# FUNDAMENTOS SOBRE LA GESTIÓN DE BASE DE DATOS

*Ángel Pisco Gómez*
*Julio Johnny Regalado Jalca*
*Jimmy Gutiérrez García*
*Omar Quimis Sánchez*
*Kleber Marcillo Parrales*
*Javier Marcillo Merino*

Ingeniería y Tecnología

3ciencias


# FUNDAMENTOS SOBRE LA GESTIÓN DE BASE DE DATOS

*Ángel Pisco Gómez*
*Julio Johnny Regalado Jalca*
*Jimmy Gutiérrez García*
*Omar Quimis Sánchez*
*Kleber Marcillo Parrales*
*Javier Marcillo Merino*

**Editorial Área de Innovación y Desarrollo,S.L.**



ÁREA DE INNOVACIÓN Y DESARROLLO, S.L.
C/ Els Alzamora, 17 - 03802 - ALCOY (ALICANTE) info@3ciencias.com

Primera edición: **diciembre 2017**

ISBN: **978-84-947995-6-3**

DOI: **http://dx.doi.org/10.17993/IngyTec.2017.23**

# ÍNDICE GENERAL

# ÍNDICE DE TABLA



# ÍNDICE DE FIGURA

# PRÓLOGO

Actualmente las bases de datos han tomado gran importancia mundial, es totalmente indispensable para las organizaciones puesto que brinda mucha facilidad en el acceso a la información y una gran capacidad de almacenaje de datos. Es por este motivo que resulta muy grato dar a conocer este libro que se titula **"**FUNDAMENTOS SOBRE LA GESTIÓN DE BASE DE DATOS*",* el mismo que ha sido creado para orientar a cualquier persona: desde un estudiante en sus primeros pasos en el aprendizaje del mundo de la base de datos, hasta un profesional ávido de nuevos conocimientos. Con este libro se pretende que el lector pueda despejar toda clase de interrogantes acerca de bases de datos y le permita familiarizarse rápidamente con su contenido, no solo teórico sino también práctico mediante una serie de ejercicios. La estructura de este libro está sujeta a un sinnúmero de temas, los mismos que se encuentran organizados en 5 capítulos, cuyo contenido se presenta de manera clara y precisa. Cada capítulo es sumamente importante en el proceso de aprendizaje de las bases de datos. Los temas de este libro versan sobre las bases de datos relacionales y el modelado de datos, cada uno con definiciones claras y entendibles, así mismo reglas de transformación totalmente necesarias para crear un modelado relacional y además el modelo orientado a objetos. El objetivo del libro es lograr que cada lector enriquezca sus conocimientos con los temas plasmados en cada capítulo, y servir como ayuda didáctica a cientos de docentes en el área de bases de datos.

Este libro está dirigido a los estudiantes universitarios que cursan carreras relacionadas con las tecnologías de la información, los sistemas de información o con la informática. El libro está pensado como texto para un curso de un cuatrimestre sobre el diseño de bases de datos, también se dirige a los profesionales como: analistas de sistemas, programadores de aplicaciones y personas en general que están relacionadas con el uso de bases de datos o interesadas en aprender esta tecnología.

***LOS AUTORES***

# CAPÍTULO 1: ADMINISTRACIÓN DE BASE DE DATOS

## 1.1 Introducción a las Bases de Datos

En la medida en las empresas comenzaron a darse cuenta del valor de la información y del enorme potencial que los sistemas computacionales representaban para organizar y administrar estos recursos, se fue produciendo una demanda muy fuerte de sistemas de información y un reconocimiento siempre creciente de que la información como recurso que tiene valor necesita estar organizada y administrada. Aun cuando en las empresas se acostumbre a trabajar con activos tangibles, tales como el dinero, las instalaciones y el personal, cuyo valor puede evaluarse con cierta precisión, ha sido muy difícil de medir el valor de la información. Sin embargo, está claro que si los directivos tienen buena información, es más probable que puedan tornar decisiones pertinentes y certeras con un mayor impacto positivo en su negocio. Y viceversa, si su información es pobre, ellos deben trabajar con más incertidumbre y es menos probable que tomen decisiones convenientes. El desarrollo de los sistemas de bases de datos se convirtió en crucial para proporcionar información correcta y oportuna a los directivos.

Las bases de datos son muy utilizadas en la actualidad, las más utilizadas son las bases de datos relacionales, pero las primeras que existieron eran jerárquicas, estas evolucionaron a las reticulares hasta el advenimiento de las relacionales que son las que trataremos en esta asignatura. Existen bases de datos orientadas a objetos, espaciales, deductivas, los llamados almacenes de datos o datawarehouse, entre otras.

## 1.2 Surgimiento histórico de las Bases de Datos

Década de 1950 y principios de la década de 1960. La regla era el tratamiento de archivos secuenciales. Se desarrollaron las cintas magnéticas para el almacenamiento de datos. Las tareas de procesamiento de datos tales como las nóminas fueron automatizadas, con los datos almacenados en cintas. El procesamiento de datos consistía en leer datos de una o más cintas y escribir datos en una nueva cinta. Los datos también se podían introducir desde paquetes de tarjetas perforadas e impresos en impresoras. Todos los datos se almacenaban en archivos secuenciales, que exigían el tratamiento de archivos completos por los programas de aplicación.

Las cintas (y los paquetes de tarjetas perfora-das) sólo se podían leer secuencialmente, y los tamaños de datos eran mucho mayores que la memoria principal; así, los programas de procesamiento de datos tenían que procesar los datos según un determinado orden, leyendo y mezclando datos de cintas y paquetes de tarjetas perforadas.

Finales de la década de 1960 y la década de 1970.El amplio uso de los discos fijos a finales de la década de 1960 cambió en gran medida el escenario del procesamiento de datos, ya que los discos fijos permitieron el acceso directo a los datos. La ubicación de los datos en disco no era importante, ya que a cualquier posición del disco se podía acceder en sólo decenas de milisegundo. Los datos se liberaron de la tiranía de la secuencialidad. Con los discos pudieron

desarrollarse las bases de datos de red y jerárquicas, que permitieron que las estructuras de datos tales como listas y árboles pudieran almacenarse en disco.

En la medida en que los sistemas computacionales de procesamiento de datos se hicieron más importantes, los negocios comenzaron a reconocer que la información era un recurso corporativo de valor considerable. Estos percibieron más y más que los datos necesarios para contestar numerosas preguntas del negocio estaban disponibles en sus archivos de procesamiento de datos. Como consecuencia, comenzaron a presionar a los sistemas de información para la gestión en cuanto a la utilización de la potencia del computador para producir información a partir de los datos corporativos. Esto inició la demanda de los sistemas de base de datos, los que garantizarían más efectivamente el acceso a los datos y su manipulación.

Un artículo histórico de Codd en 1970 definió el modelo relacional y formas no procedimentales de consultar los datos en el modelo relacional, y nacieron las bases de datos relacionales. La simplicidad del modelo relacional y la posibilidad de ocultar completamente los detalles de implementación al programador fueron realmente atractivas.
Inmediatamente después, se desarrollaron los sistemas de base de datos en redes que soportaron interrelaciones entre registros de archivos diferentes mucho más complejas. Ambos modelos de base de datos, el jerárquico y el de red, requirieron el uso de punteros físicos predefinidos para enlazar los registros relacionados.

Década de 1980. Aunque académicamente interesante, el modelo relacional no se usó inicialmente en la práctica debido a sus inconvenientes por el rendimiento; las bases de datos relacionales no pudieron competir con el rendimiento de las bases de datos de red y jerárquicas existentes. Esta situación cambió con System R, un proyecto innovador en IBM Research que desarrolló técnicas para la construcción de un sistema de bases de datos relacionales eficiente.
Los primeros sistemas de bases de datos relacionales, como DB2 de IBM, Oracle, Ingres y Rdb de DEC, jugaron un importante papel en el desarrollo de técnicas para el procesamiento eficiente de consultas declarativas.

Las bases de datos relacionales fueron tan sencillas de usar que finalmente reemplazaron a las bases de datos jerárquicas y de red; los programadores que usaban estas bases de datos estaban forzados a tratar muchos detalles de implementación de bajo nivel y tenían que codificar sus consultas de forma procedimental.
Desde su escalada en el dominio en la década de 1980, el modelo relacional ha conseguido el reinado supremo entre todos los modelos de datos.

La década de 1980 también fue testigo de una gran investigación en las bases de datos paralelas y distribuidas, así como del trabajo inicial en las bases de datos orientadas a objetos.

Principios de la década de 1990. El lenguaje SQL se diseñó fundamentalmente para las aplicaciones de ayuda a la toma de decisiones, que son intensivas en consultas, mientras que el objetivo principal de las bases de datos en la década de 1980 fue las aplicaciones de procesamiento de transacciones, que son intensivas en actualizaciones. La ayuda a la toma de

decisiones y las consultas reemergieron como una importante área de aplicación para las bases de datos. Las herramientas para analizar grandes cantidades de datos experimentaron un gran crecimiento de uso.

**Finales de la década de 1990 hasta la actualidad**
En este periodo, el principal acontecimiento fue el crecimiento explosivo de World Wide Web. Las bases de datos se implantaron mucho más extensivamente que nunca antes. Los sistemas de bases de datos tienen ahora soporte para tasas de transacciones muy altas, así como muy alta fiabilidad y disponibilidad 24 horas, que significa que no hay tiempos de inactividad debidos a actividades de mantenimiento planificadas. Los sistemas de bases de datos también tuvieron interfaces Web a los datos.

## 1.3 Base de Datos

Una base de datos es un conjunto ordenado y estructurado de datos que representan una realidad objetiva y que están organizados independientemente de las aplicaciones, significa que puedan ser utilizadas y compartidas por usuarios y aplicaciones diferentes. O sea, que un BD puede considerarse una colección de datos variables en el tiempo.

Un sistema de gestión de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente.

Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información.
La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o los intentos de acceso sin autorización. Si los datos van a ser compartidos entre diversos usuarios, el sistema debe evitar posibles resultados anómalos.

