasta que se envíe toda la información al proceso de usuario. 13) En el caso de que haya errores en cualquier momento del proceso, 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. 14) Finalmente, el usuario ve los datos solicitados. Para una BD específica sólo hay un esquema interno y uno conceptual, pero puede haber varios esquemas externos definidos para uno o para varios usuarios.

3.4 Formas de ejecución de un SGBD. Actualmente casi todos los Sistemas Gestores de Bases de Datos funcionan de manera semejante, en realidad hay tres posibilidades de funcionamiento: SGBDs monocapa: Es la más sencilla, pero la que tiene menos escalabilidad (posibilidad de crecer). El Sistema Gestor se instala en una máquina y los usuarios acceden directamente a esa máquina y ese Sistema Gestor. En estos sistemas no se accede de forma remota a la base de datos. SGBDs bicapa: 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. Hay dos posibilidades: → **Estructura Cliente-Servidor.** La base de datos está en un solo servidor al que acceden los clientes (incluso simultáneamente). → **Cliente Multiservidor.** En este caso los clientes acceden a un conjunto de servidores que distribuyen la base de datos. 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. Se usa cuando el número de clientes ha crecido mucho y un solo servidor no podría atender sus peticiones. SGBD de tres o más capas: Es una estructura de tipo cliente/servidor, pero en la que hay al menos una capa intermedia entre las dos. Esa capa se suele encargar de procesar las peticiones y enviarlas al SGBD con el que se comunica. Un ejemplo habitual es que la tercer capa sea un servidor web que evita que el cliente se conecte directamente al SGBD. Ese servidor web se encarga de traducir lo que el cliente realiza a una forma entendible por la base de datos. Esta forma de trabajar permite además que para usar una base de datos, baste un simple navegador al cliente. Puede haber más capas con otros fines. Estas capas siempre están entre el cliente y el servidor.

4. Tipos de Sistemas Gestores de Bases de Datos

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.

Modelos de datos utilizados en el desarrollo de una BD 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 esquemas señalados en la Ilustración anterior. 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 está más cerca del modelo físico, el que utiliza internamente el ordenador. El modelo conceptual es el más cercano al usuario, el lógico es el encargado de establecer el paso entre el modelo conceptual y el modelo físico del sistema.

4. Tipos de Sistemas Gestores de Bases de Datos

Algunos ejemplos de modelos conceptuales son:

- **Modelo Entidad Relación**
- **Modelo RM/T**
- **Modelo UML**

Ejemplos de modelos lógicos son:

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

A continuación se comentarán los modelos lógicos más importantes.

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

4.2 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 parente. 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.

4.3 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 que veremos.

4.4 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). **4.5 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 relationales 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 relationales (Oracle, SQL Server, Informix, ...) son objeto relationales. **4.6 Bases de datos NoSQL.** Bajo este nombre se agrupan las bases de datos (con arquitecturas muy diversas) pensadas para grabar los datos de manera veloz para así poder atender a miles y miles de peticiones. Es decir, es el modelo de las bases de datos que se utilizan en los grandes servicios de Internet (como twitter, Facebook, Amazon,...). La idea es que los datos apenas necesitan validarse y relacionarse y lo importante es la disponibilidad de la propia base de datos. El nombre NoSQL, hace referencia a que este modelo de bases de datos rompe con el lenguaje SQL (el lenguaje de las bases de datos relationales, las bases de datos dominantes hasta la actualidad) para poder manipular los datos con lenguajes de otro tipo.

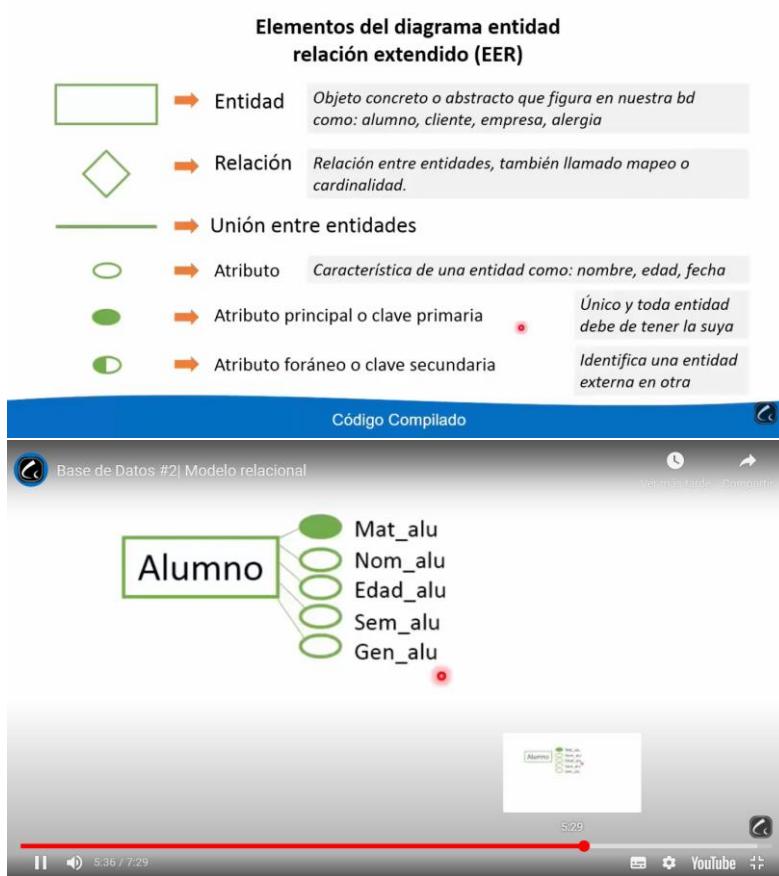
5. Diseño de bases de datos relationales

En este capítulo nos centraremos en comprender el modelo relaciones, entidad/relación y normalización de base de datos. Es muy importante que al finalizar este bloque sepan realizar a partir de un problema, el **Modelo de Pata de Gallo**, normalizado mínimamente en **1ra, 2da y 3ra Forma Normal**, y mejor aún si logran normalizar en **4ta y 5ta FN**.

No avancen al siguiente bloque si no lograron aprender aquello resaltado en negrita, ya que si se acarrea este problema no podrán comprender correctamente los bloques siguientes. Comencemos...

Diseño de bases de datos relationales

En este tema se estudia un aspecto fundamental de las bases de datos: su diseño. En las bases de datos se ha establecido un ciclo de desarrollo que consta de tres etapas de diseño: el diseño conceptual, el diseño lógico y el diseño físico. Mientras que las dos primeras etapas y el paso de una a otra están muy fundamentados, no ocurre lo mismo con la tercera, dado que las primeras son lo suficientemente abstractas como para no depender de ninguna implementación en concreto; sin embargo, el diseño físico depende del SGBD usado, y no hay reglas formales para llevarlo a cabo.



3.1. Etapas de diseño

La metodología de diseño de bases de datos relacionales se ha consolidado a lo largo de los años satisfaciendo las propiedades de generalidad (independencia de la plataforma hardware/software), calidad del producto (precisión, completitud y eficacia) y facilidad de uso.

Consta de las siguientes etapas:

- **1. Diseño conceptual.**

Su objetivo es definir las entidades y las relaciones entre ellos de forma abstracta, sin centrarse en ningún modelo lógico en concreto (como el relacional, el orientado a objetos, el jerárquico o el de red).

Herramienta: Modelo conceptual de datos. Se usa alguna variante del modelo entidad-relación para las bases de datos relacionales.

Resultado: Esquema conceptual de la base de datos.

- **2. Diseño lógico.**

Su objetivo es definir el esquema de la base de datos según el modelo que implementa el SGBD objetivo.

Herramienta: Modelo lógico de datos. Se usa el modelo lógico que implemente el sistema de gestión de bases de datos objetivo, pero es independiente de los aspectos físicos. Se usan técnicas formales para verificar la calidad del esquema lógico; la más usual es la normalización. En el modelo relacional se usan las tablas.

Resultado: Esquema lógico de la base de datos.

- **3. Diseño físico.**

Su objetivo es definir el esquema físico de la base de datos de forma que se den todas las instrucciones para que un DBA pueda implementar la base de datos sin ninguna ambigüedad. Se considera el rendimiento como un aspecto que no se ha tratado en las etapas anteriores.

Herramienta: Modelo físico de datos. Se consideran todos los detalles de la implementación física: organización de archivos e índices para el SGBD considerado.

Resultado: Esquema físico de la base de datos.

La siguiente figura muestra resumido el ciclo de desarrollo clásico de bases de datos:

3.2. Diseño conceptual

En este apartado se estudia el modelo entidad-relación que permite diseñar el esquema conceptual de una BD, y es muy adecuado para las BDs relacionales. Su resultado es un diagrama entidad-relación.

A lo largo de este apartado se usará un ejemplo de aplicación correspondiente a las necesidades de una secretaría de un centro docente, en la que hay alumnos matriculados en asignaturas y profesores que las imparten en ciertas aulas. Los alumnos consiguen una nota determinada en cada asignatura en que están matriculados.

3.2.1. Conceptos

- **Entidad:** Es el menor objeto con significado en una instancia.

Por ejemplo, para el diseño de una BD de la secretaría de un centro docente, el alumno con los siguientes datos:

DNI	=	01234567Z,
Nombre y apellidos	=	Manuel Vázquez Prieto,
Teléfono	=	91-12345678

Domicilio = Calle del Jazmín 7, 4 Izq.

Constituye una entidad. Igual sucede con cada asignatura concreta, cada profesor, etc.

En el caso del enfoque "clásico" correspondería a cada registro guardado en un fichero.

- **Atributo:** Es cada uno de los componentes que determinan una entidad.

Cada atributo tiene asociado un dominio: el conjunto de valores que puede tomar.

La entidad del ejemplo anterior viene determinada por los valores de sus atributos DNI, Nombre y Apellidos, Teléfono, Domicilio y COU.

En el enfoque clásico serían los campos de los registros.

- **Atributos monovalorados y multivalorados:** Los atributos multivalorados son los que pueden contener más de un valor simultáneamente, y monovalorados a los que sólo pueden contener un valor.

Por ejemplo, una persona puede tener varios números de teléfono (casa, trabajo, móvil) y puede que nos interese tenerlos todos. En este caso haremos de teléfono un atributo multivalorado.

- **Atributos simples y compuestos:** Un atributo es compuesto cuando puede descomponerse en otros componentes o atributos más pequeños, y simple en otro caso.

Por ejemplo, en el caso del domicilio puede que nos interese descomponerlo a su vez en calle, el número y la ciudad por separado.

- **Clave:** Es un atributo o conjunto de atributos cuyos valores identifican únicamente cada entidad.

Por ejemplo, DNI es un atributo clave del tipo de entidad Alumnos. Esto significa que los valores de la clave no se pueden repetir en el conjunto de entidades. En el ejemplo anterior ningún DNI se debería repetir en una instancia del tipo de entidad Alumnos.

El concepto de clave distingue tres claves diferentes:

Superclave. Es cualquier conjunto de atributos que pueden identificar únicamente a una tupla.

Clave candidata. Es el menor conjunto de atributos que puede formar clave. Puede haber varias en una tabla.

ClavePrimaria. Es la clave candidata que distingue el usuario para identificar únicamente cada tupla. Es importante en cuanto al aspecto del rendimiento, como se verá en el apartado dedicado al diseño físico.

- **Tipo de entidad.** Es el conjunto de entidades que comparten los mismos atributos (aunque con diferentes valores para ellos).

Por ejemplo, Alumnos será un tipo de entidad que representa cualquier conjunto de entidades en el que todas tengan como atributos

DNI, Nombre y Apellidos, ... y valores dentro de los dominios correspondientes. Asignaturas será otro tipo de entidad, etc.

Intuición: En el enfoque "clásico" sería el tipo de los registros.

Estamos describiendo el esquema de la base de datos.

- **Relación.** Es una correspondencia entre dos o más entidades. Se habla de relaciones binarias cuando la correspondencia es entre dos entidades, ternarias cuando es entre tres, y así sucesivamente.

Por ejemplo, la relación (José García, Bases de datos) es una relación entre dos entidades que indica que el alumno José García está matriculado en la asignatura Bases de datos.

- **Tipos de relación.** Representan a todas las posibles relaciones entre entidades del mismo tipo.

Por ejemplo, el tipo de relación matrícula relaciona el tipo de entidad alumnos con el tipo de entidad asignaturas.

Observaciones:

- Las relaciones también pueden tener atributos. Por ejemplo, Matrícula puede tener el atributo Nota que indica la nota que el alumno ha obtenido en una asignatura determinada.

Es posible que el mismo tipo de entidad aparezca dos o más veces en un tipo de relación. En este caso se asigna un nombre a cada papel que hace el tipo de entidad en el tipo de relación. Por ejemplo, algunos profesores tienen un supervisor, por lo que se define un tipo de relación Supervisa que relaciona profesores con profesores, el primero tendrá el papel de supervisor y el segundo de supervisado.