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

## 1.4 Aplicaciones de los SGBD

Las bases de datos son ampliamente usadas. Las siguientes son algunas de sus aplicaciones más representativas:

- ✓ Bancos. Para información de los clientes, cuentas y préstamos, y transacciones bancarias.
- ✓ Líneas aéreas. Para reservas e información de planificación. Las líneas aéreas fueron de los primeros en usar las bases de datos de forma distribuida geográficamente (los terminales situados en todo el mundo accedían al sistema de bases de datos

centralizado a través de las líneas telefónicas y otras redes de datos).

- ✓ Universidades. Para información de los estudiantes, matrículas de las asignaturas y cursos.
- ✓ Transacciones de tarjetas de crédito. Para compras con tarjeta de crédito y generación mensual de extractos.
- ✓ Telecomunicaciones. Para guardar un registro de las llamadas realizadas, generación mensual de facturas, manteniendo el saldo de las tarjetas telefónicas de prepago y para almacenar información sobre las redes de comunicaciones.
- ✓ Finanzas. Para almacenar información sobre grandes empresas, ventas y compras de documentos formales financieros, como bolsa y bonos.
- ✓ Ventas. Para información de clientes, productos y compras.
- ✓ Producción. Para la gestión de la cadena de producción y para el seguimiento de la producción de elementos en las factorías, inventarios de elementos en almacenes y pedidos de elementos.
- ✓ Recursos humanos. Para información sobre los empleados, salarios, impuestos y beneficios, y para la generación de las nóminas.

La revolución de Internet a finales de la década de aumentó significativamente el acceso directo del usuario a las bases de datos. Las organizaciones convirtieron muchas de sus interfaces telefónicas a las bases de datos en interfaces Web, y pusieron disponibles en línea muchos servicios. Por ejemplo, cuando se accede a una tienda de libros en línea y se busca un libro o una colección de música se está accediendo a datos almacenados en una base de datos.
Cuando se solicita un pedido en línea, el pedido se almacena en una base de datos. Cuando se accede a un banco en un sitio Web y se consulta el estado de la cuenta y los movimientos, la información se recupera del sistema de bases de datos del banco. Cuando se accede a un sitio Web, la información personal puede ser recuperada de una base de datos para seleccionar los anuncios que se deberían mostrar. Más aún, los datos sobre los accesos Web pueden ser almacenados en una base de datos.

Así, aunque las interfaces de datos ocultan detalles del acceso a las bases de datos, y la mayoría de la gente ni siquiera es consciente de que están interactuando con una base de datos, el acceso a las bases de datos forma una parte esencial de la vida de casi todas las personas actualmente.

La importancia de los sistemas de bases de datos se puede juzgar de otra forma: actualmente, los vendedores de sistemas de bases de datos como Oracle están entre las mayores compañías software en el mundo, y los sistemas de bases de datos forman una parte importante de la línea de productos de compañías más diversificadas, como Microsoft e IBM.

## 1.5 Objetivos de los SGBD

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

### 1.5.1 Los objetivos fundamentales de los SBD

**1- Independencia de los datos y los programas de aplicación.**

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

- ✓ Diferentes aplicaciones necesitarán diferentes aspectos de los mismos datos (decimal o binario).
- ✓ Se debe poder modificar la estructura de almacenamiento o el método de acceso según los cambios en el fenómeno o proceso de la realidad sin necesidad de modificar los programas de aplicación (también para buscar mayor eficiencia).

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

**2- Minimización de la redundancia**

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

**3- Integración y sincronización de las bases de datos**

La integración consiste en garantizar una respuesta a los requerimientos de diferentes aspectos de los mismos datos por diferentes usuarios, de forma que, aunque el sistema almacene la información con cierta estructura y cierto tipo de representación, debe garantizar entregar al programa de aplicación datos que solicita y en la forma en que lo solicita.

Está vinculada a la sincronización, que consiste en la necesidad de garantizar el acceso múltiple y simultáneo a la BD, de modo que los datos puedan ser compartidos por diferentes usuarios a la vez. Están relacionadas, ya que lo usual es que diferentes usuarios trabajen con diferentes enfoques y requieran los mismos datos, pero desde diferentes puntos de vista.

**4- Integridad de los datos**

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

**5- Seguridad y protección de los datos**

Protección: garantizar el acceso autorizado a los datos, de forma de interrumpir cualquier intento de acceso no autorizado, ya sea por error del usuario o por mala intención. Seguridad: que el sistema de bases de datos disponga de métodos que garanticen la restauración de las BD al producirse alguna falla técnica, interrupción de la energía eléctrica,
etc.

**6- Facilidad de manipulación de la información**

Los usuarios de una BD pueden referirse a ella con las solicitudes para resolver muchos problemas diferentes. El SBD debe contar con la capacidad de una búsqueda rápida por diferentes criterios, permitir que los usuarios planteen sus demandas de una forma simple, aislándolo de las complejidades del tratamiento de los ficheros y del direccionado de los datos.

**7- Control centralizado**

Uno de los objetivos más importantes de los SBD es garantizar el control centralizado de la información. Permite controlar de manera sistemática y única los datos que se almacenan en la BD, así como el acceso a ella. Lo anterior implica que debe existir una persona o conjunto de personas que tenga la responsabilidad de los datos operacionales: el administrador de la BD, que puede considerarse parte integrante del SBD. Entre las tareas del administrador de la BD está:

- ✓ Decidir el contenido informativo de la BD
- ✓ Decidir la estructura de almacenamiento y la estrategia de acceso
- ✓ Garantizar el enlace con los usuarios
- ✓ Definir los chequeos de autorización y procedimientos de validación
- ✓ Definir la estrategia de reorganización de las BD para aumentar la eficiencia del sistema

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

Propiedades que deben tener las Bases de Datos para cumplir con los objetivos que persiguen:

- ✓ Estructuras independientes de la aplicaciones, significa que puedas ser utilizadas y compartidas por los usuarios y aplicaciones diferentes.
- ✓ Presentar la menor redundancia de datos posible. Con ello se ahorra tiempo de procesamiento, espacio de almacenamiento en los soportes y facilita la actualización de la información.

- ✓ Control centralizado, garantizando la seguridad, confiabilidad y precisión de los datos.

## 1.6 Arquitectura de un SGBD

Uno de los propósitos principales de un sistema de bases de datos es proporcionar a los usuarios una visión abstracta de los datos. Es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos.
Para que el sistema sea útil debe recuperar los datos eficientemente. Esta preocupación ha conducido al diseño de estructuras de datos complejas para la representación de los datos en la base de datos.
La arquitectura se divide en tres niveles generales: nivel de vistas (externo), lógico (conceptual) y físico (interno).

**Nivel físico:** El nivel más bajo de abstracción, describe cómo se almacenan realmente los datos. En el nivel físico se describen en detalle las estructuras de datos complejas de bajo nivel

**Nivel lógico:** El siguiente nivel más alto de abstracción describe qué datos se almacenan en la base de datos y qué relaciones existen entre esos datos. La base de datos completa se describe así en términos de un número pequeño de estructuras relativamente simples. Aunque la implementación de estructuras simples en el nivel lógico puede involucrar estructuras complejas del nivel físico, los usuarios del nivel lógico no necesitan preocuparse de esta complejidad. Los administradores de bases de datos, que deben decidir la información que se mantiene en la base de datos, usan el nivel lógico de abstracción.

Nivel de vistas: El nivel más alto de abstracción describe sólo parte de la base de datos completa. Es lo que el usuario final puede visualizar del sistema terminado, describe sólo una parte de la base de datos al usuario acreditado para verla. El sistema puede proporcionar muchas vistas para la misma base de datos.

Además de esconder detalles del nivel lógico de la base de datos, las vistas también proporcionan un mecanismo de seguridad para evitar que los usuarios accedan a ciertas partes de la base de datos. Por ejemplo, los cajeros de un banco ven únicamente la parte de la base de datos que tiene información de cuentas de clientes; no pueden acceder a la información referente a los sueldos de los sueldos de los empleados.

## 1.7 Estructura de un SGBD

Un sistema de bases de datos se divide en módulos que se encargan de cada una de las responsabilidades del sistema completo. Los componentes funcionales de un sistema de bases de datos se pueden dividir a grandes rasgos en los componentes: gestor de almacenamiento y procesador de consultas.

Un gestor de almacenamiento es un módulo de programa que proporciona la interfaz entre los datos de bajo nivel en la base de datos y los programas de aplicación y consultas emitidas al

sistema. El gestor de almacenamiento es responsable de la interacción con el gestor de archivos. Los datos en bruto se almacenan en disco usando un sistema de archivos, que está disponible habitualmente en un sistema operativo convencional. El gestor de almacenamiento traduce las diferentes instrucciones LMD a órdenes de un sistema de archivos de bajo nivel. Así, el gestor de almacenamiento es responsable del almacenamiento, recuperación y actualización de los datos en la base de datos.

**Los componentes del gestor de almacenamiento incluyen:**

- ✓ Gestor de autorización e integridad, que comprueba que se satisfagan las restricciones de integridad y la autorización de los usuarios para acceder a los datos.
- ✓ Gestor de transacciones, que asegura que la base de datos quede en un estado consistente (correcto) a pesar de los fallos del sistema, y que las ejecuciones de transacciones concurrentes ocurran sin conflictos.
- ✓ Gestor de archivos, que gestiona la reserva de espacio de almacenamiento de disco y las estructuras de datos usadas para representar la información almacenada en disco.
- ✓ Gestor de memoria intermedia, que es responsable de traer los datos del disco de almacena- miento a memoria principal y decidir qué datos tratar en memoria caché. El gestor de memoria intermedia es una parte crítica del sistema de bases de datos, ya que permite que la base de datos maneje tamaños de datos que son mucho mayores que el tamaño de la memoria principal.
- ✓ El gestor de almacenamiento implementa varias estructuras de datos como parte de la implementación física del sistema:
- ✓ Archivos de datos, que almacenan la base de datos en sí.
- ✓ Diccionario de datos, que almacena metadatos acerca de la estructura de la base de datos, en particular, el esquema de la base de datos.
- ✓ Índices, que proporcionan acceso rápido a elementos de datos que tienen valores particulares.