3.2.2. Diagramas entidad-relación (E-R)

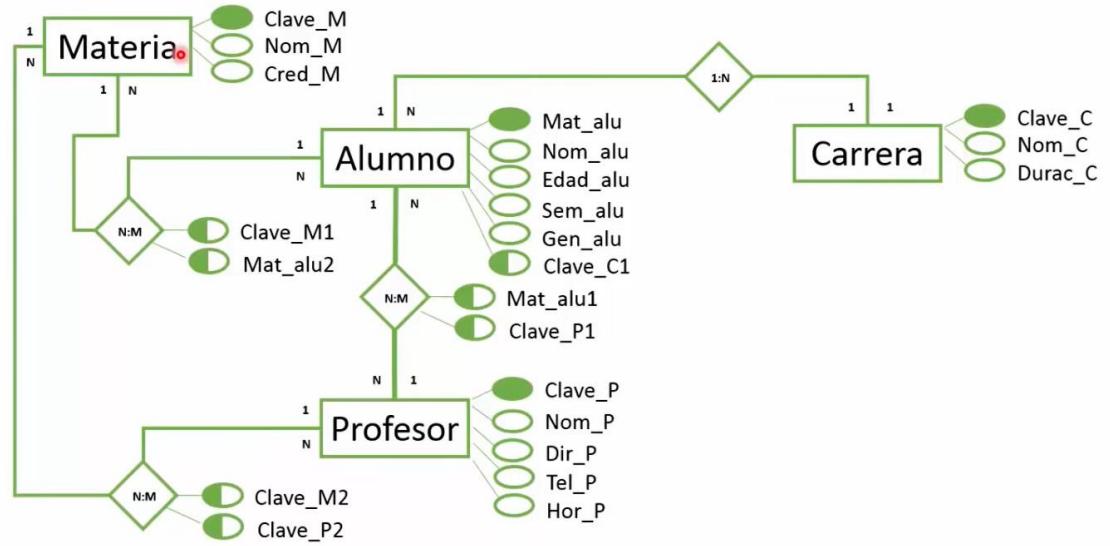
El diseño del modelo E-R a partir del análisis inicial no es directo. A un mismo análisis le corresponden muchos diseños "candidatos". Hay varios criterios, pero ninguno es definitivo. De un buen diseño depende:

Eficiencia: Es muy importante en las BD cuando se manejan grandes cantidades de datos.

Simplicidad del código: Se cometen menos errores.

Flexibilidad: Se refiere a que el diagrama sea fácil de modificar.

Los componentes básicos de los diagramas E-R son los atributos, los tipos de entidades y los tipos de relaciones.



Código Compilado



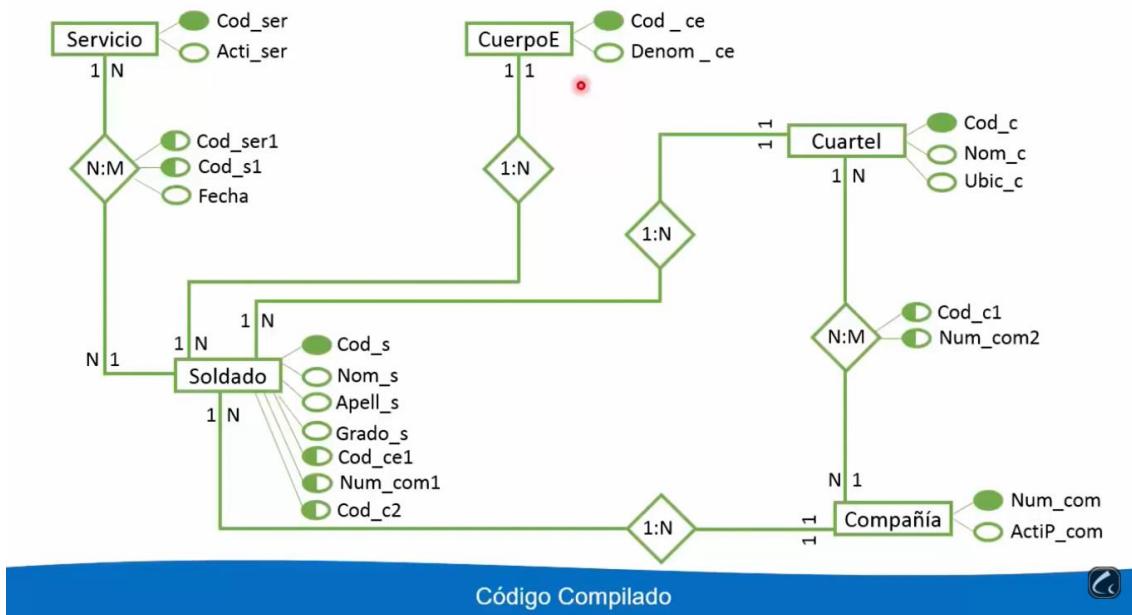
Los datos significativos a tener en cuenta son:

1. Un soldado se define por su código de soldado (único), su nombre, apellidos y su grado.
2. Existen varios cuarteles, cada uno se define por su código de cuartel, nombre y ubicación.
3. Hay Cuerpos del Ejército (Infantería, Artillería, Armada,...) y cada uno se define por un código de cuerpo y denominación.
4. Los soldados están agrupados en compañías, siendo significativa para cada una de éstas, el número de compañía y la actividad principal que realiza.
5. Los soldados realizan servicios (guardia, cuartelero,...) y se definen por el código de servicio y actividad.

Consideraciones de diseño:

- Un soldado pertenece a un único cuerpo y a una única compañía, durante todo el servicio militar.
- Pueden pertenecer soldados de diferentes cuerpos a una compañía, no habiendo relación directa entre compañías y cuerpos.
- Una compañía puede estar ubicada en varios cuarteles, y en un cuartel puede haber varias compañías. Eso sí, un soldado sólo está en un cuartel.
- Un soldado realiza varios servicios a lo largo del SM y un mismo servicio puede ser realizado por más de un soldado (con independencia de la compañía), siendo significativa la fecha de realización.





CodigoCompilado. (Febrero 2015). Base de Datos #3 | Ejercicio Diagrama Entidad Relación [Vídeo]. Youtube. <https://www.youtube.com/watch?v=u2bXiPJf9oQ>

CodigoCompilado. (Febrero 2015). Base de Datos #4 | Modelado de bd (sin normalizar) [Vídeo]. Youtube. <https://www.youtube.com/watch?v=te-i37IIfeU>

3.2.3. Elección de los tipos de entidad y sus atributos

De la especificación del problema de la secretaría se deduce que va ha haber un tipo de entidad alumnos, pero no cuáles son sus atributos. ¿Debe incluir las asignaturas en las que está matriculado? La respuesta es no y hacerlo así sería un error grave. Aparte de la idea 'filosófica' (cada asignatura es un objeto con significado propio, es decir, una entidad), al mezclar en una sola entidad alumnos y asignaturas cometemos cuatro errores:

1. Un alumno no tiene una asignatura asociada sino un conjunto de asignaturas asociadas. En cambio, sí tiene un DNI asociado, una

dirección asociada, etc. Por tanto las entidades serán de la forma

{DNI=12345678V, Nomb.Ape=Luis Martínez, Telf.=01234567,

Cod=MD, Título=Matemática Discreta, Créditos=9}

{DNI=12345678V, Nomb.Ape=Luis Martínez, Telf.=01234567,

Cod=IS, Título=Ingeniería del Software, Créditos=X}

{DNI=12345678V, Nomb.Ape=Luis Martínez, Telf.=01234567,

Cod=LPI, Título=Laboratorio de programación I, Créditos=X}

Hay redundancia en la información de alumnos: se repite en cada entidad.

2. Las asignaturas son siempre las mismas, con lo que por cada alumno que se matricula en la misma asignatura hay que repetir toda la información:

```
{ DNI=12345678V, Nomb.Ape=Luis Martínez, Telf.=01234567, ... ,  
Asignaturas={ {Cod=MD, Título=...}, {COD=IS,Título=...},  
{Cod=LPI,Título=...} } }  
{ DNI=0000001, Nomb.Ape=Eva Manzano, Telf.=01234567, ... ,  
Asignaturas={ {Cod=MD, Título=...}, {COD=IS,Título=...},  
{Cod=BDSI,Título=...} } }
```

En este caso hay redundancia en la información de las asignaturas.

3. Por cada profesor hay que apuntar las asignaturas que imparte. La información de las asignaturas debe estar por tanto relacionada con la de los profesores, pero ya está incluida con los alumnos. Hay que repetir la información de las asignaturas por lo que se consigue más redundancia.

4. No se pueden guardar los datos de una asignatura hasta que no se matricule un alumno en ella. Puede ser que en secretaría quieran meter los datos de las asignaturas antes de empezar el proceso de matrícula:

No pueden. Una solución sería incluirlos con los datos de los alumnos vacíos (nulos), lo cual no sería nada aconsejable. Los valores nulos se deben evitar siempre que sea posible.

Por tanto, hay que distinguir entre el tipo de entidad Alumnos y el tipo de entidad Asignaturas. Ambas se relacionarán mediante un tipo de relación Matrícula. Los restantes tipos de entidad serán: Profesores y Aulas.

Los atributos de cada tipo de entidad:

- Alumnos: DNI, Apellidos y Nombre, Domicilio, Teléfono y COU
- Asignaturas: Código, Título, Créditos
- Profesores: DNI, Apellidos y nombre, Domicilio y Teléfono

- Aulas: Edificio y Número

Aún nos falta un atributo, que es la **nota**: ¿Dónde se coloca? En Alumnos no porque un alumno tiene muchas notas, tantas como asignaturas en las que esté matriculado. En Asignaturas no porque en la misma asignatura están matriculados muchos alumnos. Va a ser un atributo del tipo de relación **matrícula**.

3.2.4. Elección de los tipos de relación

El primer tipo de relación es Matrícula que relaciona cada alumno con las asignaturas en las que está matriculado. Además, esta relación tiene un atributo, nota, que se asocia cada tupla de la relación. El segundo tipo de relación es Supervisa que va de Profesores a Profesores y que incluye los papeles Supervisor y Supervisado. La última es Imparte, que relaciona cada profesor con la asignatura que imparte y el aula en la que da esa asignatura. Aquí también surgen varias posibilidades:

- Hacer dos relaciones binarias. Por ejemplo, profesor con asignatura y asignatura con aula.
- Hacer una relación ternaria entre profesores, aulas y asignaturas.
- Hacer tres relaciones binarias Profesores-Asignaturas, Profesores-Aulas, Asignaturas-Aulas.

Diferencias:

1. En las opciones a) y c) se permite que un profesor imparta una asignatura que aún no tiene aula (esto puede ser deseable o no).

2. El problema de a) es:

Profesores-Asignaturas = { ({DNI=6666666, NombreYape=Rómulo Melón},{Cod=MD,...}) ...}

Asignaturas-Aulas = {
({Cod=MD....},{Edif=Mates, NumAula=S103}), ({Cod=MD....},{Edif=Biológicas, =104}) }

Estas relaciones son posibles, hay más de un curso de primero y por eso la misma asignatura se imparte en varias aulas. Ahora bien, Don Rómulo quiere saber en qué aula debe dar su clase de Matemática discreta, y para ello pregunta en secretaría. ¿Qué se le contesta?

Sin embargo, con la propuesta b) sí se le puede asignar a cada profesor un aula concreta para cada una de sus asignaturas.

El problema de c) sigue siendo el mismo:

Profesores-asignaturas = { ({DNI=6666666, NombreYape=Rómulo Melón,...}, {Cod=MD,...}), ({DNI=66666666, NombreYape=Rómulo Melón,...}, {Cod=IS,...}), ... }

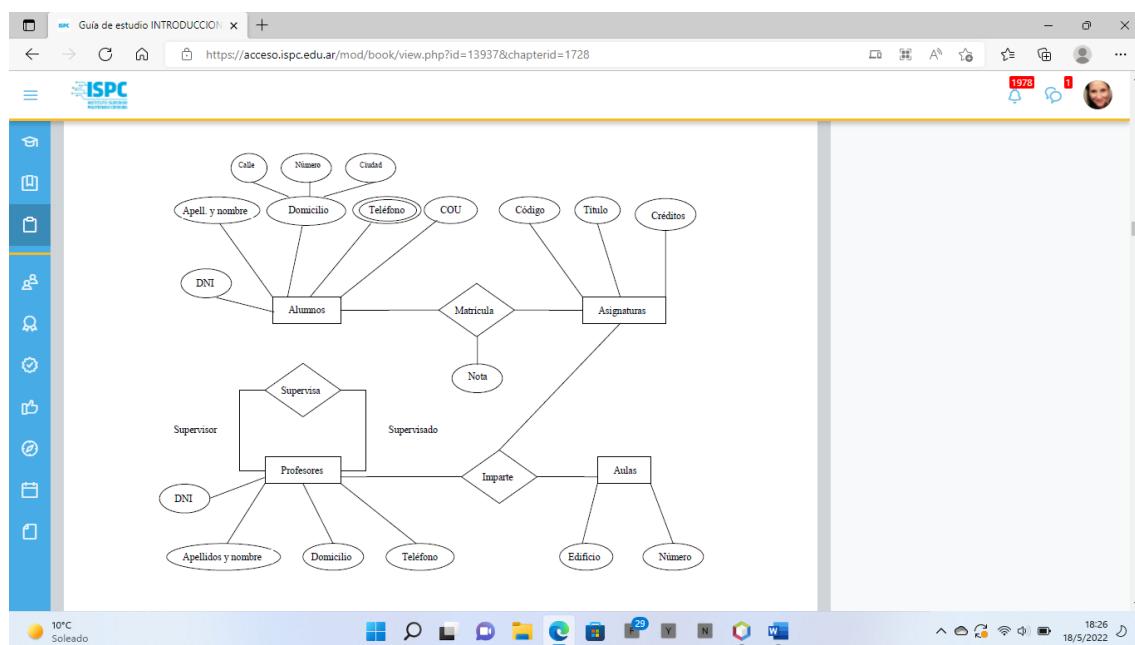
Asignaturas-Aulas = { ({Cod=MD...}, {Edif=Mates, NumAula=S103}), ({Cod=MD...}, {Edif=Biológicas, =104}), ({Cod=IS...}, {Edif=Mates, NumAula=S103}), ({Cod=IS...}, {Edif=Biológicas, NumAula=104}) }

Profesores-Aulas = {({DNI=6666666, NombreYape=Rómulo Melón,...}, {Edif=Mates, NumAula=S103}), ({DNI=6666666, NombreYape=Rómulo Melón,...}, {Edif=Biológicas, NumAula=104}), ... }

Don Rómulo sabe que da 2 asignaturas, cada una en un aula, pero sigue sin saber a dónde tiene que ir a dar MD.

Conclusión: Una relación ternaria tiene en general más información que 3 binarias.

El diagrama entidad-relación del ejemplo quedaría como se ilustra a continuación:



3.2.5. Adelanto de las restricciones de integridad

Con los elementos anteriores tenemos una primera aproximación a los diagramas E-R en la que tenemos definidos los elementos principales de los diagramas. Sin embargo, en el modelo E-R también se pueden definir numerosas restricciones de integridad sobre los tipos de entidades y tipos de relaciones.

Por ejemplo, en la relación Supervisa un profesor puede tener a lo sumo un supervisor, pero el diagrama anterior permite

The screenshot shows a web browser window with the URL <https://acceso.ispc.edu.ar/mod/book/view.php?id=13937&chapterid=1728>. The page title is "Guía de estudio INTRODUCCIÓN". On the left, there is a vertical sidebar with icons for navigation and user profile. The main content area has a blue header "3.2.5. Adelanto de las restricciones de integridad". Below it, a text block discusses how previous elements provide a first approximation of E-R diagrams, mentioning that while they define the main elements of diagrams, they also allow for numerous integrity restrictions on entity types and relationship types. It then provides an example from the "Supervisa" relationship where a professor can have at most one supervisor, but the diagram shown earlier did not restrict this. A table follows:

SUPERVISOR	SUPERVISADO
{DNI=666666...}, {DNI=000001...},	{DNI=444444...}

Below the table, a note states that this combination should not be a valid instance of the relationship. It explains that to avoid such situations, integrity restrictions are introduced, which are properties associated with an entity type or relationship type. Valid instances are those that satisfy all associated restrictions. The restrictions are part of the database design, equal to the entity and relationship types. The DBMS checks if the instance satisfies these usual restrictions. In the previous example, once the restriction was included, the DBMS would not allow the second row to be inserted.

Que no debería ser una instancia válida de la relación.

Para evitar estas situaciones se introducen las denominadas restricciones de integridad de la base de datos. Las restricciones de

integridad (o simplemente restricciones) son propiedades que se asocian a un tipo de entidad o de relación. Las instancias válidas del tipo de entidad o relación son aquellas en las que se verifique el conjunto de restricciones asociadas. Las restricciones son parte del diseño de la base de datos igual que los tipos de entidades o de relaciones. Los SGBD se encargan de comprobar que la instancia verifica las restricciones más usuales. En el ejemplo anterior, una vez incluida la restricción, el SGBD no nos permitiría insertar la segunda fila.

Un tipo de restricción de integridad que interesa conocer en esta etapa es la restricción de clave. Una restricción de clave consiste en imponer que un conjunto de atributos sea el que defina únicamente a una fila de un tipo de entidades. Por ejemplo, en el tipo de entidades Alumnos se puede elegir DNI para identificar a un alumno en concreto, pero no sería conveniente usar el atributo Nombre y apellidos porque es muy posible encontrar a dos personas con los mismos nombres y apellidos. Por motivos de eficiencia conviene que el número de atributos elegidos sea el menor posible. A veces, es posible elegir varios conjuntos de atributos que contengan el mismo número de atributos, pero se suele escoger uno de estos conjuntos como el representativo, que se denomina clave primaria. Por

ejemplo, si hubiese un atributo Número de la Seguridad Social, se podría usar también como clave. Todos estos conjuntos con el menor y mínimo número de atributos se denominan colectivamente como claves candidatas. Otro tipo de restricción de integridad importante ahora son las restricciones de cardinalidad. La idea de este tipo de restricción se puede entender con el siguiente ejemplo: supongamos que deseamos tener información sobre

el país de nacimiento de personas. Habría una relación Nacida entre las entidades Personas y Países, como se muestra a continuación:

de nacimiento de personas. Habría una relación Nacida entre las entidades Personas y Países, como se muestra a continuación:

```
graph LR; Personas[Personas] -->|Nacida| Paises[Países]; Personas -->|Nacida| Paises
```

Además, deseamos expresar que, si bien muchas personas pueden haber nacido en un país, una persona en concreto sólo puede haber nacido en un país. Esto se expresa en el diagrama E-R con una flecha que indica que una persona ha nacido en un país en concreto. Leyendo la relación en el sentido contrario dirímos que en un país pueden haber nacido muchas personas (el segmento que une Nacida con Personas no lleva flecha). (Esta restricción de cardinalidad también la podemos encontrar en el ejemplo de la secretaría, en la relación Supervisa, como se ha visto en el inicio de este apartado).

```
graph LR; Personas[Personas] -->|Nacida| Paises[Países]
```

El último tipo de restricción de integridad que interesa introducir ahora es la participación total. Se refiere a que podamos encontrar cada entidad de un tipo de entidad en la relación que lo liga con otro u otros. Por ejemplo, en la base de datos de la secretaría, si hay un alumno en concreto dado de alta en la base de datos es porque se debe haber matriculado de alguna asignatura. Es decir, cada alumno definido en el tipo de entidad Alumnos debemos encontrarlo en la relación Matrícula, relacionado con la asignatura en la que esté matriculado. En el diagrama E-R se expresa con una línea doble, como se ve a continuación:

Además, deseamos expresar que, si bien muchas personas pueden haber nacido en un país, una persona en concreto sólo puede haber nacido en un país. Esto se expresa en el diagrama E-R con una flecha que indica que una persona ha nacido en un país en concreto. Leyendo la relación en el sentido contrario dirímos que en un país pueden haber nacido muchas personas (el segmento que une Nacida con Personas no lleva flecha). (Esta restricción de cardinalidad también la podemos encontrar en el ejemplo de la secretaría, en la relación Supervisa, como se ha visto en el inicio de este apartado).

El último tipo de restricción de integridad que interesa introducir ahora es la participación total. Se refiere a que podamos encontrar cada entidad de un tipo de entidad en la relación que lo liga con otro u otros. Por ejemplo, en la base de datos de la secretaría, si hay un alumno en concreto dado de alta en la base de datos es porque se debe haber matriculado de alguna asignatura. Es decir, cada alumno definido en el tipo de entidad Alumnos debemos encontrarlo en la relación Matrícula, relacionado con la asignatura en la que esté matriculado. En el diagrama E-R se expresa con una línea doble, como se ve a continuación:

Guía de estudio INTRODUCCION

https://acceso.ispc.edu.ar/mod/book/view.php?id=13937&chapterid=1728

ISPC

1978 1

personas ha nacido en un país en concreto. Leyendo la relación en el sentido contrario dirímos que en un país pueden haber nacido muchas personas (el segmento que une Nacida con personas no lleva flecha). (Esta restricción de cardinalidad también la podemos encontrar en el ejemplo de la secretaría, en la relación Supervisa, como se ha visto en el inicio de este apartado).

```

    graph LR
        Personas[Personas] --> Nacida{Nacida}
        Nacida --> Paises[Paises]
    
```

El último tipo de restricción de integridad que interesa introducir ahora es la participación total. Se refiere a que podamos encontrar cada entidad de un tipo de entidad en la relación que lo liga con otro u otros. Por ejemplo, en la base de datos de la secretaría, si hay un alumno en concreto dado de alta en la base de datos es porque se debe haber matriculado de alguna asignatura. Es decir, cada alumno definido en el tipo de entidad Alumnos debemos encontrarlo en la relación Matrícula, relacionado con la asignatura en la que esté matriculado. En el diagrama E-R se expresa con una línea doble, como se ve a continuación:

```

    graph LR
        Alumnos[Alumnos] <--> Matricula{Matrícula}
        Matricula --> Asignaturas[Asignaturas]
    
```

10°C Soleado 18:34 18/5/2022

3.3. Diseño lógico

El diseño lógico es la segunda etapa del diseño de bases de datos en general y de las bases de datos relacionales en particular. En nuestro caso, las BD relacionales, el resultado de esta etapa es un esquema relacional basado en un modelo relacional. En este apartado se describirá en primer lugar el modelo relacional y en segundo lugar cómo pasar de un esquema entidad-relación a un esquema relacional.

3.3.1. El modelo relacional

Este modelo fue creado por Codd a principios de los 70 al que dotó de una sólida base teórica. Actualmente está implementado en la mayoría de los SGBD usados en la empresa. El concepto principal de este modelo es la relación o tabla. Es importante no confundir la tabla con las relaciones del modelo E-R. Aquí las relaciones se aplican tanto a tipos de relaciones como a tipos de entidades. En este modelo no se distingue entre tipos de entidades y tipos de relaciones porque la idea es que una relación o tabla expresa la relación entre los tipos de valores que contiene.

A continuación se introducen los conceptos de este modelo:

- **Entidad.** Igual que en el modelo E-R. También se les llama tuplas o filas de la relación.
- **Atributo.** Igual que en el modelo E-R. También se le llaman campos o columnas de la relación. El dominio de los atributos tiene que ser simple: no se admiten atributos multivvalorados ni compuestos.
- **Esquema de una relación.** Viene dado por el nombre de la relación y una lista de atributos. Es el tipo de entidad.
- **Conjunto de entidades.** Relación o tabla.

Por ejemplo, el tipo de entidad Alumnos del modelo E-R del apartado del diseño conceptual se representaría como la siguiente relación:

Alumnos (DNI, NombreYApellidos, Domicilio, Teléfono, COU)

El orden de los atributos en la lista no importa. Lo fijamos porque nos viene bien para representarlo como tabla, pero cualquier permutación es válida.

- Clave. Igual que en el modelo E-R. Hay que darse cuenta que en el modelo relacional todas las tablas deben tener claves, y que algunas tablas van a representar relaciones del esquema E-R.

- Instancia de una relación. Son conjuntos de entidades. Cada entidad se representa como una tupla. Cada componente de la tupla corresponde con el valor del atributo correspondiente, según el orden enunciado en el esquema de la relación.

Por ejemplo, una instancia de la relación Alumnos sería:

{ (01234567Z, Manuel Vázquez Prieto, Calle del Jazmín 7 4 Izq, 91-12345678, COU = Sí), }

En el modelo relacional no se representan diagramas del esquema de la BD. Por el contrario, el esquema relacional se representa por los conjuntos de entidades como hemos visto antes (nombre de la tabla y entre paréntesis el nombre de sus atributos). Las instancias de una relación se representan con tablas, como se muestra en el ejemplo del conjunto de entidades Alumnos:

The screenshot shows a web browser window titled "Guía de estudio INTRODUCCIÓN". The URL is <https://acceso.ispc.edu.ar/mod/book/view.php?id=13937&chapterid=1728>. The page content discusses the relational schema for Alumnos, mentioning keys and instances. It includes a table representing the Alumnos relation:

DNI	NombreyApellidos	Domicilio	Teléfono	COU
01234567Z	Manuel Vázquez Prieto	Calle del Jazmín 7 4 Izq	9112345678	Sí
...				

3.3.2. Paso de un esquema E-R a un esquema relacional

A continuación se describe la forma de traducir cada uno de los elementos que aparecen en el modelo E-R a un

10°C Soleado 1978 1 18/5/2022 18:39

3.3.2. Paso de un esquema E-R a un esquema relacional

A continuación se describe la forma de traducir cada uno de los elementos que aparecen en el modelo E-R a un esquema relacional.

Tipos de entidades

Para cada tipo de entidad se crea una relación con el mismo nombre y conjunto de atributos. Por ejemplo, en el caso de la BD de secretaría los tipos de entidades dan lugar a las siguientes relaciones:

Alumnos(DNI, Apellidos y Nombre, Domicilio, Teléfono, COU)
Asignaturas(Código, Título, Créditos)
Profesores(DNI, Apellidos y nombre, Domicilio, Teléfono)
Aulas(Edificio, Número)

Tipos de relaciones

Para cada tipo de relación R se crea una relación que tiene como atributos:

Los atributos de la clave primaria de cada tipo de entidad que participa en la relación.

Los atributos de la propia relación.

En ocasiones hay que renombrar atributos para evitar tener varios con el mismo nombre.

Como ejemplo, en el caso de la BD de secretaría los tipos de relación dan lugar a las siguientes relaciones:

Matrícula(DNI, Código, Nota)
Supervisa(DNISupervisor, DNISupervisado)
Imparte(DNI, Código, Edificio, NumAula)

Obsérvese que en la relación Supervisa se han renombrado los atributos DNI para indicar el papel de cada uno de ellos en la relación y además evitar que se use el mismo nombre para más de un atributo,

Claves

Hay dos casos:

1. La relación proviene de un tipo de entidad en el esquema E-R. La clave es la misma que la del tipo de entidad.

Por ejemplo:

Alumnos(DNI, Apellidos y nombre, Domicilio, Teléfono, COU)

2. La relación proviene de un tipo de relación en el esquema E-R. Si la relación R es un tipo de relación entre varios tipos de entidades se va a construir una relación bajo el modelo E-R a partir de R con los atributos que forman clave primaria en todas las entidades participantes más los propios de R. De ellos formarán clave primaria las claves primarias de cada una de las entidades participantes.