**Los componentes del procesador de consultas incluyen:**

- ✓ Intérprete del Lenguaje de Definición de Datos (LDD), que interpreta las instrucciones del LDD y registra las definiciones en el diccionario de datos.
- ✓ Compilador del Lenguaje de Manipulación de Datos (LMD), que traduce las instrucciones del LMD en un lenguaje de consultas a un plan de evaluación que consiste en instrucciones de bajo nivel que entiende el motor de evaluación de consultas.
- ✓ Una consulta se puede traducir habitualmente en varios planes de ejecución alternativos que proporcionan el mismo resultado. El compilador del LMD también realiza optimización de consultas, es decir, elige el plan de evaluación de menor coste de entre todas las alternativas. Motor de evaluación de consultas, que ejecuta las instrucciones de bajo nivel generadas por el compilador del LMD

## 1.8 Arquitectura de las Aplicaciones

La mayoría de usuarios de un sistema de bases de datos no están situados actualmente junto al sistema de bases de datos, sino que se conectan a él a través de una red.
Se puede diferenciar entonces entre las máquinas cliente, en donde trabajan los usuarios remotos de la base de datos, y las máquinas servidor, en las que se ejecuta el sistema de bases de datos.
Las aplicaciones de bases de datos se dividen usualmente en dos o tres partes. En una arquitectura de dos capas, la aplicación se divide en un componente que reside en la máquina cliente, que llama a la funcionalidad del sistema de bases de datos en la máquina servidor mediante instrucciones del lenguaje de consultas.
En cambio, en una arquitectura de tres capas, la máquina cliente actúa simplemente como frontal y no contiene ninguna llamada directa a la base de datos. En su lugar, el cliente se comunica con un servidor de aplicaciones, usualmente mediante una interfaz de formularios. El servidor de aplicaciones, a su vez, se comunica con el sistema de bases de datos para acceder a los datos.
La lógica de negocio de la aplicación, que establece las acciones a realizar bajo determinadas condiciones, se incorpora en el servidor de aplicaciones, en lugar de ser distribuida a múltiples clientes. Las aplicaciones de tres capas son más apropiadas para grandes aplicaciones, y para las aplicaciones que se ejecutan en World Wide Web.

**Conclusiones**

- ✓ Las bases de datos surgen en un proceso evolutivo de tratamiento de la información.
- ✓ Elementos de la arquitectura e importancia de las relaciones entre los tres niveles de la arquitectura.

# CAPITULO 2: MODELACIÓN CONCEPTUAL DE SISTEMAS DE BASES DE DATOS

## 2.1 Modelación Conceptual de Sistemas de Bases de Datos. El Modelo Entidad-Relación

Contenido

- Características del Modelo Conceptual.
- Modelo Entidad-Relación.
- Diagrama Entidad-Relación.

### 2.1.1 Características del Modelo Conceptual

Como vimos en la conferencia anterior al hablar de los 3 niveles de abstracción, el proceso de diseño de la BD transita a través de una serie de pasos en los cuales se va avanzando de un nivel de abstracción menor a otro más profundo, mediante la elaboración de una sucesión de modelos.

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

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

**Modelos de Datos**

**Modelo:** Es una representación de la realidad que contiene las características generales de algo que se va a realizar. En base de datos, esta representación la elaboramos de forma gráfica.

**¿Qué es modelo de datos?**

Es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos, semántica asociada a los datos y restricciones de consistencia.

**Los modelos de datos se dividen en tres grupos:**

- Modelos lógicos basados en objetos.
- Modelos lógicos basados en registros.
- Modelos físicos de datos.

**Modelos lógicos basados en objetos.**

Se usan para describir datos en los niveles conceptual y de vistas, es decir, con este modelo representamos los datos de tal forma como nosotros los captamos en el mundo real, tienen una capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. Existen diferentes modelos de este tipo, pero el más utilizado por su sencillez y eficiencia es el modelo Entidad-Relación que veremos más adelante en esta misma conferencia

**Modelos lógicos basados en registros.**

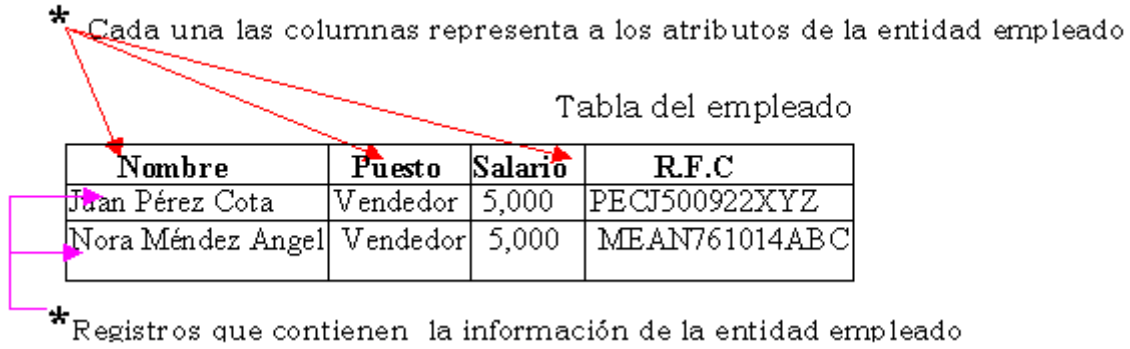
Se utilizan para describir datos en los niveles conceptual y físico. Estos modelos utilizan registros e instancias para representar la realidad, así como las relaciones que existen entre estos registros. A diferencia de los modelos de datos basados en objetos, se usan para especificar la estructura lógica global de la base de datos y para proporcionar una descripción a nivel más alto de la implementación.

**Los tres modelos de datos más ampliamente aceptados son:**

- Modelo Relacional
- Modelo de Red
- Modelo Jerárquico

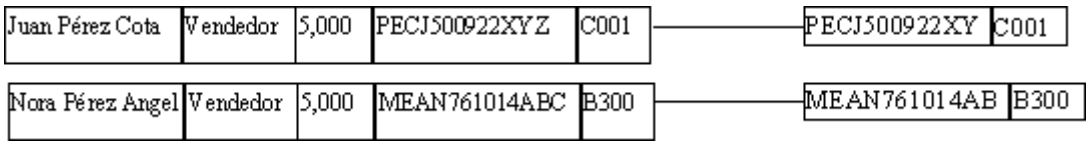
**Modelo Relacional**: En este modelo se representan los datos y las relaciones entre estos, a través de una colección de tablas, en las cuales los renglones (tuplas) equivalen a los cada uno de los registros que contendrá la base de datos y las columnas corresponden a las características (atributos) de cada registro localizado en la tupla.

**Por ejemplo:**

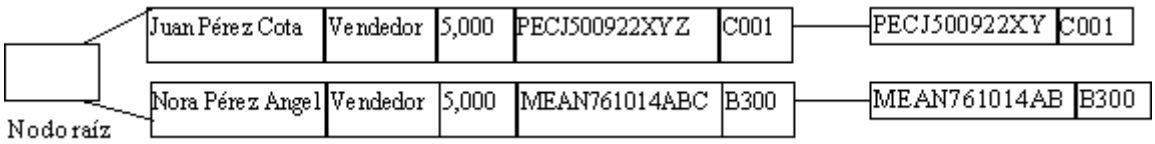



**Modelo de Red:** Este modelo representa los datos mediante colecciones de registros y sus relaciones se representan por medio de enlaces, los cuales pueden verse como punteros. Los registros se organizanen un conjunto de gráficas arbitrarias.

**Ejemplo:**




**Modelo Jerárquico:** Es similar al modelo de red en cuanto a las relaciones y datos, ya que estos se representan por medio de registros y sus enlaces. La diferencia radica en que están organizados por conjuntos de árboles en lugar de gráficas arbitrarias.




**Modelos físicos de datos**

Se usan para describir a los datos en el nivel más bajo, aunque existen muy pocos modelos de este tipo, básicamente capturan aspectos de la implementación de los sistemas de base de datos. Existen dos clasificaciones de este tipo que son:

- Modelo unificador
- Memoria de elementos.

Entre los diferentes modelos que explicamos anteriormente estudiaremos en detalle 2 modelos: el Modelo Entidad-Relación y el Modelo Relacional.

### 2.1.2 El Modelo Entidad-Relación (MER)

El MER es un modelo de datos de alto nivel. Está basado en una percepción de un mundo real que consiste en una colección de objetos básicos, denominados entidades, y de relaciones entre estos objetos, así como las características de estos objetos llamados atributos.

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

Entidades y conjunto de entidades

Una entidad es un objeto que existe y se distingue de otros objetos de acuerdo a sus características llamadas atributos. Las entidades pueden ser concretas como una persona o abstractas como una fecha. Por ejemplo, una silla es una entidad, un automóvil, un empleado, un profesor, un estudiante, que son cosas concretas; pero también puede ser algo no tangible, como un suceso cualquiera, una cuenta de ahorro, o un concepto abstracto.

Un conjunto de entidades es un grupo de entidades del mismo tipo. Por ejemplo el conjunto de entidades CUENTA, podría representar al conjunto de cuentas de un banco X, o ALUMNO representa a un conjunto de entidades de todos los alumnos que existen en una institución.

Una entidad se caracteriza y distingue de otra por los atributos, en ocasiones llamados propiedades, que representan las características de una entidad. Los atributos de una entidad pueden tomar un conjunto de valores permitidos al que se le conoce como **dominio** del atributo. Así cada entidad se describe por medio de un conjunto de parejas formadas por el atributo y el valor de dato. Habrá una pareja para cada atributo del conjunto de entidades.

**Ejemplo:**

Entidad alumno con los atributos No control, Nombre y Especialidad.

| Nombre atributo | Valor |
|---|---|
| No control | 96310418 |
| Nombre | Sánchez Osuna Ana |
| Especialidad | Ing. Mecánica |

Una colección identificable de campos asociados es un artículo o registro y representa un objeto con sus propiedades. Una ocurrencia de artículo consiste en un grupo de ocurrencias de campos relacionados, representando una asociación entre ellos. Por ejemplo, tenemos un artículo correspondiente al objeto profesor.