Por ejemplo:

Matrícula(DNI, Código, Nota)

Restricciones de cardinalidad

Es posible incorporar este tipo de restricciones de integridad cuando se desean indicar relaciones una a una, una a varias y varias a varias. (En el ejemplo de la secretaría tenemos la relación Supervisa, del tipo una a varias). A continuación se muestran estos casos para relaciones binarias, siendo c1 y c2 las claves primarias de E1 y E2, respectivamente:

a) Una a una

The screenshot shows a web page from 'Guía de estudio INTRODUCCIÓN' at <https://acceso.ispc.edu.ar/mod/book/view.php?id=13937&chapterid=1728>. The page discusses binary relationships where each entity in E1 is related to exactly one entity in E2. It includes a diagram showing two rectangles (E1 and E2) connected by a diamond-shaped relationship R with a double-headed arrow, and a text explanation about primary keys c1 and c2. Below the diagram are four tables: 'Hombres' (DNI: 1, 2), 'Mujeres' (DNI: 3, 4), 'Casados' (DNIH, DNIM: (1,3), (2,4)), and 'Casados' (DNIH, DNIM: (1,4), (2,3)). A note below the tables states that no column value repeats between the two possibilities of R, so either A or B could be a key. The browser interface shows a sidebar with various icons, a toolbar at the top, and a status bar at the bottom indicating weather (10°C Soleado) and date (18/5/2022).

De los atributos de la traducción de R al modelo relacional podemos escoger como clave o bien c1 o bien c2, de ellas la más adecuada será la que tenga menos atributos. Esto es posible porque cada entidad de E1 está relacionada con sólo una de E2 y viceversa, por lo que no es posible que la misma entidad de E1 o de E2 aparezca más de una vez en R. Por tanto, cualquiera de sus claves primarias puede ser clave de R.

Por ejemplo, supongamos que:

Hombres(DNI)= {1,2}, Mujer(DNI)= {3,4}, la relación Casado(DNIH, DNIM) sólo puede contener {(1,3), (2,4)} o {(1,4), (2,3)} o, en representación tabular:

Dado que no se repite ningún valor de las columnas de las dos posibilidades de R, tanto A como B podrían ser clave. Así, tendríamos

las alternativas siguientes para la relación

Casados: Casados(DNIH, DNIM) y Casados(DNIH, DNIM).

b) Una a varias

Dado que no se repite ningún valor de las columnas de las dos posibilidades de R, tanto A como B podrían ser clave. Así, tendríamos las alternativas siguientes para la relación Casados: Casados(DNIH, DNIM) y Casados(DNIH, DNIM).

b) Una a varias

```
graph LR; E1[E1] --> R{R}; R --> E2[E2]
```

En este caso, la clave de R debe ser la clave primaria de c2. Es decir, en la relación R no puede aparecer repetido ningún valor de E2.

En el ejemplo de las personas nacidas en países, tendríamos una instancia de la relación Nacida:

IDPersona	País
1	España
2	España
3	Portugal

En donde vemos que los valores de los países se pueden repetir en la relación, pero no el identificador de la persona porque, si así fuese, significaría que la misma persona ha nacido en diferentes países. Otro ejemplo es el de la secretaría, en el que la relación Supervisa quedaría Supervisa(DNISupervisor, DNISupervisado).

En este caso, la clave de R debe ser la clave primaria de c2. Es decir, en la relación R no puede aparecer repetido ningún valor de E2.

En el ejemplo de las personas nacidas en países, tendríamos una instancia de la relación Nacida:

En donde vemos que los valores de los países se pueden repetir en la relación, pero no el identificador de la persona porque, si así

fuese, significaría que la misma persona ha nacido en diferentes países. Otro ejemplo es el de la secretaría, en la que la relación

Supervisa quedaría Supervisa(DNISupervisor, DNISupervisado).

c) Varias a varias

En donde vemos que los valores de los países se pueden repetir en la relación, pero no el identificador de la persona porque, si así fuese, significaría que la misma persona ha nacido en diferentes países. Otro ejemplo es el de la secretaría, en la que la relación Supervisa quedaría Supervisa(DNISupervisor, DNISupervisado).

c) Varias a varias

```
graph LR; E1[E1] --> R{R}; R --> E2[E2]
```

Éste es el caso más general en el que no se puede imponer ninguna restricción además de la ya indicada de clave. Por ejemplo, Los tres casos anteriores se referían a relaciones binarias. En el caso de que se trate de relaciones n-arias, supongamos que la relación proviene de un tipo de relación R entre tipos de entidad E1, E2, ..., Ek, entonces:

- Si todos participan con cardinalidad varios en R, entonces una clave es la unión de las claves de E1, E2, ..., Ek.
- Si algunos tipos de entidad participan con cardinalidad una en R, entonces uno de ellos puede ser excluido de la superclave.

3.3.3. Restricciones de integridad

Éste es el caso más general en el que no se puede imponer ninguna restricción además de la ya indicada de clave. Por ejemplo,

Los tres casos anteriores se referían a relaciones binarias. En el caso de que se trate de relaciones n-arias, supongamos que la relación proviene de un tipo de relación R entre tipos de entidad E1, E2, ..., Ek, entonces:

- Si todos participan con cardinalidad varios en R, entonces una clave es la unión de las claves de E1, E2, ..., Ek.
- Si algunos tipos de entidad participan con cardinalidad una en R, entonces uno de ellos puede ser excluido de la superclave.

3.3.3. Restricciones de integridad

Hemos visto cómo se traducen las claves de los tipos de entidades y cómo aparecen claves en la traducción de los tipos de relaciones. Sin embargo, no es el único tipo de restricciones de integridad que aparece automáticamente al traducir un esquema E-R en otro relacional. Hay dos: **restricciones de integridad referencial** y **restricciones de participación total**.

Las claves y las restricciones de integridad referencial son características que se expresan directamente en la práctica totalidad de los SGBD relacionales. Estos sistemas se ocupan automáticamente de que no se violen estas restricciones. Sin embargo, no ocurre lo mismo con las de participación total y otras restricciones, como se verá en el tema dedicado a las restricciones de integridad.

Restricciones de integridad referencial

Al traducir un tipo de relación R, en cualquier instancia de R se debe cumplir que los valores de los atributos que hereda de una entidad (de su clave primaria) deben aparecer previamente en el conjunto de entidades. En el ejemplo de los hombres y mujeres casados está claro que en la relación Casados no puede aparecer un valor del DNI de un hombre o de una mujer que no estén previamente en el conjunto de entidades Hombres o Mujeres.

Es decir:

The screenshot shows a Microsoft Edge browser window with the URL <https://acceso.ispc.edu.ar/mod/book/view.php?id=13937&chapterid=1728>. The page contains a diagram illustrating referential integrity constraints between three entities: Hombres, Mujeres, and Casados. The Hombres entity has attributes DNI (values 1, 2). The Mujeres entity has attributes DNI (values 3, 4). The Casados entity has attributes DNIH (value 1) and DNIM (value 5). A callout box labeled '¡Incorrecto!' points to the entry '5' in the DNIM column of the Casados table.

restrictiones. Sin embargo, no ocurre lo mismo con las de participación total y otras restricciones, como se verá en el tema dedicado a las restricciones de integridad.

Restricciones de integridad referencial

Al traducir un tipo de relación R, en cualquier instancia de R se debe cumplir que los valores de los atributos que hereda de una entidad (de su clave primaria) deben aparecer previamente en el conjunto de entidades. En el ejemplo de los hombres y mujeres casados está claro que en la relación Casados no puede aparecer un valor del DNI de un hombre o de una mujer que no estén previamente en el conjunto de entidades Hombres o Mujeres.

Es decir:

Hombres		Mujeres		Casados	
DNI		DNI		DNIH	DNIM
1		3		1	5
2		4		2	4

En esta instancia se estaría dando la idea de que el hombre con DNI 1 está casado con la mujer con DNI 5. Pero no sabemos nada de esta mujer dado que no está en el conjunto de entidades Mujeres (hay que considerar que este tipo de entidad tendría otros muchos atributos, como el nombre y apellidos de la mujer, su domicilio, etc., que podrían ser útiles para la aplicación de la base de datos). No obstante, lo que sí es posible que algunos hombres o algunas mujeres no estén casados entre sí.

Es necesario, por tanto imponer una restricción de integridad referencial entre los atributos clave heredados de una entidad con las clave de esa entidad. En este ejemplo, se podría expresar:

Casados.DNIH ⊆ Hombres.DNI
Casados.DNIM ⊆ Mujeres.DNI

10°C Soleado 18:50 18/5/2022

En esta instancia se estaría dando la idea de que el hombre con DNI 1 está casado con la mujer con DNI 5. Pero no sabemos nada de esta mujer dado que no está en el conjunto de entidades Mujeres (hay que considerar que este tipo de entidad tendría otros muchos atributos, como el nombre y apellidos de la mujer, su domicilio, etc., que podrían ser útiles para la aplicación de la base de datos). No obstante, lo que sí es posible que algunos hombres o algunas mujeres no estén casados entre sí.

Es necesario, por tanto imponer una restricción de integridad referencial entre los atributos clave heredados de una entidad con las clave de esa entidad. En este ejemplo, se podría expresar:

$$\text{Casados.DNIH} \subseteq \text{Hombres.DNI}$$

$$\text{Casados.DNIM} \subseteq \text{Mujeres.DNI}$$

Es decir, los valores de DNI que aparecen en el atributo DNIH de Casados deben aparecer previamente en el atributo DNI de Hombres (y lo análogo para las mujeres).

Restricciones de participación total

Cuando cada valor de un tipo de entidad debe aparecer en un tipo de relación, como Alumnos en Matrícula, significa que, además de la restricción de integridad referencial comentada en el apartado anterior, se debe cumplir que todo valor de DNI en Alumnos debe aparecer en el atributo DNI de Matrícula. Esto se puede expresar:

$$\text{Alumnos.DNI} \subseteq \text{Matrícula.DNI}$$

Por otra parte, dado que la restricción de integridad referencial sobre esta tabla arroja:

Matrícula.DNI \subseteq Alumnos.DNI

Llegamos a la conclusión de que:

Matrícula.DNI = Alumnos.DNI

Es decir, si aparece un valor de DNI en Matrícula, también debe aparecer en el atributo DNI de Alumnos, y viceversa.

3.3.4. Cuestiones de diseño

En ocasiones es posible combinar dos o más tablas en una sola. Generalmente se combinan por motivos de rendimiento. Por ejemplo, dado el ejemplo de personas nacidas en países:

The screenshot shows a web page from the ISPC website. The title is "Guía de estudio INTRODUCCIÓN". The main content is titled "3.3.4. Cuestiones de diseño". It discusses combining tables for performance reasons, specifically using the example of people born in countries. An E-R diagram is shown with entities "Personas" and "Países", and a relationship "Nacida" connecting them. "Personas" has attributes "Apell" and "DNI". "Países" has attribute "Nombre". Below the diagram, it says "La traducción de este esquema E-R al relacional sería:" followed by the relational schema: "Personas(DNI, Apell)" and "Países(Nombre)". It also notes that the first two tables can be combined into a new schema: "Personas(DNI, Apell, PaísNac)" and "Países(Nombre)". A note at the bottom states: "Un inconveniente de esta combinación es que, dado que no se exige participación total de Personas en Nacida, no tendremos información del país de nacimiento de algunas personas, y en la tabla Personas va a aparecer un valor NULL (nulo) en el atributo PaísNac, que indica que no se dispone de esa información." The browser interface includes a sidebar with navigation icons, a header with a user profile, and a taskbar at the bottom.

Un inconveniente de esta combinación es que, dado que no se exige participación total de Personas en Nacida, no tendremos información del país de nacimiento de algunas personas, y en la tabla Personas va a aparecer un valor NULL (nulo) en el atributo PaísNac, que indica que no se dispone de esa información.

El valor NULL es un valor que puede contener cualquier atributo, y lo soportan todos los SGBD. Es un valor especial que se debe tratar con cuidado y, en general, evitar, porque puede representar muchas cosas, tales como:

- Ausencia de información.
- Este atributo no se aplica o no tiene sentido para esta entidad en concreto.
- Valor desconocido.

Además causan problemas a la hora de realizar consultas sobre la base de datos. Por otra parte, ningún atributo que forme parte de una clave puede tomar el valor NULL.

3.4. Diseño físico

El objetivo del diseño físico es la generación del esquema físico de la base de datos en el modelo de datos que implementa el SGBD. Esto incluye la definición sobre el SGBD de las tablas con sus campos, la imposición de todas las restricciones de integridad y la definición de índices.

Los índices son estructuras de datos implementadas con ficheros que permiten un acceso más eficaz a los datos. Se organizan con respecto a uno o más campos (los denominados campos clave del índice, que no hay que confundir con el concepto de clave del modelo entidad-relación y relacional) y guardan sólo la información del valor de la clave y la dirección física a partir

de la cual se pueden encontrar registros con ese valor.

Los índices son secuencias de registros que tienen dos campos: el valor de la clave y la dirección física del registro del fichero de datos en donde se puede encontrar una tupla con ese valor.

El rendimiento (el tiempo que se tarda en realizar una determinada operación) de una base de datos depende fundamentalmente de las operaciones de lectura y escritura en disco. Como las tablas generalmente no caben todas en la memoria principal, en general es necesario leer los datos del disco. Cuantos menos datos se lean o escriban en disco mejor será el rendimiento. Los índices permiten disminuir el tiempo de entrada/salida a disco.

Internamente, cuando el SGBD necesita buscar un registro según un valor de un campo (por ejemplo, un número de DNI) sobre el que se ha definido un índice para resolver alguna consulta, busca el valor en el índice, consulta la dirección del registro adjunto y a continuación busca en el fichero de datos (donde se almacenan los datos de la tabla correspondiente) el registro.