El nombre o tipo de artículo puede ser PROFESOR, que esté formado por los siguientes tipos de campos o atributos:

NUM_IDENT: número de identidad del profesor
NOM_PROF: nombre del profesor

CAT_DOC: categoría docente del profesor
DPTO: departamento docente al que pertenece el profesor

Una ocurrencia de este artículo puede ser:

45112801731 Hdez Roberto P.A. Computación

### 2.1.3 Diagrama Entidad Relación (DER)

Constituye la representación gráfica asociada al MER.

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

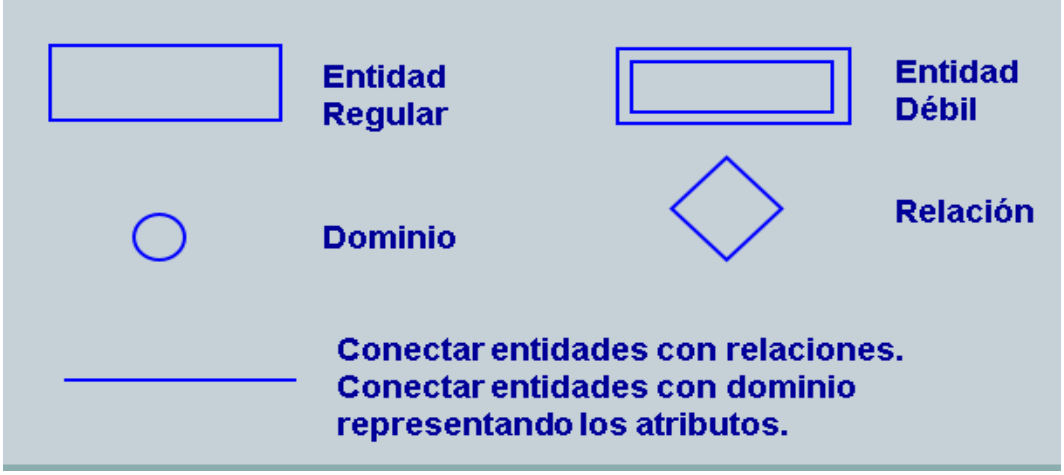



**Figura 1**. Diagrama Entidad Relación**.**

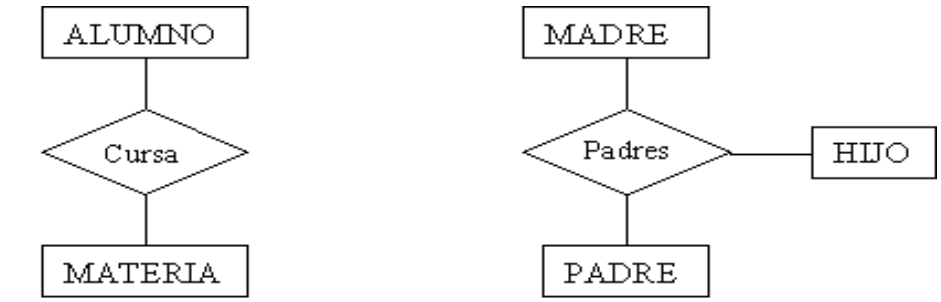
**Relaciones y Conjunto de Relaciones**

Una relación es la asociación que existe entre dos a más entidades.

Un conjunto de relaciones es un grupo de relaciones del mismo tipo.

La cantidad de entidades en una relación determina el grado de la relación, por ejemplo la relación ALUMNO-MATERIA es de grado 2, ya que intervienen la entidad ALUMNO y la entidad MATERIA, la relación PADRES, puede ser de grado 3, ya que involucra las entidades PADRE, MADRE e HIJO.

**Figura 2.** Ejemplo Entidad Relación.




Ejemplo de relación de grado 2

Ejemplo de relación de grado 3

Aunque el modelo E-R permite relaciones de cualquier grado, la mayoría de las aplicaciones del modelo sólo consideran relaciones del grado 2. Cuando son de tal tipo, se denominan relaciones binarias.

La función que tiene una relación se llama **papel**, generalmente no se especifican los papeles o roles, a menos que se quiera aclarar el significado de una relación.

**Tipos de relaciones:**

- **Relación uno a uno. (1 a 1)**

Se presenta cuando existe una relación como su nombre lo indica uno a uno. Una entidad del tipo A solo se puede relacionar con una entidad del tipo B, y viceversa;

Por ejemplo: la relación asignación de automóvil que contiene a las entidades EMPLEADO, AUTO, es una relación 1 a 1, ya que asocia a un empleado con un único automóvil por lo tanto ningún empleado posee más de un automóvil asignado, y ningún vehículo se asigna a más de un trabajador.

Es representado gráficamente de la siguiente manera:

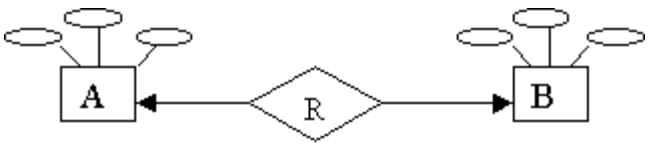



**Figura 3.** Relación uno a uno. (1 a 1).

**A:** Representa a una entidad de cualquier tipo diferente a una entidad B.
**R**: en el diagrama representa a la relación que existe entre las entidades.
El extremo de la flecha que se encuentra punteada indica el uno de la relación, en este caso, una entidad A ligada a una entidad B.

- **Relación uno a muchos. (1 a M)**

Significa que una entidad del tipo A puede relacionarse con cualquier cantidad de entidades del tipo B, y una entidad del tipo B solo puede estar relacionada con una entidad del tipo A.

Su representación gráfica es la siguiente:

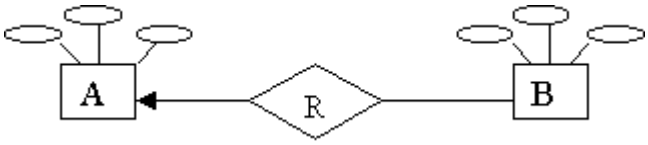



**Figura 4.** Relación uno a muchos. (1 a M).

Nótese en este caso que el extremo punteado de la flecha de la relación de A y B, indica una entidad A conectada a muchas entidades B.

- **Muchos a uno. (M a 1)**

Indica que una entidad del tipo B puede relacionarse con cualquier cantidad de entidades del tipo A, mientras que cada entidad del tipo A solo puede relacionarse con solo una entidad del tipo B.

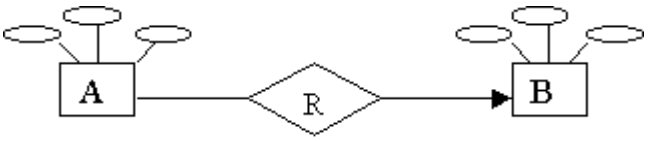



**Figura 5.** Muchos a uno. (M a 1).

- **Muchas a muchas. (M a M)**

Establece que cualquier cantidad de entidades del tipo A pueden estar relacionados con cualquier cantidad de entidades del tipo B.

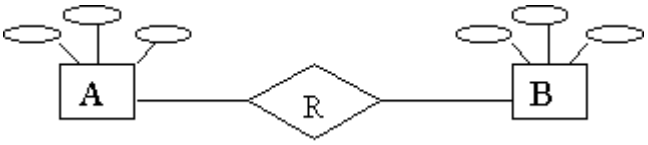



**Figura 6.** Muchas a muchas. (M a M).

A los tipos de relaciones antes descritos, también se le conoce como cardinalidad.

## 2.2 Cardinalidad o Llaves primarias

Como ya se ha mencionado anteriormente, la distinción de una entidad entre otra se debe a sus atributos, lo cual lo hacen único. Una llave primaria es aquel atributo el cual consideramos clave para la identificación de los demás atributos que describen a la entidad.

Por ejemplo, si consideramos la entidad ALUMNO, podríamos tener los siguientes atributos: Nombre, Semestre, Especialidad, Dirección, Teléfono, Número de control, de todos estos atributos el que podremos designar como llave primaria es el número de control, ya que es diferente para cada alumno y este nos identifica en la institución.

Claro que puede haber más de un atributo que pueda identificarse como llave primaria en este caso se selecciona la que consideremos más importante, los demás atributos son denominados **llaves secundarias.**

Una clave o llave primaria es indicada gráficamente en el modelo E-R con una línea debajo del nombre del atributo o lo que es lo mismo se subraya el atributo.

Ejemplos de modelos E-R, considerando las cardinalidades que existen entre ellos:

**Relación Uno a Uno**

**Problema:**

Diseñar el modelo E-R, para la relación Registro de automóvil que consiste en obtener la tarjeta de circulación de un automóvil con los siguientes datos:

- Automóvil- Modelo, Placas, Color
- Tarjeta de circulación -Propietario, No serie, Tipo.

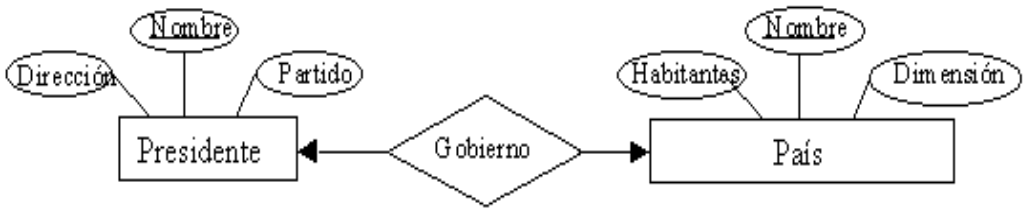



**Figura 7.** Ejemplo Relación Uno a Uno.

Indicamos con este ejemplo que existe una relación de pertenencia de uno a uno, ya que existe una tarjeta de circulación registrada por cada automóvil.

En este ejemplo, representamos que existe un solo presidente para cada país.

**Relación uno a muchos**

El siguiente ejemplo indica que un cliente puede tener muchas cuentas, pero que una cuenta puede llegar a pertenecer a un solo cliente (Decimos puede, ya que existen cuentas registradas a favor de más de una persona).

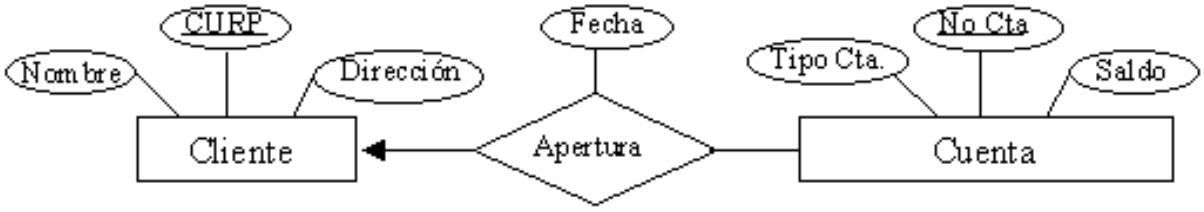



**Figura 8:** Ejemplo Relación Uno a Muchos.

**Relación muchos a muchos**

El ejemplo de los estudiantes y las asignaturas donde un estudiante recibe varias asignaturas y estas a su vez la reciben varios estudiantes.

MER Extendido


Es posible extender la capacidad semántica del MER aplicando sobre sus objetos básicos (entidad y relación) diferentes operaciones. En este caso se habla de Modelo Entidad Relación extendido.

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

## 2.3 Generalización

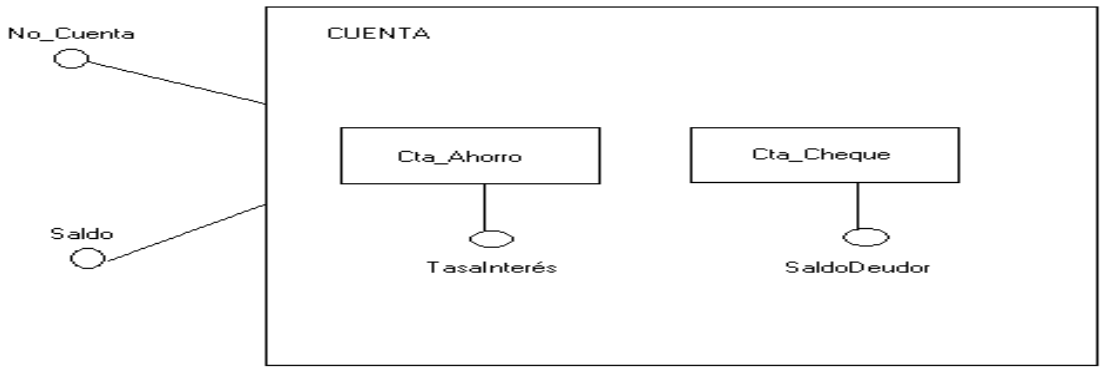
Es el resultado de la unión de 2 o más conjuntos de entidades (de bajo nivel) para producir un conjunto de entidades de más alto nivel.

La generalización consiste en identificar todos aquellos atributos iguales de un conjunto de entidades para formar una entidad(es) global(es) con dichos atributos semejantes, dicha entidad(es) global(es) quedara a un nivel más alto al de las entidades origen.

**Ejemplo:**
Se tiene las entidades Cta_Ahorro y Cta_Cheques, ambas tienen los atributos semejantes de No_Cta y Saldo, aunque además de estos dos atributos, Cta_Ahorro tiene el atributo Tasa Interés y Cta_Cheques el atributo Saldo Deudor. De todos estos atributos podemos juntar (generalizar) No_Cta y Saldo que son iguales en ambas entidades.

Entonces tenemos:

**Figura 9.** Generalización.




Podemos leer esta gráfica como: La entidad Cta_Ahorro hereda de la entidad CUENTA los atributos No_Cta y saldo, además del atributo de TasaInteres, de forma semejante Cta_cheque tiene los atributos de No_Cta, Saldo y Saldo Deudor.

Como podemos observar la Generalización trata de eliminar la redundancia (repetición) de atributos, al englobar los atributos semejantes. La entidad(es) de bajo nivel heredan todos los atributos correspondientes.

Como se puede apreciar Cuenta es el supertipo de esta jerarquía y constituye la generalización de los subtipos de entidades Cta_Ahorro y Cta_Cheque y a su vez estos son una especialización del supertipo Cuenta.

## 2.4 Agregación

La agregación surge de la limitación que existe en el modelado de E-R, al no permitir expresar las relaciones entre relaciones de un modelo E-R en el caso de que una relación X se quiera unir con una entidad cualquiera para formar otra relación.

La Agregación consiste en agrupar por medio de un rectángulo a la relación (representada por un rombo) junto con las entidades y atributos involucrados en ella, para formar un grupo que es considerado una entidad y ahora sí podemos relacionarla con otra entidad.

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

Con la operación de Agregación esta situación se resuelve fácilmente, tal y como se muestra en la figura.

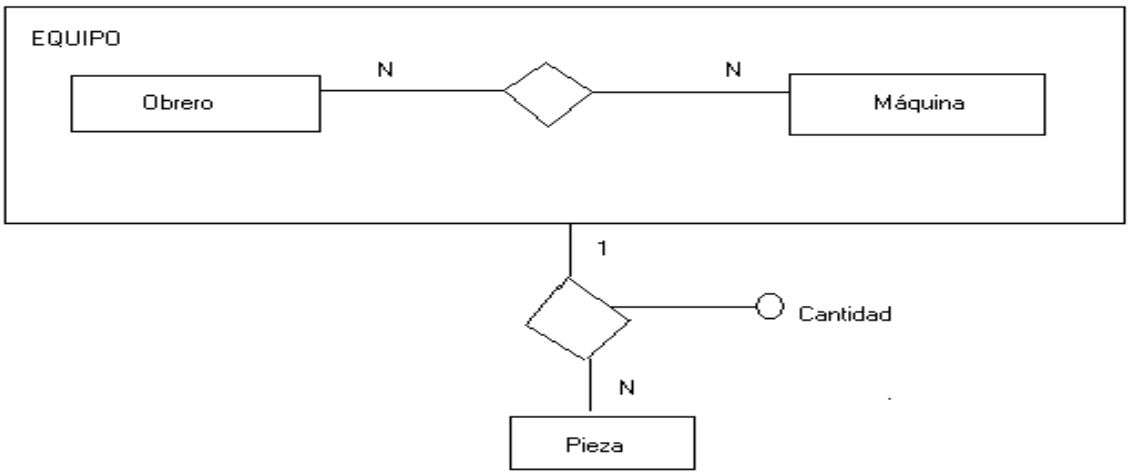



**Figura 10.** Agregación.

## 2.5 Reducción de diagramas E-R a tablas

Un diagrama E-R, puede ser representado también a través de una colección de tablas. Para cada una de las entidades y relaciones existe una tabla única a la que se le asigna como nombre el del conjunto de entidades y de las relaciones respectivamente, cada tabla tiene un número de columnas que son definidas por la cantidad de atributos, las columnas tendrán el nombre del atributo.

Ejemplo: Venta, en la que intervienen las entidades de Empleado con los atributos RFC o CI, nombre, puesto, salario y Artículo con los atributos Clave, descripción, costo.

Cuyo diagrama E-R es el siguiente:

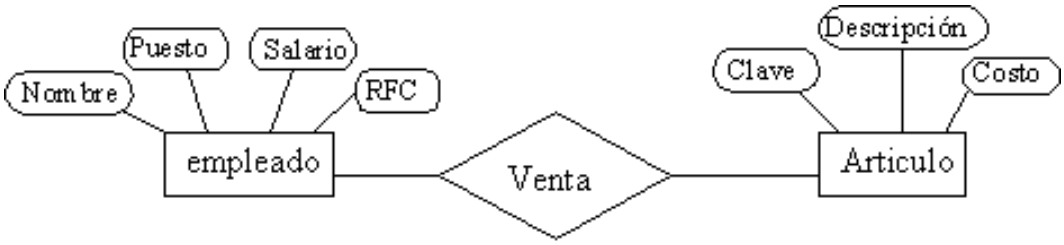



**Figura 11.** Diagrama E-R.

Entonces las tablas resultantes siguiendo la descripción anterior son:

**Tabla 1.** Empleado.

| Nombre | Puesto | Salario | RFC (CI) |
|---|---|---|---|
| Teófilo | Vendedor | 2000 | 56121203642 |
| Cesar | Auxiliar ventas | 1200 | 65030204567 |

**Tabla 2**. Artículo.

| Clave | Descripción | Costo |
|---|---|---|
| A100 | Abanico | 460 |
| C260 | Colcha matrimonial | 1200 |

**Tabla 3.** Venta.

| RFC | Clave |
|---|---|
| 56121203642 | C260 |
| 65030204567 | A100 |

Nótese que en la tabla de relación Venta contiene como atributos a las llaves primarias de las entidades que intervienen en dicha relación, en caso de que exista un atributo en las relaciones, este atributo es anexado como una fila más de la tabla.

Por ejemplo, si anexamos el atributo fecha a la relación venta, la tabla que se originaria sería la siguiente:

**Tabla 4.** Relación Venta.

| RFC | Clave | Fecha |
|---|---|---|
| 56121203642 | C260 | 10/12/96 |
| 65030204567 | A100 | 11/12/96 |

## 2.6 Estudio Independiente

1. Estudiar el contenido relacionado con el tema “Modelo Entidad Relación”
   Mato García, Rosa María. “Sistemas de Bases de Datos”. Capítulo 2.
2. Diseñar el DER del siguiente ejercicio:

En un organismo se reciben distintos productos que son importados de diferentes países. Es necesario controlar las cantidades que se importan de cada país y el valor de las importaciones.

- Las propiedades de los productos son: número, nombre, unidad de medida, peso y precio unitario.
- Las propiedades de los países son: número, nombre, zona geográfica y área de moneda.
- Existe relación comercial con varios países de donde se importan varios productos en ciertas cantidades
- Final del formulario

# CAPITULO 3: MODELACIÓN CONCEPTUAL DE BASES DE DATOS.

## 3.1 El Modelo Relacional. Normalización

Contenido

- Modelo Relacional
- Normalización.
- Formas Normales

### 3.1.1 Modelo Relacional

El modelo relacional se ha establecido actualmente como el principal modelo de datos para las aplicaciones de procesamiento de datos. Ha conseguido la posición principal debido a su simplicidad, que facilita el trabajo del programador en comparación con otros modelos anteriores como el de red y el jerárquico.