Esto es más rápido que hacer una búsqueda secuencial en el fichero de datos. Por un lado porque los índices no requieren en general una exploración secuencial de sus registros hasta encontrar el valor deseado, sino que se organizan como estructuras que permiten localizar el valor en menos tiempo. Por otro lado, cuando se recorre el índice se hace sobre registros pequeños, en comparación con los registros más grandes que contiene el fichero de datos; por lo tanto, el número de accesos a disco es menor.

No obstante, todo tiene un precio. Si se declara un índice, ese índice se debe mantener actualizado cada vez que la tabla sufra cualquier modificación. Si se definen muchos índices, el rendimiento de las actualizaciones se puede reducir significativamente. Por otra parte, si hay alternativas, siempre es mejor definir índices para los campos de menor tamaño, ya que cuanto más pequeño sea el campo clave, más pequeño será el índice y se necesitarán menos operaciones de lectura del índice.

Los índices se pueden definir como únicos o no (es decir, con duplicados o sin ellos). Los índices únicos indican que se aplican sobre

campos en los cuales no debe haber elementos repetidos. Todas las claves primarias llevan asociado un índice de forma predeterminada. También se puede indicar que acepten valores nulos o no. Si se aceptan, el índice permitirá esos valores nulos, pero los registros que los contengan no estarán apuntados por el índice.

3.5. Restricciones de integridad

En este tema se trata uno de los aspectos más importantes para añadir consistencia a los diseños de bases de datos: son las restricciones de integridad que ayudan a mantener la consistencia semántica de los datos. Además de las restricciones de integridad definidas por las claves, las restricciones de cardinalidad y las de participación total estudiadas anteriormente, se tratan las restricciones de los dominios, la integridad referencial, las dependencias funcionales y las dependencias multivaloradas.

Las restricciones de integridad proporcionan un medio de asegurar que las modificaciones hechas a la base de datos por los usuarios autorizados no provoquen la pérdida de la consistencia de los datos. Protegen a la base de datos contra los daños accidentales (no contra daños intencionados, de lo cual se ocupa la seguridad de las bases de datos). Los tipos de restricciones de integridad en una base de datos se pueden resumir como sigue:

- Claves.
- Cardinalidad de la relación.
- Restricciones de los dominios.
- Integridad referencial.
- Participación total.
- Dependencias funcionales.
- Otras restricciones.

En este último caso se recogen las restricciones que no se pueden catalogar en los casos anteriores. Por ejemplo, si se tiene una tabla en la que se describan un compuestos químicos y sus componentes, sería deseable imponer que la suma de los componentes, expresados en porcentajes, fuese el 100 por cien.

En las bases de datos tenemos los siguientes mecanismos para implementar las restricciones de integridad:

- Declaración de claves.
- Declaración de integridad referencial.
- Declaración de tipo de cardinalidad.
- Disparadores (Triggers).

3.5.1. Restricciones de los dominios

Las restricciones de los dominios son la forma más simple de restricción de integridad. Se especifica para cada atributo un dominio de

valores posibles. Una definición adecuada de las restricciones de los dominios no sólo permite verificar los valores introducidos en la base de datos sino también examinar las consultas para asegurarse de que tengan sentido las comparaciones que hagan. Por ejemplo, normalmente no se considerará que la consulta "Hallar todos los clientes que tengan el nombre de una

"sucursal" tenga sentido. Por tanto, nombre-cliente y nombre-sucursal deben tener dominios diferentes.

La cláusula **check** de SQL:1999 permite restringir los dominios de maneras poderosas que no permiten la mayor parte de los sistemas de tipos de los lenguajes de programación. La cláusula check permite especificar un predicado que debe satisfacer cualquier valor asignado a una variable cuyo tipo sea el dominio. Por ejemplo:

3.5.1. Restricciones de los dominios

Las restricciones de los dominios son la forma más simple de restricción de integridad. Se especifica para cada atributo un dominio de valores posibles. Una definición adecuada de las restricciones de los dominios no sólo permite verificar los valores introducidos en la base de datos sino también examinar las consultas para asegurarse de que tengan sentido las comparaciones que hagan. Por ejemplo, normalmente no se considerará que la consulta "Hallar todos los clientes que tengan el nombre de una sucursal" tenga sentido. Por tanto, nombre-cliente y nombre-sucursal deben tener dominios diferentes.

La cláusula **check** de SQL:1999 permite restringir los dominios de maneras poderosas que no permiten la mayor parte de los sistemas de tipos de los lenguajes de programación. La cláusula **check** permite especificar un predicado que debe satisfacer cualquier valor asignado a una variable cuyo tipo sea el dominio. Por ejemplo:

```
create domain sueldo-por-hora numeric(4,2)
constraint comprobacion-valor-sueldo check(value >= 6.00)
```

• **Restricciones de existencia**

Dentro de las restricciones de los dominios, un tipo especial de restricción que se puede aplicar a cualquier dominio es la restricción de existencia. Esta restricción evita la aparición de valores nulos en las columnas. Se especifica indicando en la creación de la tabla cuáles son los atributos que no pueden contener valores nulos. De manera predeterminada, los atributos que formen parte de la clave primaria tienen esta restricción impuesta. Para

Restricciones de existencia

Dentro de las restricciones de los dominios, un tipo especial de restricción que se puede aplicar a cualquier dominio es la restricción de existencia. Esta restricción evita la aparición de valores nulos en las columnas. Se especifica indicando en la creación de la tabla cuáles son los atributos que no pueden contener valores nulos. De manera predeterminada, los atributos que formen parte de la clave primaria tienen esta restricción impuesta. Para declarar esta restricción en la definición de la tabla se usaría NOT NULL después del nombre del atributo y su dominio.

Restricciones de unicidad

Otro tipo especial de restricción que se puede aplicar a cualquier dominio es la restricción de unicidad. Esta restricción evita la aparición de valores duplicados en las columnas (de forma parecida a lo que hace la clave primaria). Por ejemplo:

Sólo se admite una sucursal en cada ciudad.

The screenshot shows a web browser window with the URL <https://acceso.ispc.edu.ar/mod/book/view.php?id=13937&chapterid=1728>. The page content discusses database constraints, specifically uniqueness restrictions. It includes a code snippet for creating a table named 'Sucursales' with columns 'nombre-sucursal' and 'ciudad-sucursal'. The 'ciudad-sucursal' column is defined as NOT NULL and has a PRIMARY KEY constraint, while the entire column is also defined as UNIQUE. A note at the bottom states: "Sólo se admite una sucursal en cada ciudad." Below this, a section titled "3.5.2. Restricciones de integridad referencial" is shown, explaining how referential integrity ensures values from one relation appear in another. It includes a formula: Relación1.(Atributo1-1,...,Atributo1-N) ⊑ Relación2.(Atributo2-1,...,Atributo2-N). The browser interface includes a sidebar with various icons, a toolbar at the top, and a taskbar at the bottom showing system status and date.

3.5.2. Restricciones de integridad referencial

La integridad referencial permite asegurar que un valor que aparece en una relación para un conjunto de atributos determinado aparezca también en otra relación para ese mismo conjunto de atributos.

Este tipo de restricciones se denota simplificadamente:

$$\begin{aligned} \text{Relación1.(Atributo1-1,...,Atributo1-N)} &\subseteq \\ \text{Relación2.(Atributo2-1,...,Atributo2-N)} \end{aligned}$$

El conjunto de atributos {Atributo1-1,...,Atributo1-N} se denomina clave externa, mientras que el conjunto de atributos {Atributo2-1,...,Atributo2-N} es la clave primaria de Relación2.

Ya se ha visto cómo aparece este tipo de restricción de integridad en la traducción del esquema E-R al relacional. El sistema, por su parte, puede asegurar la imposición de estas restricciones de integridad, evitando la aparición de valores que las violen.

En concreto, la modificación de la base de datos puede ocasionar violaciones de la integridad referencial. Se distinguen tres casos:

- Al insertar una tupla es necesario comprobar que haya otra con los valores de la clave externa igual a los de sus atributos clave.
- Al borrar una tupla de R el sistema debe calcular el conjunto de tuplas de las otras relaciones que hacen referencia a R. Si este conjunto no es el conjunto vacío, o bien se rechaza la orden borrar como error, o bien se deben borrar las todas las tuplas que hacen referencia a

R. La última solución puede llevar a borrados en cascada, dado que las tuplas pueden hacer referencia a tuplas que hagan referencia a R, etcétera.

- Actualizar. Hay que considerar dos casos: las actualizaciones de la relación que realiza la referencia (r2) y las actualizaciones de la relación a la que se hace referencia (r1).

**Si se actualiza la tupla t2 de la relación r2 y esta actualización modifica valores de la clave externa α , se realiza una comprobación parecida al del caso de la inserción. El sistema debe asegurar que la nueva tupla encuentra sus valores de la clave externa en los de la clave primaria de r1.

**Si se actualiza la tupla t1 de la relación r1 y esta actualización modifica valores de la clave primaria, se realiza una comprobación parecida al del caso del borrado. Si quedan tuplas colgantes (sin correspondencia de clave externa con clave primaria), la actualización se rechaza como error o se ejecuta en cascada de manera parecida al borrado.

3.5.3. Dependencias funcionales

Una dependencia funcional (DF) es una propiedad semántica de un esquema de relación que impone el diseñador. Determina el valor de un conjunto de atributos a partir del valor de otro conjunto de atributos. Por ejemplo, en la siguiente relación se combinan los datos de los empleados, como su código de identificación y nombre, y de los centros a los que están adscritos, como la dirección y el teléfono.

The screenshot shows a web page from 'Guía de estudio INTRODUCCIÓN' at 'https://acceso.ispc.edu.ar/mod/book/view.php?id=13937&chapterid=1720'. The page contains the following text and a table:

realiza una comprobación parecida al del caso del borrado: si quedan tuplas colgantes (sin correspondencia de clave externa con clave primaria), la actualización se rechaza como error o se ejecuta en cascada de manera parecida al borrado.

3.5.3. Dependencias funcionales

Una dependencia funcional (DF) es una propiedad semántica de un esquema de relación que impone el diseñador. Determina el valor de un conjunto de atributos a partir del valor de otro conjunto de atributos. Por ejemplo, en la siguiente relación se combinan los datos de los empleados, como su código de identificación y nombre, y de los centros a los que están adscritos, como la dirección y el teléfono.

		Empleados_Centros					
Id_empleado	NombreE	DirecciónE	Puesto	Salario	Centro	DirecciónC	TeléfonoC
123A	Ana Almansa	c. Argentinas	Profesor	30.000	Informática	c/ Complutense	123
012D	David Díaz	c/ Daroca	Ayudante	10.000	Informática	c/ Complutense	123
789C	Carlos Crespo	c/ Cruz	Catedrático	30.000	Empresariales	c/ Coruña	789

En este ejemplo se muestra gráficamente que el valor del conjunto de campos DirecciónC y TeléfonoC depende del valor del campo Centro. En concreto, a un centro en particular le corresponden únicamente una dirección y un teléfono. Es decir, cada vez que aparezca una fila con el valor Informática para Centro, siempre le corresponderá los mismos valores para los campos DirecciónC y TeléfonoC.

Se dice entonces que tanto DirecciónC como TeléfonoC son dependientes funcionalmente de Centro. Por cada fila con un mismo valor de Centro se repiten los valores DirecciónC y TeléfonoC, lo que implica una redundancia de valores no deseable que se estudiará en el tema dedicado a la normalización.

Las dependencias funcionales se denotan de la siguiente forma:

En este ejemplo se muestra gráficamente que el valor del conjunto de campos DirecciónC y TeléfonoC depende del valor del campo Centro. En concreto, a un centro en particular le corresponden únicamente una dirección y un teléfono. Es decir, cada vez que aparezca una fila con el valor Informática para Centro, siempre le corresponderá los mismos valores para los campos DirecciónC y TeléfonoC.

Se dice entonces que tanto DirecciónC como TeléfonoC son dependientes funcionalmente de Centro. Por cada fila con un mismo valor de Centro se repiten los valores DirecciónC y TeléfonoC, lo que implica una redundancia de valores no deseable que se estudiará en el tema dedicado a la normalización.

Las dependencias funcionales se denotan de la siguiente forma:

Conjunto de atributos que determinan → Conjunto de atributos determinados

En el ejemplo anterior: {Centro} → { DirecciónC, TeléfonoC }

3.5.4. Disparadores

Un disparador es un mecanismo que se puede usar para implementar restricciones de integridad no soportadas directamente por el SGBD. Un disparador es una orden que el sistema ejecuta de manera automática como efecto secundario de la modificación de la base de datos. Los disparadores son mecanismos útiles para implementar restricciones de integridad, alertar a los usuarios o para realizar de manera automática ciertas tareas cuando se cumplen determinadas condiciones.

Un ejemplo que se aplica a la imposición de restricciones de integridad sería la implementación de la detección de la violación de las

dependencias funcionales.

Otro ejemplo sería enviar indicar cuándo se ha alcanzado un stock mínimo de un producto y que, por tanto, se debe reponer.

Un último ejemplo sería una aplicación bancaria en la que, en lugar de permitir saldos de cuenta negativos, un banco puede tratar los

descubiertos dejando a cero el saldo de las cuentas y creando un préstamo por el importe del descubierto. Este préstamo recibe un número de préstamo idéntico al número de cuenta que ha tenido el descubierto.

3.6. Normalización

La traducción del esquema conceptual al lógico no es única. Es necesario contar con una medida de la calidad de la agrupación de los

atributos en relaciones. Como herramienta principal se usan las dependencias funcionales para agrupar los atributos en esquemas de

relación, que se dice que se encuentran en una determinada forma normal. La forma normal de un esquema de relación determina su grado de calidad con respecto a reducir dos efectos no deseados: la redundancia de datos y las anomalías que produce esta redundancia. Es importante determinar en qué forma normal se encuentra un esquema de relación y el procedimiento, conocido como normalización, para descomponerlo en otros esquemas de relación que se encuentren en formas normales más exigentes.