Se basa en la teoría matemática de las relaciones, suministrándose por ello una fundamentación teórica que permite aplicar todos los resultados de dicha teoría a problemas tales como el diseño de sublenguajes de datos y otros.

Representación de una relación

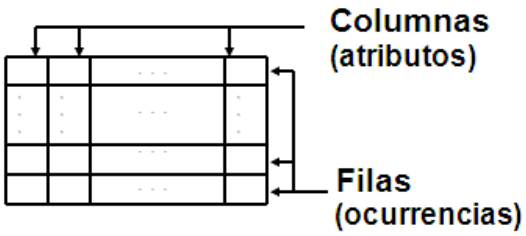



**Figura 12.** Representación de una relación.

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

Cada relación está compuesta por filas (las ocurrencias de los objetos) y por columnas (los atributos o campos que toman valores en sus respectivos dominios).

Veamos como un ejemplo de suministradores y productos se puede representar fácil y claramente mediante el modelo relacional.

Los atributos de estas dos entidades son:

Suministrador: número, que lo identifica, nombre, tipo y municipio donde radica.
Producto: número, que lo identifica, nombre, precio unitario y peso.

Además, un suministrador puede suministrar muchos productos y un producto puede ser suministrado por varios suministradores. Se conoce la cantidad de un determinado producto que suministra un suministrador dado.

SUMIN (SNUM, SNOM, MUN, TIPO)

PROD (PNUM, PNOM, PRECIO, PESO)

SP (SNUM, PNUM, CANT)

La representación en el modelo relacional es más simple que con el modelo jerárquico y el modelo reticular, ya que con tres tablas se tiene todo el modelo representado.

**Suministrador**

| SNUM | SNOM | TIPO | MUN |
|---|---|---|---|
| S1 | Pérez | 30 | Cerro |
| S2 | Ramos | 10 | Plaza |
| S3 | Arenas | 20 | Cerro |
| S4 | Valle | 20 | Playa |
| S5 | López | 15 | Plaza |

**Producto**

| PNUM | PNOM | PRECIO | PESO |
|---|---|---|---|
| P1 | Clavo | 0,10 | 12 |
| P2 | Tuerca | 0,15 | 17 |
| P3 | Martillo | 3,50 | 80 |
| P4 | Tornillo | 0,20 | 10 |
| P5 | Alicate | 2,00 | 50 |
| P6 | Serrucho | 4,00 | 90 |

**SP**

| SNUM | PNUM | CANT |
|---|---|---|
| S1 | P1 | 3 |
| S1 | P2 | 2 |
| S1 | P3 | 4 |
| S1 | P4 | 2 |
| S1 | P5 | 1 |
| S1 | P6 | 1 |
| S2 | P1 | 3 |
| S2 | P2 | 4 |
| S3 | P3 | 4 |
| S3 | P5 | 2 |
| S4 | P2 | 2 |
| S4 | P4 | 3 |
| S4 | P5 | 4 |

**Figura 13**. Representación Modelo Relación.

Las diversas formas de expresar las recuperaciones dan lugar a los lenguajes relacionales cuyas formas más representativas son:

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

Esto no lo veremos en este curso.

Ventajas del modelo relacional:

- Una de las principales ventajas es su simplicidad, pues el usuario formula sus demandas en términos del contenido informativo de la BD sin tener que atender a las complejidades de la realización del sistema, lo que implica gran independencia de los datos.
- La información se maneja en forma de tablas, lo que constituye una manera familiar de representarla.
- Si se tienen relaciones normalizadas, no surgen dificultades grandes en la actualización.

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

Creación: añadir un producto P7. Se agrega la nueva ocurrencia en la tabla PRODUCTO. Es posible hacerlo aunque ningún suministrador lo suministre.

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

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

No obstante, veremos que el proceso de normalización no es suficiente hasta el punto aquí visto.

### 3.1.2 Normalización

La teoría de la normalización se ha desarrollado para obtener estructuras de datos eficientes que eviten las anomalías de actualización.

La normalización es el proceso de simplificar la relación entre los campos de un registro. Por medio de la normalización un conjunto de datos en un registro se reemplaza por varios registros que son más simples y predecibles y, por lo tanto, más manejables. La normalización se lleva a cabo por cuatro razones:

- ✓ Estructurar los datos de forma que se puedan representar las relaciones pertinentes entre los datos.
- ✓ Permitir la recuperación sencilla de los datos en respuesta a las solicitudes de consultas y reportes.
- ✓ Simplificar el mantenimiento de los datos actualizándolos, insertándolos y borrándolos.
- ✓ Reducir la necesidad de reestructurar o reorganizar los datos cuando surjan nuevas aplicaciones.

La teoría de normalización tiene como fundamento el concepto de formas normales; se dice que una relación está en una determinada forma normal si satisface un conjunto de restricciones.

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

Existen, además, la cuarta (4FN) y la quinta (5FN) formas normales.

### 3.1.3 Formas normales

Son las técnicas para prevenir las anomalías en las tablas. Dependiendo de su estructura, una tabla puede estar en primera forma normal, segunda forma normal o en cualquier otra.

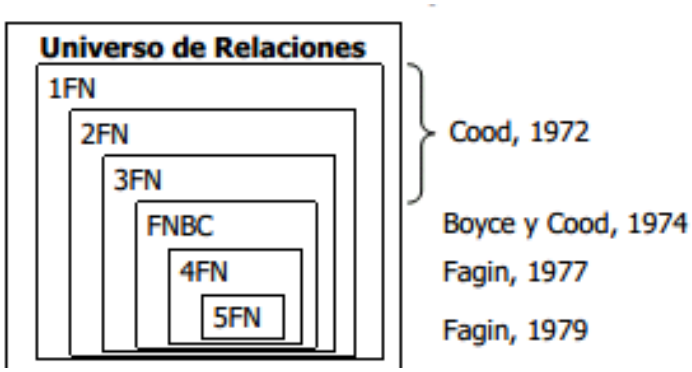



**Figura 14.** Forma Normales.

Para evitar anomalías de actualización, es recomendable llegar al menos hasta la tercera forma normal o, mejor aún, hasta la forma normal de Boyce-Codd.

**Primera Forma Normal:**

Una relación R se encuentra en 1FN si y solo sí por cada renglón columna contiene valores atómicos.

Abreviada como 1FN, se considera que una relación se encuentra en la primera forma normal cuando cumple lo siguiente:

1. Las celdas de las tablas poseen valores simples y no se permiten grupos ni arreglos repetidos como valores, es decir, contienen un solo valor por cada celda.
2. Todos los ingresos en cualquier columna (atributo) deben ser del mismo tipo.
3. Cada columna debe tener un nombre único, el orden de las columnas en la tabla no es importante.
4. Dos filas o renglones de una misma tabla no deben ser idénticas, aunque el orden de las filas no es importante.

Por lo general la mayoría de las relaciones cumplen con estas características, así que podemos decir que la mayoría de las relaciones se encuentran en la primera forma normal.

Visto de forma más sencilla para que la relación esté en 1FN no pueden existir grupos repetitivos.

Para ejemplificar como se representan gráficamente las relaciones en primera forma normal consideremos la relación alumno cursa materia cuyo diagrama E-R es el siguiente:

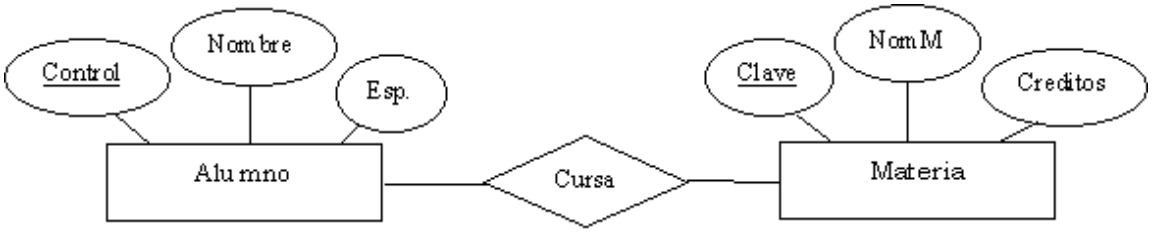



**Figura 15**. Primera Forma Normales**.**

Como esta relación maneja valores atómicos, es decir un solo valor por cada uno de los campos que conforman a los atributos de las entidades, ya se encuentra en primera forma normal, gráficamente así representamos a las relaciones en 1FN.

Para explicar el contenido veamos un ejemplo: Pedido de productos

**Tabla 5.** Ejemplo: Pedido de productos.

| Fecha: 16/2/83 | | | Pedido No.: 123456<br>Proveedor No.: 75621<br>Nombre Prov.: J. Pérez<br>Dir. Prov.: Cerro | |
|---|---|---|---|---|
| Deseamos envíen: | | | | |
| No. Producto | Descripción | Precio Unitario | Cantidad | Total |
| 969715 | Televisor | 600 | 1 | 600 |
| 439124 | Antena | 20 | 10 | 200 |
| 439126 | Espiga | 10 | 10 | 100 |
| | | | | Importe Total: 900 |

El análisis de este pedido muestra que los siguientes datos son de interés; # pedido es único y se utiliza para referirse a un pedido, por tanto, se puede usar como clave (llave).

En forma de relación se escribiría:

PEDIDO (nuped, fecha, nuprov, noprov, direc, nuprod, desc, prun, cant, prprod, prped)

Se puede observar que la relación PEDIDO contiene cinco grupos repetitivos: nuprod, desc, prun, cant, prprod, ya que un pedido puede contener más de una línea de pedido y, por lo tanto, puede contener varios números de producto (nuprod), varias descripciones de producto (desc), varios precios unitarios (prun), varias cantidades (cant) y varios precios por concepto del producto (prprod).

Hay que eliminar esos grupos repetitivos para que la relación esté en 1FN. Para ello se crea:

1. Una relación para los campos que sean únicos, es decir, se dejan en la relación original solo los atributos que no son repetitivos:

   PEDIDO (nuped, fecha, nuprov, noprov, direc, prped)

2. Una relación para los grupos repetitivos, es decir, se extraen en una nueva relación los atributos repetitivos, además de la llave primaria de la relación original:

PED-PROD (nuped, nuprod, desc, prun, cant, prprod)

Ambos tienen como llave o parte de la llave a nuped. Pero en PED-PROD es necesaria la llave compuesta para identificar los productos individuales.