3.6.1. Redundancia de datos

Un objetivo del diseño de bases de datos relacionales es agrupar atributos en relaciones de forma que se reduzca la redundancia de datos y así el espacio de almacenamiento necesario. Por ejemplo, los siguientes dos esquemas

Empleados(Id_empleado, NombreP, DirecciónP, Puesto, Salario, Centro)

Centros(NombreC, DirecciónC, Teléfono)

contienen la misma información que el siguiente:

Empleados_Centros(Id_empleado, NombreP, DirecciónP, Puesto, Salario, NombreC, DirecciónC, Teléfono)

Empleados					
Id_empleado	NombreE	DirecciónE	Puesto	Salario	Centro
123A	Ana Almansa	c/ Argentales	Profesor	20.000	Informática
456B	Bernardo Botín	c/ Barcelona	Administrativo	15.000	Matemáticas
789C	Carlos Crespo	c/ Cruz	Catedrático	30.000	CC. Empresariales
012D	David Diaz	c/ Daroca	Ayudante	10.000	Informática

Centros		
NombreC	DirecciónC	Teléfono
Informática	Complutense	123
Matemáticas	Complutense	456
CC. Empresariales	Somosaguas	789

Empleados_Centros							
Id_empleado	NombreE	DirecciónE	Puesto	Salario	Centro	DirecciónC	Teléfono
123A	Ana Almansa	c/ Argentales	Profesor	20.000	Informática	Complutense	123
456B	Bernardo Botín	c/ Barcelona	Administrativo	15.000	Matemáticas	Complutense	456
789C	Carlos Crespo	c/ Cruz	Catedrático	30.000	CC. Empresariales	Somosaguas	789
012D	David Diaz	c/ Daroca	Ayudante	10.000	Informática	Complutense	123

La relación Empleados_Centros presenta redundancia de datos porque se repite para cada empleado la información asociada al centro.

La relación **Empleados_Centros** presenta redundancia de datos porque se repite para cada empleado la información asociada al centro. Las relaciones con datos redundantes presentan diferentes anomalías de actualización: son las anomalías de inserción, borrado y modificación.

3.6.2. Anomalías de actualización

- Anomalías de inserción. Se produce en dos casos. En primer lugar, cuando se inserta una nueva fila sin respetar las dependencias funcionales. En el ejemplo anterior puede ocurrir si se añade una fila de un empleado adscrito a Informática y con un teléfono distinto de 123. En segundo lugar, la imposibilidad de añadir nuevos datos para el consecuente de la dependencia funcional sin que exista un antecedente para ella. En el ejemplo anterior no se puede dar de alta un centro a menos que exista un empleado destinado en él. Sería necesario dejar valores nulos en la clave (Id_empleado).

- Anomalías de modificación. Se produce cuando se modifican las columnas con datos redundantes de sólo un subconjunto de las filas con el mismo dato. En el ejemplo puede ocurrir cuando se modifica el teléfono de Informática sólo en la primera fila.

- Anomalías de eliminación. Se produce cuando se eliminan todas las filas en las que aparecen los datos redundantes por lo que se pierde los datos de la dependencia funcional. Si

se elimina la segunda fila porque el empleado se da de baja, se pierden también los datos del centro.

3.6.3. Formas normales y normalización

La forma normal de una relación se refiere a la mejor forma normal que satisface un esquema de relación indicando así el grado hasta el que se ha normalizado. La indicación del grado de calidad de un esquema de relación se refiere en general en el contexto global del esquema de la base de datos relacional, es decir, en el conjunto de todos los esquemas de relación de la base de datos. Dos propiedades que se deben cumplir para poder asegurarlo son:

- La propiedad de preservación de dependencias, que asegura que las dependencias funcionales originales se mantienen en algún esquema de relación después de la descomposición.
- La propiedad de la posibilidad de reproducir la información de la tabla antes de su descomposición a partir de las tablas resultado de ella.

Las formas normales más habituales, por orden ascendente de exigencia de las propiedades deseadas, son:

Primera (1FN)

Segunda (2FN)

Tercera (3FN)

Boyce/Codd (FNBC)

Hay otras formas normales más exigentes y que se refieren a otras propiedades deseables. Sin embargo, la utilidad práctica de estas formas normales es cuestionable cuando las restricciones en que se basan son difíciles de entender o identificar por los diseñadores y usuarios. Así, en la práctica se usa hasta la forma normal de Boyce/Codd, aunque en general, los diseños prácticos exigen al menos 3FN.

El proceso de asegurar una forma normal para un esquema se denomina normalización.

A continuación se estudiarán las formas normales mencionadas y para cada una de ellas se indicará el procedimiento que asegura que un esquema que no esté en una forma normal determinada se convierta en un conjunto de esquemas que asegure esa forma normal.

Normalización

Proceso de simplificación de los datos

- Tener almacenado con el menor espacio posible.
- Eliminar datos repetidos.
- Eliminar errores lógicos.
- Datos ordenados

Primer Forma Normal

Segunda Forma Normal

Tercer Forma Normal

Forma Normal Boyce Codd

Cuarta Forma Normal

Quinta Forma Normal

La simplificación debe darse sin que haya pérdida de información.

Código Compilado

Primer Forma Normal

Identificar si hay un grupo de repetición sobre el mismo registro.

Matrícula	Nombre	Dirección	Teléfono	Materia	Num Materia	Carrera
1	Sergio	Puebla 22	56565656	Base de datos	123	Sistemas
1	Sergio	Puebla 22	56565656	Programación web	234	Sistemas
1	Sergio	Puebla 22	56565656	Programación visual	234*	Sistemas
2	Ana	Reforma 1	23232323	Base de datos	123	Sistemas

Código Compilado

*Programación visual 237

Primer Forma Normal

Matrícula	Nombre	Dirección	Teléfono	Carrera
1	Sergio	Puebla 22	56565656	Sistemas
2	Ana	Reforma 1	23232323	Sistemas

Matrícula	Materia	Num Materia
1	Base de datos	123
1	Programación web	234
1	Programación visual	234
2	Base de datos	123



*Programación visual 237

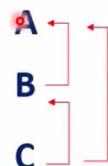
Segunda Forma Normal

- *La tabla debe estar en Primer Forma Normal*
- *Identificar las dependencias funcionales y transitivas.*

Dependencia Funcional



Dependencia Transitiva



Código Compilado



Segunda Forma Normal

Matrícula	Nombre	Dirección	Teléfono	Carrera
1	Sergio	Puebla 22	56565656	Sistemas
2	Ana	Reforma 1	23232323	Sistemas

Matrícula	Num Materia	Materia	Num Materia
1	123	Base de datos	123
1	234	Programación web	234
1	234	Programación visual	234
2	123	Base de datos	123

Código Compilado



*Programación visual 237

Tercer Forma Normal

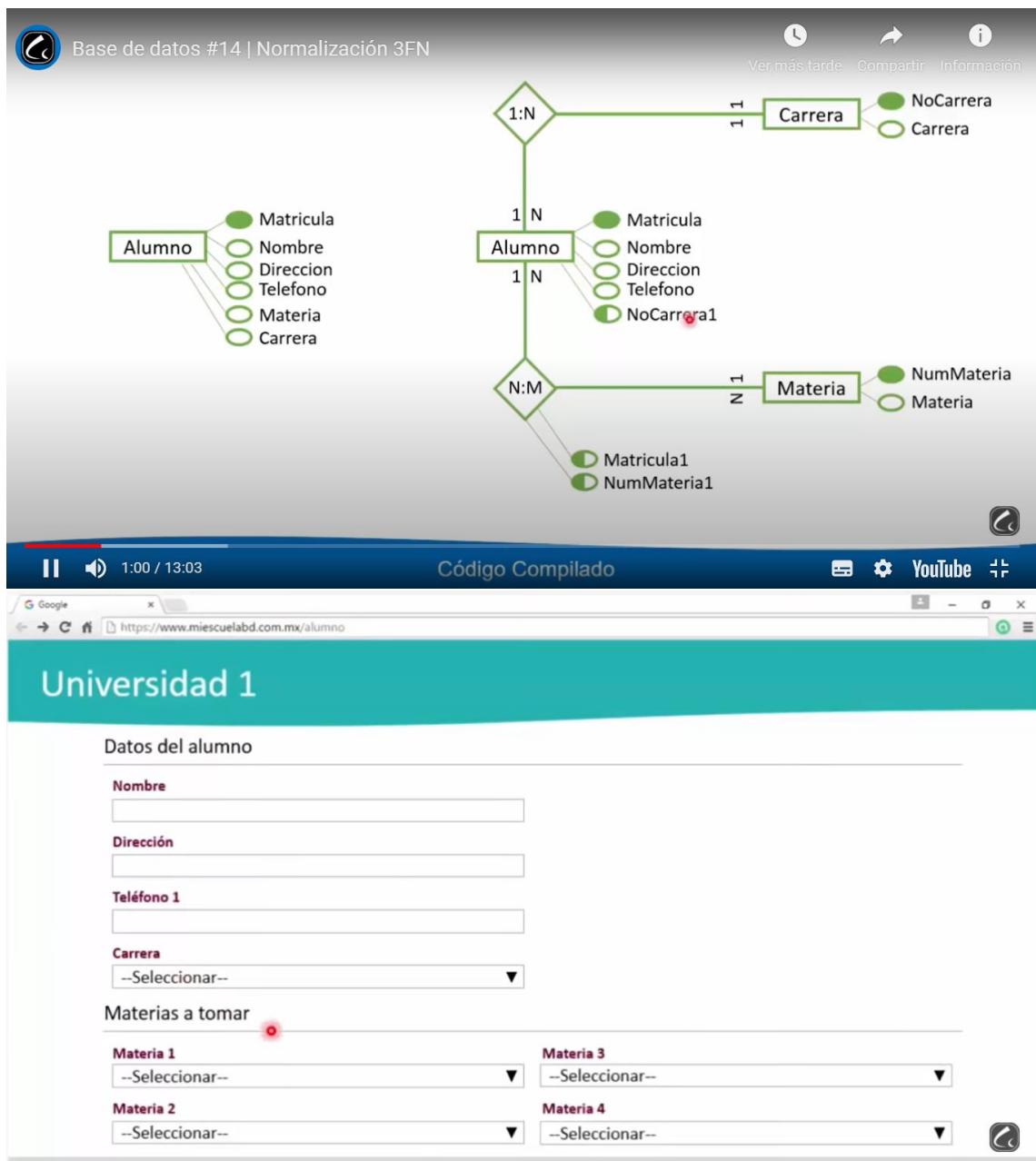
Matrícula	Nombre	Dirección	Teléfono	No Carrera	No Carrera	Carrera
1	Sergio	Puebla 22	56565656	1234	1234	Sistemas
2	Ana	Reforma 1	23232323	1234	6789	Mecatrónica

Matrícula	Num Materia	Materia	Num Materia
1	123	Base de datos	123
1	234	Programación web	234
1	237	Programación visual	237
2	123	Base de datos	123

Código Compilado



CódigoCompilado. (Marzo 2015). Base de datos #13 | Normalización (1FN, 2FN y 3FN) [Vídeo]. Youtube. <https://www.youtube.com/watch?v=bO18omSzeR4>



Utilizando listas de selección evitamos que el usuario escriba mal las opciones válidas

Cuando pedimos Nombre dos estudiantes distintos pueden ingresar Apellido 1º nombre y segundo nombre y otro 1º nombre y apellido directamente, perdemos información y control de esta información.

Lo mismo ocurre con la dirección.

Universidad 1

Datos del alumno

Nombre (s)	Apellido paterno	Apellido materno		
<input type="text"/>	<input type="text"/>	<input type="text"/>		
Delegación --Seleccionar--	Colonia --Seleccionar--	Calle --Seleccionar--	No Ext.	No Int.
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
CP <input type="text"/>				
Carrera --Seleccionar--		Teléfono 1 <input type="text"/>	Teléfono 2 <input type="text"/>	

Materias a tomar

Materia 1 --Seleccionar--	Materia 3 --Seleccionar--
Materia 2 --Seleccionar--	Materia 4 --Seleccionar--

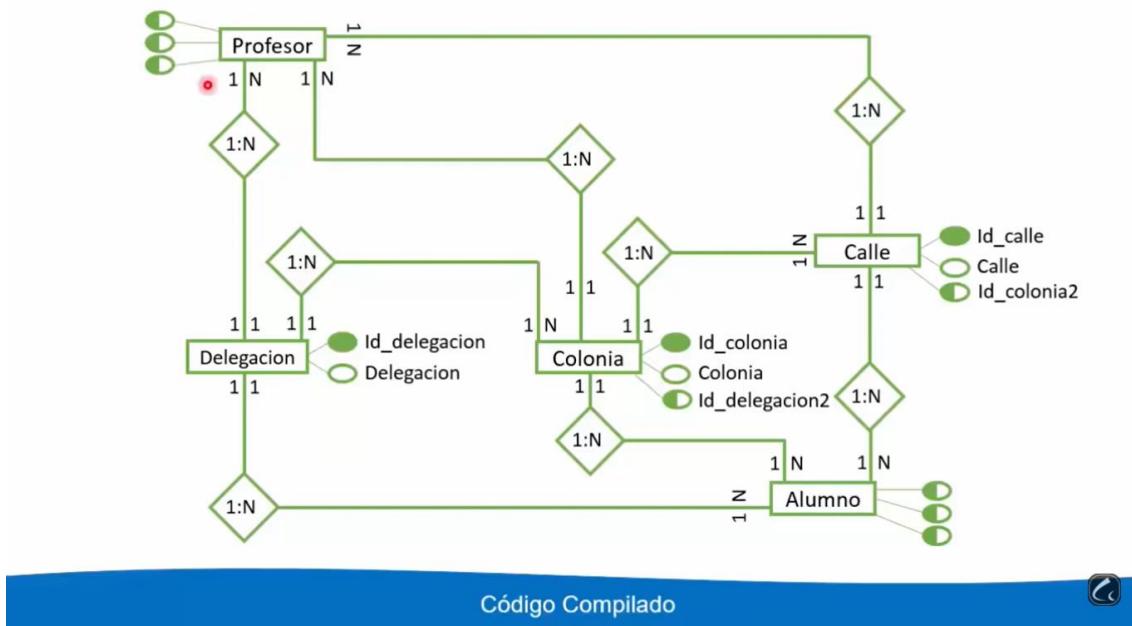
```

    graph TD
        Delegacion[Delegacion] -- "1:N" --> Alumno[Alumno]
        Delegacion -- "1:1" --> IdDelegacion[Id_delegacion]
        Delegacion -- "1:1" --> Delegacion[Delegacion]
        Colonia[Colonia] -- "1:N" --> Alumno
        Colonia -- "1:1" --> IdColonia[Id_colonia]
        Colonia -- "1:1" --> Colonia[Colonia]
        Colonia -- "1:1" --> IdDelegacion2[Id_delegacion2]
        Calle[Calle] -- "1:N" --> Alumno
        Calle -- "1:1" --> IdCalle[Id_calle]
        Calle -- "1:1" --> Calle[Calle]
        Calle -- "1:1" --> IdCalle2[Id_calle2]
        TipoTel[Tipo_tel] -- "N:M" --> Alumno
        TipoTel -- "1" --> IdTipotel[Id_tipotel]
        TipoTel -- "1" --> Tipotel[tipotel]
        TipoTel -- "1" --> IdTipotel1[Id_tipotel1]
        Materia[Materia] -- "N:M" --> Alumno
        Materia -- "1" --> Materia1[Materia]
        Materia -- "1" --> NumMateria1[NumMateria1]
        Matricula[Matricula] -- "1:N" --> Alumno
        Matricula -- "1" --> MatriculaAlu1[Matricula_alu1]
        Matricula -- "1" --> NumMateria11[NumMateria1]
        Alumno -- "1:N" --> Carrera[Carrera]
        Carrera -- "1" --> Carrera1[Carrera]
        Carrera -- "1" --> NoCarrera1[NoCarrera1]
        Alumno -- "1:N" --> Delegacion
        Alumno -- "1:N" --> Colonia
        Alumno -- "1:N" --> Calle
        Alumno -- "1:N" --> Matricula
        Alumno -- "1:N" --> Materia
        Alumno -- "1:N" --> Carrera
    
```

Delegaciones son la Localidades y Colonias son los Barrios.

Puede parecer excesivo tener registros de todas las localidades, barrios y calles, pero en realidad esto evita que el estudiante pueda ingresar información incoherente o equivocada, y no solo el estudiante pues si lo utilizamos luego para otras personas como por ejemplo los

profesores le estremos dando aun más utilidad.



Código Compilado



CodigoCompilado. (Marzo 2015). Base de datos #14 | Normalización 3FN [Vídeo]. Youtube.
https://www.youtube.com/watch?v=-LrUJR0G_6g

3.6.4. Primera forma normal

Actualmente se considera como parte de la definición formal de relación, porque establece que los dominios de los atributos sólo pueden ser atómicos, para evitar atributos multivalorados, compuestos y sus combinaciones. En definitiva evita las relaciones dentro de las relaciones.

Por ejemplo, si se asume que en la entidad Centros, un centro puede tener más de un teléfono, podríamos tener una representación como la siguiente.

The screenshot shows a web browser window with the following details:

- Title Bar:** Guía de estudio INTRODUCCIÓN
- Address Bar:** https://acceso.ispc.edu.ar/mod/book/view.php?id=13937&chapterid=1728
- Video Player:** A video player is displayed, showing the title "CodigoCompilado. (Marzo 2015). Base de datos #14 | Normalización 3FN [Vídeo]. Youtube. https://www.youtube.com/watch?v=-LrUJR0G_6g".
- Content Area:**
 - Section Header:** **3.6.4. Primera forma normal**
 - Text:** Actualmente se considera como parte de la definición formal de relación, porque establece que los dominios de los atributos sólo pueden ser atómicos, para evitar atributos multivalorados, compuestos y sus combinaciones. En definitiva evita las relaciones dentro de las relaciones.
 - Text:** Por ejemplo, si se asume que en la entidad Centros, un centro puede tener más de un teléfono, podríamos tener una representación como la siguiente.
 - Table:** A table titled "Centros" is shown with the following data:

NombreC	DireccionC	Teléfonos
Informatica	Completense	{123, 321, 213}
Matematicas	Completense	{456}
CC. Empresariales	Somosaguas	{789, 987}
 - Text:** Sin embargo, esto supondría el uso del atributo multivalorado Teléfonos. Hay tres posibilidades de representar la entidad para satisfacer la primera forma normal:
 - Eliminar el atributo Teléfonos y crear una nueva relación que asocie en cada fila un centro con un teléfono. La clave de la primera relación debe formar parte de la clave de la segunda relación. Presenta como inconveniente añadir una nueva relación al esquema de la base de datos y redundancia. Presenta anomalías cuando se borra un centro y no se borran los teléfonos asociados. La Integridad referencial asegura evitar las anomalías.
- Bottom Bar:** Shows system status including weather (10°C Soleado), date (18/5/2022), and time (20:17).

Sin embargo, esto supondría el uso del atributo multivalorado Teléfonos. Hay tres posibilidades de representar la entidad para satisfacer la primera forma normal:

- Eliminar el atributo Teléfonos y crear una nueva relación que asocie en cada fila un centro con un teléfono. La clave de la primera relación debe formar parte de la clave de la segunda relación. Presenta como inconveniente añadir una nueva relación al esquema de la base de datos y redundancia. Presenta anomalías cuando se borra un centro y no se borran los teléfonos asociados. La integridad referencial asegura evitar las anomalías.

Centros	
NombreC	DirecciónC
Informática	Complutense
Matemáticas	Complutense
CC. Empresariales	Somosaguas

Telefonos	
NombreC	Teléfono
Informática	123
Informática	321
Informática	213
Matemáticas	456
CC. Empresariales	789
CC. Empresariales	987

• Ampliar la clave de la relación de manera que incluya al atributo multivalorado. Presenta como inconveniente añadir redundancia que provoca anomalías.

- Ampliar la clave de la relación de manera que incluya al atributo multivalorado. Presenta como inconveniente añadir redundancia que provoca anomalías.

- Si se conoce la cardinalidad máxima del atributo multivalorado, se pueden crear tantas columnas como la cardinalidad máxima. Presenta como inconveniente el uso de valores Null.

Si el atributo multivalorado es compuesto, como es el caso de representar varias direcciones para un empleado:

```
Empleados(Id_empleado, NombreE, {Direcciones(Calle, Ciudad, CódigoPostal)})
```

Esta relación se puede descomponer en dos:

```
Empleados(Id_empleado, NombreE)
```

```
DireccionesE(Id_empleado, Calle, Ciudad, CódigoPostal)
```

Este procedimiento de desanidamiento se puede aplicar recursivamente a cualquier relación con atributos multivalorados. teniendo

en cuenta que es necesario propagar la clave de la relación original a la clave de la nueva relación, que contiene además la clave que identifica únicamente al atributo multivalorado.

3.6.5. Segunda forma normal

En el ejemplo a continuación se puede observar que existen anomalías de actualización causadas por las dependencias funcionales DF2 y DF3, ya que como sus antecedentes no son clave, puede haber varias filas con el mismo consecuente. Se usa una notación gráfica para la expresión de las dependencias funcionales. Así:

3.6.5. Segunda forma normal

En el ejemplo a continuación se puede observar que existen anomalías de actualización causadas por las dependencias funcionales DF2 y DF3, ya que como sus antecedentes no son clave, puede haber varias filas con el mismo consecuente. Se usa una notación gráfica para la expresión de las dependencias funcionales. Así:

$DF1 = \{Id_empleado, NúmeroP\} \rightarrow \{Horas\}$
 $DF2 = \{Id_empleado\} \rightarrow \{NombreE\}$
 $DF3 = \{NúmeroP\} \rightarrow \{NombreP\}$

Personal_Proyectos				
Id_empleado	NúmeroP	Horas	NombreE	NombreP
123A	P-1	16	Ana Almansa	Proyecto 1
012D	P-1	8	David Diaz	Proyecto 1
012D	P-2	4	David Diaz	Proyecto 2

Diagrama de dependencias funcionales:

```
graph TD; DF1[DF1] --> Horas[Horas]; DF1 --> NombreE[NombreE]; DF2[DF2] --> NombreE; DF3[DF3] --> NombreP[NombreP]
```

La segunda forma normal evita este tipo de anomalías.
Se dice que una relación está en segunda forma normal si cada atributo que no forme parte de la clave primaria depende funcionalmente de la clave primaria.

La segunda forma normal evita este tipo de anomalías.

Se dice que una relación está en segunda forma normal si cada atributo que no forme parte de la clave primaria depende funcionalmente de la clave primaria y completamente de cada clave.

Un esquema que no se encuentre en segunda forma normal puede traducirse en varios esquemas que sí lo estén. El procedimiento de normalización es crear tantas nuevas relaciones como dependencias funcionales no sean completas.

Así, el ejemplo anterior se traduce en:

La segunda forma normal evita este tipo de anomalías. Se dice que una relación está en segunda forma normal si cada atributo que no forme parte de la clave primaria depende funcional y completamente de cada clave.

Un esquema que no se encuentre en segunda forma normal puede traducirse en varios esquemas que sí lo estén. El procedimiento de normalización es crear tantas nuevas relaciones como dependencias funcionales no sean completas.

Así, el ejemplo anterior se traduce en:

PP1	PP2	PP3
Id_empleado NúmeroP Horas	Id_empleado NombreE	NúmeroP NombreP
DF1	DF2	DF3

Hay que observar que este procedimiento asegura que el resultado está, al menos, en segunda forma normal. En particular, y como se puede contrastar con la definición de otras formas normales, el resultado conseguido en este ejemplo se encuentra en FNBC, como se podrá comprobar más adelante.

3.6.6. Tercera forma normal

En el ejemplo a continuación se puede observar que existen anomalías de actualización causadas por la dependencia funcional DF3.

Hay que observar que este procedimiento asegura que el resultado está, al menos, en segunda forma normal. En particular, y como se puede contrastar con la definición de otras formas normales, el resultado conseguido en este ejemplo se encuentra en FNBC, como se podrá comprobar más adelante.

3.6.6. Tercera forma normal

En el ejemplo a continuación se puede observar que existen anomalías de actualización causadas por la dependencia funcional DF2. Sin embargo, este esquema está en segunda forma normal porque los dos últimos atributos, que son los que causan las anomalías, dependen completa (y transitivamente) del único atributo que forma la clave primaria.

En el ejemplo anterior, DF3 es una dependencia funcional transitiva:

Empleados_Departamentos					
Id_empleado	NombreE	DirecciónE	CódigoD	NombreD	DirectorD
123A	Ana Almansa	c/ Argentinas	DS	Sistemas	999Z
012D	David Diaz	c/ Daroca	DS	Sistemas	999Z

DF1 | DF2 | DF3

Un esquema está en tercera forma normal si satisface la segunda forma normal y ninguno de los atributos que no forman parte de una clave candidata depende transitivamente de la clave primaria.

El procedimiento para normalizar esta relación consiste en descomponerla en los atributos definidos por la dependencia funcional responsable de la transitividad. En el ejemplo anterior, el resultado de la descomposición es:

ED1	ED2
Id_empleado NombreE DirecciónE CódigoD	CódigoD NombreD DirectorD
DF1	DF2

El siguiente es un algoritmo que se puede aplicar para calcular estas descomposiciones, donde R es el esquema

La tercera forma normal se basa en el concepto de dependencia funcional transitiva.

Una dependencia funcional $X \rightarrow Y$ es una dependencia funcional transitiva si existe un conjunto de atributos Z que ni forman clave candidata ni son subconjunto de ninguna clave y además se cumple $X \rightarrow Z$ y $Z \rightarrow Y$.

En el ejemplo anterior, DF3 es una dependencia funcional transitiva:

Un esquema está en tercera forma normal si satisface la segunda forma normal y ninguno de los atributos que no forman parte de una clave candidata depende transitivamente de la clave primaria.

El procedimiento para normalizar esta relación consiste en descomponerla en los atributos definidos por la dependencia funcional

responsable de la transitividad. En el ejemplo anterior, el resultado de la descomposición es:

El siguiente es un algoritmo que se puede aplicar para calcular estas descomposiciones, donde R es el esquema original a descomponer y F el conjunto de dependencias funcionales que se cumple en R :

D= {R}

- 1.** Encontrar un recubrimiento mínimo T de F
- 2.** for each $X \rightarrow Y \in T$, crear en D un esquema R_i con $\{X \cup \{A_1\} \cup \dots \cup \{A_k\}\}$
si $R_i \notin D$, dadas las D.F. $X \rightarrow \{A_1\}, \dots, X \rightarrow \{A_k\}$
- 3.** Si ninguno de los esquemas contiene una clave de R , se crea uno nuevo.