Ahora estas nuevas dos relaciones en 1FN modelan el fenómeno que nos ocupa. Los problemas de actualización mencionados anteriormente quedan resueltos con este nuevo modelo. En lugar de tener varios valores en cada campo de acuerdo a la cantidad de líneas de pedido, tal y como ocurría en la tabla PEDIDO original, se tienen varias ocurrencias en la tabla PED-PROD, una por cada producto que se solicita en el pedido. Esto permite que se soliciten tantos productos como se desee en cada pedido, pues solo significa agregar una nueva ocurrencia en la relación PED-PROD por cada producto solicitado.

Sin embargo, este modelo en 1FN tiene aún problemas de actualización, como se muestra en las siguientes operaciones:

Creación: la información sobre un nuevo producto no se puede insertar si no hay un pedido que lo incluya.
Supresión: eliminar una línea de pedido que sea la única que pida un producto implica perder la información del producto.
Modificación: por cada línea de pedido en la que se solicite determinado producto se tiene ocurrencia en PED-PROD, que repite la información sobre este. Si cambia algún atributo del producto, entonces es necesario hacer muchas actualizaciones.

Entonces será necesario aplicar formas normales más fuertes a este modelo para eliminar los problemas de actualización que presenta, como veremos a continuación.

**Segunda Forma Normal.**

Para definir formalmente la segunda forma normal requerimos saber que es una **dependencia funcional.**

**DF:** Consiste en identificar que atributos dependen de otro(s) atributo(s).

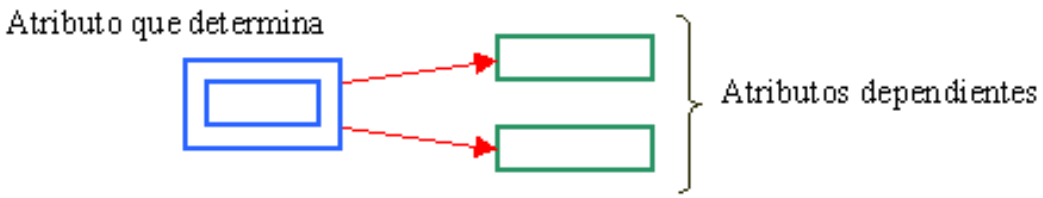



**Figura 16.** Dependencia funcional.

.

**Definición formal:**

Una relación R está en 2FN si y solo si está en 1FN y los atributos no primos dependen funcionalmente de la llave primaria.

Una relación se encuentra en segunda forma normal, cuando cumple con las reglas de la primera forma normal y todos sus atributos que no son claves (llaves) dependen por completo de la clave. De acuerdo con esta definición, cada tabla que tiene un atributo único como clave, está en segunda forma normal.

Entonces este segundo paso se aplica sólo con relación a llaves compuestas.
Continuando con el ejemplo de los Pedidos de productos, habíamos visto que en la relación PED-PROD subsistían problemas de actualización. Analicemos las DF que existen en dicha relación:

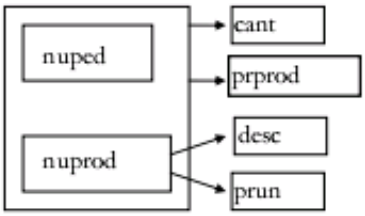



**Figura 17.** Segundo paso se aplica sólo con relación a llaves compuestas.

Esta relación no está en 2FN, pues desc y prun no dependen funcional y completamente de la llave (nuped, nuprod).

La 2FN se hace:

1. Creando una relación para todos los atributos que dependen funcional y completamente de la llave (y los atributos que no se analizan por ser atributos llaves, pertenecientes a claves candidatas).

   PED-PROD (nuped, nuprod, cant, prprod)

2. Creando una relación para los atributos que dependan de cada parte (subconjunto) de la llave. La llave de la relación así formada será la parte (subconjunto) de la llave primaria de la cual dependen los atributos.

   PRODUCTO (nuprod, desc, prun)

**Los problemas planteados en la 1FN se resuelven con la 2FN. Veamos:**

Creación: se puede insertar la información sobre un producto aunque no haya un pedido que lo solicite.
Supresión: se puede eliminar una línea de pedido y no se pierde la información sobre el producto, aunque sea el único pedido que pide ese producto.
Modificación: si cambia un atributo del producto, solo hay que cambiarlo en un lugar. Se elimina redundancia.

Pero aún tenemos problemas en este caso, que son similares a los vistos, pero con la relación PEDIDO y, específicamente, cuando se trata de insertar, eliminar o modificar la información de proveedores:
Creación: no podemos insertar la información de un proveedor, a menos que haya un pedido para él.
Supresión: se perderá la información sobre un proveedor al borrar un pedido que era el único que se le hacía a ese proveedor.
Modificación: para cambiar información sobre un proveedor, hay que recorrer todos los pedidos de ese proveedor. Hay redundancia.

**Tercera Forma Normal**

La 3FN es una extensión de la 2FN. La 2FN elimina las dependencias funcionales respecto a un subconjunto de la clave. La 3FN elimina la dependencia funcional entre atributos no llaves.

**Definición: Relación 3FN**

Una relación R está en 3FN si:

1. Está en 2FN.
2. Los atributos no llaves son independientes de cualquier otro atributo no llave primaria.

Es lo mismo que decir que se deben eliminar las dependencias transitivas de atributos no llaves respecto a la llave primaria, estando ya la relación en 2FN.

Definición: Dependencia transitiva

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

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

Analicemos las dependencias funcionales que existen en la relación PEDIDO:

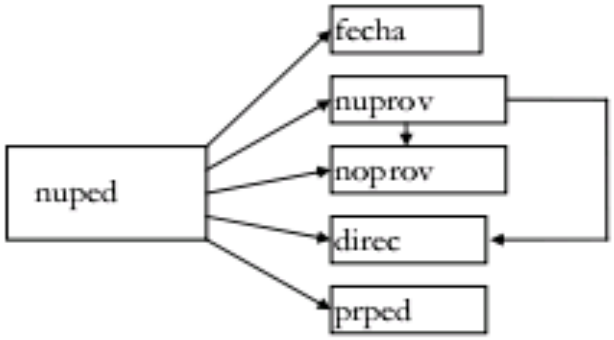



**Figura 18.** Relación 3FN.

La 3FN se hace:

1. Creando una relación para los atributos no llaves que no dependen transitivamente de la llave primaria (y los atributos que no se analizan por ser atributos llaves, pertenecientes a claves candidatas).

PEDIDO (nuped, fecha, nuprov, prped)

2. Creando una relación para los atributos no llaves que dependen transitivamente de la llave primaria a través de otro atributo o conjunto de atributos no llave primaria (que no son parte de la llave primaria.) La llave primaria de la relación así formada será el atributo o conjunto de atributos a través de los cuales existe la dependencia transitiva.

PROVEEDOR (numprov, noprov, direc)

Es necesario analizar las otras relaciones, en las cuales puede comprobarse que no hay dependencia entre atributos no llaves, por lo que están en 3FN.

Entonces el modelo de datos relacional en 3FN que representa el fenómeno de los pedidos de productos está formado por las siguientes relaciones:

PEDIDO (nuped, fecha, nuprov, prped)
PED-PROD (nuped, nuprod, cant, prprod)
PRODUCTO (nuprod, desc, prun)
PROVEEDOR (numprov, noprov, direc)

La 3FN ha eliminado los problemas asociados con la información sobre el proveedor en la 2FN. Veamos:

Creación: se puede insertar la información de un proveedor, aunque no haya un pedido para él.
Supresión: al borrar un pedido que era el único que se le hacía a un proveedor, no se perderá la información sobre el proveedor.
Modificación: la información sobre un proveedor está en una sola ocurrencia, por lo que, para cambiar cierta información de este, solo hay que hacerlo en dicha ocurrencia.
Ya en esta etapa se puede optimizar la 3FN. Las relaciones “degeneradas” que contengan solo la clave y que la información que aportan esté considerada en otra relación, por lo que se pueden eliminar. Puede que varias relaciones tengan la misma clave, por lo que se pueden combinar en una sola, siempre que el resultado sea lógico y tenga sentido.

Los analistas y diseñadores con experiencia producen relaciones en 3FN casi sin saber o preocuparse de esto y es que utilizan el sentido común y la experiencia para escribir relaciones normalizadas. Sin embargo, no siempre la intuición es suficiente y la metodología para normalizar las bases de datos se convierte en una herramienta imprescindible, que garantiza un diseño idóneo de los datos.

Aún en la 3FN existen problemas y habrán de ser resueltos con las siguientes formas normales.

Resumen de la Normalización

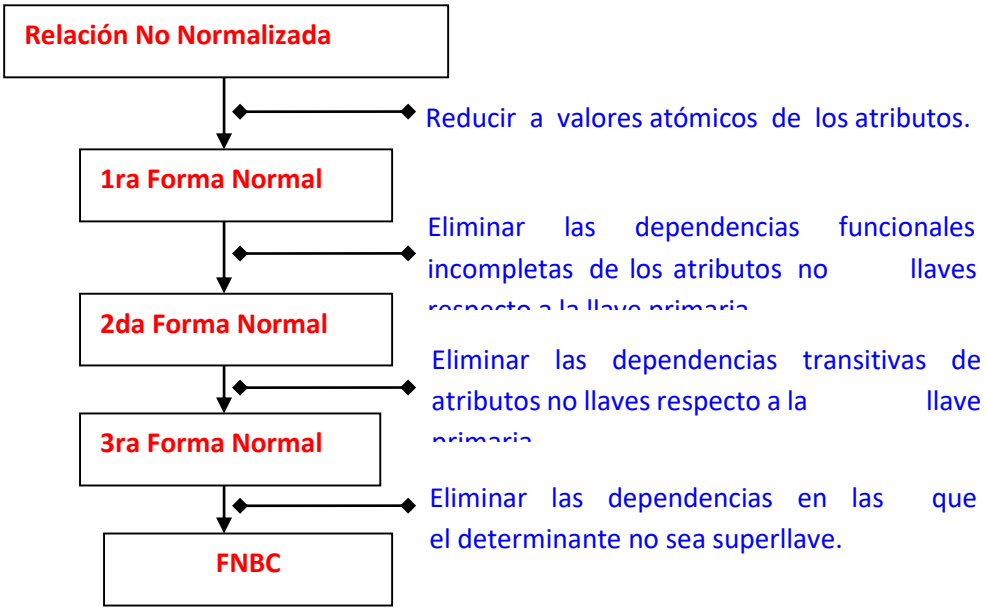



**Figura 19:** Resumen de la Normalización.

- Dejar de Estudio Independiente FNBC

Forma Normal de Boyce/Codd (FNBC)

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

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

Es decir, para una relación donde no tengan lugar las tres condiciones anteriores, la FNBC es idéntica a la 3FN. Esas tres condiciones son necesarias, pero no suficientes, para que la relación no esté en FNBC.