Donde el recubrimiento mínimo es el resultado en primer lugar de eliminar todos los atributos de los antecedentes de las dependencias funcionales que sea posible y, en segundo lugar, todas las dependencias funcionales que sea posible.

3.6.7. Forma normal de Boyce-Codd

La forma normal de Boyce-Codd (FNBC) se propuso como una forma más simple que la tercera, pero en realidad es más estricta porque cada relación en FNBC está en 3FN pero lo contrario no se cumple.

Por ejemplo, considérese la siguiente relación, que está en 3FN pero no en FNBC. La FNBC evita redundancias que la 3FN no puede. En este ejemplo se almacena información de los investigadores participantes en proyectos, que pueden ser

codirigidos, pero los investigadores principales no pueden dirigir más de uno.

The screenshot shows a web page with a sidebar containing icons for file operations, a search bar, and a user profile. The main content area displays a relational database schema titled 'Proyectos' with three tables: 'Investigador', 'Proyecto', and 'IPrincipal'. Below the schema is a dependency diagram showing two dependencies: DF1 (Investigador → IPrincipal) and DF2 (IPrincipal → Proyecto). A note below the schema states: "En este ejemplo, que cumple la 3NF, hay una anomalía que se debería poder evitar, porque si no se vigila la dependencia funcional DF2 se podría añadir una tupla de manera que una persona fuese investigadora principal de dos proyectos." A legend indicates that arrows point from the left to the right of the dependency lines.

En este ejemplo, que cumple la 3NF, hay una anomalía que se debería poder evitar, porque si no se vigila la dependencia funcional DF2 se podría añadir una tupla de manera que una persona fuese investigadora principal de dos proyectos.

La descomposición del esquema no es inmediata, hay tres posibilidades:

P1a	P1b
Investigador IPrincipal	Investigador Proyecto
↑	
P2a	P2b
Proyecto IPrincipal	Proyecto Investigador
↑	
P3a	P3b
IPrincipal Proyecto	IPrincipal Investigador
↑	

En este ejemplo, que cumple la 3NF, hay una anomalía que se debería poder evitar, porque si no se vigila la dependencia funcional DF2 se podría añadir una tupla de manera que una persona fuese investigadora principal de dos proyectos.

La descomposición del esquema no es inmediata, hay tres posibilidades:

Todas estas descomposiciones pierden la dependencia funcional DF1 porque las dependencias funcionales se refieren al contexto local de un esquema, no hacen referencia a esquemas externos. Aún peor, las dos primeras generan tuplas incorrectas cuando se trata de recuperar la información de la tabla original. Por ejemplo, en la primera descomposición se pierde la información de los investigadores principales de cada proyecto:

The screenshot shows a web page with a sidebar containing icons for file operations, a search bar, and a user profile. The main content area displays a relational database schema with two tables: 'P1a' and 'P1b'. Below the schema is a dependency diagram showing two dependencies: DF1 (Investigador → IPrincipal) and DF2 (IPrincipal → Proyecto). A note below the schema states: "Todas estas descomposiciones pierden la dependencia funcional DF1 porque las dependencias funcionales se refieren al contexto local de un esquema, no hacen referencia a esquemas externos. Aún peor, las dos primeras generan tuplas incorrectas cuando se trata de recuperar la información de la tabla original. Por ejemplo, en la primera descomposición se pierde la información de los investigadores principales de cada proyecto:". A legend indicates that arrows point from the left to the right of the dependency lines.

Todas estas descomposiciones pierden la dependencia funcional DF1 porque las dependencias funcionales se refieren al contexto local de un esquema, no hacen referencia a esquemas externos. Aún peor, las dos primeras generan tuplas incorrectas cuando se trata de recuperar la información de la tabla original. Por ejemplo, en la primera descomposición se pierde la información de los investigadores principales de cada proyecto:

P1a	P1b
Investigador IPrincipal	Investigador Proyecto
111A 123A	111A Proyecto 1
222B 012D	222B Proyecto 1
333C 123A	333C Proyecto 1
444D 123A	444D Proyecto 1
444D 789C	444D Proyecto 2
↑	
P1a ? P1b	
Investigador Proyecto IPrincipal	
111A Proyecto 1 123A	
222B Proyecto 1 012D	
333C Proyecto 1 123A	
444D Proyecto 2 123A	
444D Proyecto 2 789C	
444D Proyecto 1 123A	
444D Proyecto 1 789C	

Por tanto, ninguna de estas dos últimas descomposiciones es aceptable. En la práctica, la mayoría de los esquemas que están en 3NF lo están también en FNBC.

Por tanto, ninguna de estas dos últimas descomposiciones es aceptable.

En la práctica, la mayoría de los esquemas que están en 3NF lo están también en FNBC.

3.6.8. Desnormalización para el rendimiento

A veces los diseñadores de bases de datos escogen un esquema que tiene información redundante; es decir, que no está normalizada. Utilizan la redundancia para mejorar el rendimiento para aplicaciones concretas. La penalización sufrida por emplear un esquema normalizado es el trabajo extra (en términos de tiempo de codificación y de tiempo de ejecución) de mantener consistentes los datos redundantes.

Por ejemplo, supóngase que hay que mostrar el nombre del titular de una cuenta junto con el número y el saldo de su cuenta cada vez que se tiene acceso a la cuenta. En el esquema normalizado esto exige una reunión de cuenta con impositor.

Una alternativa para calcular la reunión sobre la marcha es almacenar una relación que contenga todos los atributos de cuenta y de

impositor. Esto hace más rápida la visualización de la información de la cuenta. No obstante, la información del saldo de la cuenta se repite para cada persona titular de la cuenta, y la aplicación debe actualizar todas las copias, siempre que se actualice el saldo de la cuenta. El proceso de tomar un esquema normalizado y hacerlo no normalizado se denomina desnormalización, y los diseñadores lo utilizan para ajustar el rendimiento de los sistemas para dar soporte a las operaciones críticas en el tiempo.

Una alternativa mejor, soportada hoy en día por muchos sistemas de bases de datos, es emplear el esquema normalizado y, de manera adicional, almacenar la reunión en forma de vista materializada. (Una vista materializada es una vista de la base de datos cuyo resultado se almacena en la base de datos y se actualiza cuando se actualizan las relaciones utilizadas en la vista). Al igual que la desnormalización, el empleo de las vistas materializadas supone sobrecargas de espacio y de tiempo; sin embargo, presenta la ventaja de que conservar la vista actualizada es labor del sistema de bases de datos, no del programador de la aplicación.

3.7. Normativa de denominación

La normativa de denominación es una colección de reglas que permite asignar nombres a identificadores y objetos. El objetivo es que los nombres que se elijan indiquen de forma lo más clara posible el significado del elemento al que se refiere el nombre. Cada empresa suele usar una normativa diferente y más o menos complicada. A veces se tienen manuales de treinta páginas que la describen. En este documento se aconsejarán algunas reglas que se pueden usar para establecer una normativa de denominación.

3.7.1. Identificadores

Los identificadores (o nombres que se usan para designar los elementos de una base de datos) se construyen generalmente con letras y números. En muchos SGBD no se distinguen mayúsculas de minúsculas, pero su uso nos puede ayudar a hacer más legibles los identificadores. Cuando los identificadores tienen más de una palabra hay dos alternativas que se usan habitualmente:

- Separar cada palabra por un carácter de subrayado, como en Nombre_del_paciente.

- Separar cada palabra poniendo la primera letra de cada una en mayúsculas, como en NombreDelPaciente.

Otra alternativa final, menos usada, es usar espacios en blanco para la separación, como en Nombre del paciente. Algunos SGBD tienen problemas con estos espacios en blanco y por eso no se usan habitualmente.

Incluso tienen problemas con algunos caracteres como las vocales acentuadas y las eñes, por lo que en general se evitan. En Access, sin embargo, no se tienen estos problemas y hemos seguido esta normativa de denominación de los identificadores.

3.7.2. Tablas

Las tablas representan entidades y sus nombres deberían describir las entidades que representan. Por ejemplo, Pacientes sería un identificador que describe a una tabla que contiene información sobre las entidades Pacientes. Además se escribe en plural porque el tipo de entidad contiene un conjunto de entidades (la tabla contiene varios pacientes en general).

Algunas tablas, sin embargo, no presentan entidades. Pueden representar relaciones entre entidades. Por ejemplo, la relación entre

Personas y Teléfonos se puede denominar Tiene teléfono. Cuando no se pueda encontrar un identificador adecuado para una relación siempre se puede escribir un identificador con los dos nombres de las tablas, como PersonasTeléfonos.

Reglas básicas de denominación de tablas:

- Seleccionar nombres de tablas basados en los nombres posibles para las entidades involucradas.
 - Usar sustantivos para los nombres de las tablas.
 - Asegurarse de que los nombres tienen un sentido intuitivo en la cultura de quienes utilizan la base de datos.
 - Expresar las relaciones capturadas por las tablas mediante la vinculación de los nombres de las entidades vinculadas por la tabla.

En las tablas se tiene que denominar a las columnas. Las columnas representan elementos discretos de información sobre la entidad nombrada por la tabla. Normalmente tampoco representan relaciones. Debido a estos hechos el nombre de una columna debería ser un sustantivo que nombre el elemento de información que representa. Debería ser un sustantivo que reflejara la forma que los usuarios hablan sobre el elemento de información y debería reflejar características sobresalientes sobre el elemento de información.

Por ejemplo, se usará Nombre como identificador del atributo nombre de pila de un paciente.

3.7.3. Restricciones

Las restricciones se pueden denominar de formas autointerpretativas.

Hay que utilizar una abreviatura de dos letras para identificar la naturaleza de la restricción:

- CP (o PK en inglés, primary key) para clave principal
- IR (o RI en inglés, referential integrity) para integridad referencial
- CO (o CK en inglés, check) para la de comprobación
- UN para la de unicidad.

Después hay que utilizar el nombre de la tabla a la cual se aplica la restricción como segundo elemento del nombre. Una restricción de clave principal para la tabla Médicos se debería nombrar CP_Médicos.

En el caso de claves externas, donde están involucradas dos tablas, hay que poner el nombre de la segunda tabla como tercer elemento para el nombre de la restricción. Una clave externa para las tablas Médicos y Especialidades tendría el nombre IR_Médicos_Especialidades. Cuando están involucradas dos tablas hay que hacer que el segundo elemento del nombre sea la tabla con la clave externa y que el tercer elemento sea la tabla donde reside la clave principal.

Finalmente, para las restricciones de comprobación, que tienen un papel funcional, hay que agregar un elemento final al nombre que describe su función. Una restricción de comprobación que garantice el formato de un número telefónico, por ejemplo, podría tener este nombre:

CO_Médicos_FormatoCódigo.

3.7.4. Controles

Cada tipo de control se debería denominar con una indicación del tipo de control, anteponiendo a un nombre descriptor un prefijo que indique el tipo, como se propone en la siguiente tabla.

The screenshot shows a web page titled "Guía de estudio INTRODUCCIÓN" from the URL <https://acceso.ispc.edu.ar/mod/book/view.php?id=13937&chapterid=1728>. The page contains a table with the following data:

Control	Prefijo	Ejemplo
Cuadro de texto	txt (acrónimo de text, texto en inglés)	txtEntrada
Etiqueta	lbl (acrónimo de label, etiqueta en inglés)	lblEtiqueta
Cuadro combinado	cmb (acrónimo de combo box, cuadro combinado en inglés)	cmbProvincias
Botón de comando	but (acrónimo de button, botón en inglés)	butCancelar
Grupo de opciones (marcos)	fra (acrónimo de frame, marco en inglés)	frsOperaciones
Botón de opción	opt (acrónimo de option, opción en inglés)	optCubo
Botón de selección	chk (acrónimo de check, verificación en inglés)	chkVerde
Lista	lst (acrónimo de list, lista en inglés)	lstPacientes

3.7.5. Variables

Cada variable se debería denominar con una indicación del tipo de la variable, anteponiendo a un nombre descriptor un prefijo que indique el tipo, como se propone en la siguiente tabla.

The screenshot shows a web page titled "Guía de estudio INTRODUCCIÓN" from the URL <https://acceso.ispc.edu.ar/mod/book/view.php?id=13937&chapterid=1728>. The page contains a table with the following data:

Tipo de variable	Prefijo	Ejemplo
Byte	byt	bytEdad
Boolean	boo	booCasado
Integer	int	intPersonas
Long	lng	lngLlamadas
Single	sgl	sglRaíz
Double	dbl	dblSuperficie
Currency	cur	eurEuros
Date	dat	datFecha
Object	obj	objObjeto
String	str	strNombre
Variant	var	varElemento

3.7.6. Objetos de la base de datos

Cada objeto de la base de datos se debería denominar con una indicación del tipo de objeto, anteponiendo a un nombre descriptor un prefijo que indique el tipo, como se propone en la siguiente tabla.

The screenshot shows a web browser window with the title "Guía de estudio INTRODUCCIÓN". The URL is <https://acceso.ispc.edu.ar/mod/book/view.php?id=13937&chapterid=1728>. The page content is titled "3.7.6. Objetos de la base de datos" and contains a table defining prefixes for database objects. The table has columns "Tipo de objeto", "Prefijo" (with a description in parentheses), and "Ejemplo".

Tipo de objeto	Prefijo (acrónimo de table, tabla en inglés)	Ejemplo
Tabla	tbl	tblPacientes
Formulario	frm	frmMedicos
Informe	rep	repPacientes
Consulta	qry	qryFastaTrimestral

The browser interface includes a sidebar with various icons, a navigation bar with back, forward, and search buttons, and a status bar at the bottom showing weather (10°C Soleado), system icons, and the date/time (18/5/2022, 20:55).