Definición de FNBC

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

Obsérvese que se habla en términos de llaves candidatas y no sólo de la llave primaria, ya que una llave es un caso especial de superllave y la llave puede ser candidata o primaria.

Además la definición de FNBC es conceptualmente más simple aunque es una FN más "fuerte".

Una relación que está en FNBC está también en 1FN, 2FN y 3FN.

Por ejemplo, la relación:

PRODUCTO (PNUM, PNOM, PRECIO, PESO)

con las DF PNUM--->PNOM, PRECIO, PESO.
PNOM--->PNUM, PRECIO, PESO.

Suponiendo que PNUM y PNOM sean llaves candidatas, está en FNBC, ya que en todas las DF que existen los determinantes son llaves candidatas y por tanto, superllaves de PRODUCTO.

Analicemos otro ejemplo:

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

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

O sea, EA--->P
P--->A
y no existe DF de P con respecto a A

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

**Por ejemplo:**

| EAP | E | A | P |
|---|---|---|---|
| | Pérez | Matem | prof. Blanco |
| | Pérez | Físic | Váldez |
| | Rdguez | Matem | Blanco |
| | Rdguez | Físic | Hdez |

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

Esta relación presenta anomalías de actualización, por ejemplo:

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

Estos problemas pueden eliminarse sustituyendo la relación original EAP por dos relaciones:

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

A sea, se separa la DF problemática (P--->A) en una relación independiente, donde el determinante ser la llave (P) y se forma otra relación donde dicho determinante P participara como atributo, pero en la cual no puede participar el atributo determinado en la dependencia funcional conflictiva (A).

Otro ejemplo: Sea la relación R1 (C, A, P) C- ciudad
A- calle
P- código postal

Donde CA--->P y P--->C, con llaves candidatas CD y PA.

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

La solución será dividir R1 en

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

**Otro ejemplo:**

Sea la relación EXAM (E, A, L) donde E- estudia A- asignatura L- lugar en el escalafón alcanzado por un estudiante en una asignatura y en la que se supone que 2 estudiantes no pueden alcanzar el mismo lugar en una asignatura.

**Entonces:**

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

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

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

**Ejemplo de normalización hasta la 3FN**

A continuación desarrollaremos, detalladamente, un ejemplo de aplicación de la normalización hasta la 3FN:

Se desea diseñar una BD para controlar la disponibilidad de materiales de construcción. De cada proveedor de materiales se conoce su código (cprov), que lo identifica, su nombre (nomprov) y el municipio en que radica (mun). De cada material se sabe su código (cmat), que lo identifica, su descripción (desc), la unidad de medida que se aplica al material (um) y el precio por unidad de medida (precio). Para guardar estos materiales hasta su posterior distribución existen diversos almacenes. De cada almacén se conoce su código (calm), que lo identifica, su dirección (diralm) y la capacidad de almacenaje (capac). Un proveedor puede suministrar varios materiales y un material puede ser suministrado por diferentes proveedores. Se sabe que un material suministrado por un proveedor está en un solo almacén y, además, se sabe qué cantidad de un material suministrado por un proveedor se encuentra en el almacén (cantmat). En un almacén se guardan distintos materiales y pueden existir varios almacenes donde se guarde un mismo material.

**1- Determinar las DF**

Veamos las dependencias funcionales que se derivan de esta situación:

Como cprov identifica al proveedor:
a) cprov →nomprov mun

Como cmat identifica al material:
b) cmat →desc um precio

Como calm identifica al almacén:
c) calm →diralm capac

1 2 3 4 5 6 7 8 9
d) cprov cmat → nomprov mun desc um precio calm diralm capac cantmat

Se incluyen en esta dependencia funcional los atributos:

1 y 2: por lo explicado para a).
3, 4 y 5: por lo explicado para b).
6: porque, dado un proveedor y un material, se conoce en qué almacén se guarda ese material suministrado por ese proveedor.
7 y 8: porque 6 se incluye y entonces vale lo explicado para c).
9: porque, dado un proveedor y un material, se conoce también en qué cantidad se guarda en el almacén correspondiente.

(Debe destacarse que no se acostumbra a agregar atributos tales como los señalados como 1, 2, 3, 4, 5, 7 y 8 en esta dependencia funcional d), pues se deduce del hecho de que existen las DF

a), b) y c) y no es preciso señalarlo explícitamente. En este ejemplo se agregan para dejar más claras las DF. En realidad, basta con señalar en esta DF lo siguiente:
d) cprov cmat → calm cantmat

**1 - Representar en una relación todos los atributos**

R (cprov, nomprov, mun, cmat, desc, um, precio, calm, diralm, capac, cantmat)

**3 - Determinar las llaves candidatas y seleccionar la primaria**

Para ello, debe analizarse cada dependencia funcional. Resulta conveniente empezar el análisis por aquellas en que se determinan más atributos en la parte derecha. Comenzaremos entonces por la DF d):

Como puede apreciarse, cprov cmat determina a todos los atributos de la relación. Los subconjuntos de este conjunto de atributos cprov cmat son cprov (por un lado) y cmat (por el otro). Ninguno de los subconjuntos determina, a su vez, a todos los atributos de la relación, como puede comprobarse en las DF a) y b) Por lo tanto, cprov cmat es una llave candidata de la relación R.

Es preciso seguir analizando las restantes DF. Veamos la DF a):

a) cprov →nomprov mun

Es necesario agregar atributos en la parte izquierda para lograr determinar a todos los atributos de la relación; deben escogerse aquellos que aparecen, a su vez, en la parte izquierda de las DF, pues son los que ayudan a determinar a los otros.

Probemos agregando el atributo cmat. Rápidamente nos daremos cuenta de que estamos en el caso de la DF d) que ya analizamos.

Probemos agregando el atributo calm:

cprov calm →nomprov mun diralm capac

Como se puede ver, no se tienen los atributos del material y sería preciso agregar el atributo cmat en el lado izquierdo:

cprov calm cmat→nomprov mun diralm capac desc um precio

y, aunque se podría decir que esos tres atributos determinan a todos los de la relación, pues se podría añadir cantmat en el lado derecho, ya que

cprov cmat→cantmat ocurre que cprov calm cmat es una superllave, pues contiene a cprov cmat, que ya sabemos que es una llave candidata.

Análisis similares a este hay que realizarlos con todas las DF, pero se van a obviar aquí. Basta decir que la única llave candidata que es posible hallar en este caso es cprov cmat y, por lo tanto, es la llave primaria:

R (cprov, nomprov, mun, cmat, desc, um, precio, calm, diralm, capac, cantmat)

**4 Aplicar la 2FN**

La relación R no está en 2FN, pues existen dependencias funcionales incompletas de atributos no llaves respecto a la llave primaria, según se aprecia en las dependencias funcionales a) y b), por lo que se crean las relaciones siguientes:

I. PROVEEDOR (cprov, nomprov, mun)
II. MATERIAL (cmat, desc, um, precio)

y la relación original queda de la siguiente forma:

R (cprov, cmat, calm, diralm, capac, cantmat)

**5 aplicar la 3FN**

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

cprov cmat → calm y
calm → diralm capac

por lo que se separa la siguiente relación:

III. ALMACÉN (calm, diralm, capac)

y la relación R anterior ahora queda así (le llamaremos SUMINISTRO, pues ya es posible asignarle un nombre adecuado):

SUMINISTRO (cprov, cmat, calm, cantmat)

Entonces, las cuatro tablas resultantes son:

PROVEEDOR (cprov, nomprov, mun)
MATERIAL (cmat, desc, um, precio)
ALMACÉN (calm, diralm, capac)
SUMINISTRO (cprov, cmat, calm, cantmat)

**6 analizar las relaciones obtenidas**

En este caso no hay relaciones que tengan igual llave ni relaciones que estén constituidas solo por la llave y cuya información se obtenga ya en otra relación, por lo que el modelo lógico representado por estas cuatro tablas es satisfactorio.
ESTUDIO INDEPENDIENTE

- Estudiar la Forma Formal Boyce Codd. Rosa María Matos. Tema 3
- EJERCICIO

Se desea controlar la actividad de una empresa de Proyectos. Para ello se cuenta con la siguiente información:

De cada trabajador:
- # de Carne de identidad.
- Nombre.
- Salario.

De cada proyecto:
- Código.
- Fecha de terminación.

Además se conoce las horas en plan de trabajo que cada trabajador dedica a cada proyecto.
Se sabe que un trabajador puede laborar en varios proyectos y en un proyecto participan varios trabajadores. Cada proyecto tiene una fecha de terminación y cada trabajador tiene un nombre y un salario, aunque un mismo salario o nombre puede serlo de varios trabajadores.

Determine las Dependencias Funcionales y represéntelas esquemáticamente.
Determine las llaves candidatas y la llave primaria.
Represente el modelo de datos en 1FN.

# CAPITULO 4: Diseño de Bases de Datos

Contenido

- Obtención del DER a partir de relaciones
- Obtención de relaciones a partir del DER.
- Ejemplo.

## 4.1 Metodología para el Diseño de la Base de Datos

**I. Determinar las entidades y los atributos**

- ✓ Para cada salida, consultando su formato en el DD y, quizás, hasta las miniespecificaciones, sustituir cada dato secundario por los primarios correspondientes.

- ✓ Tomar cada salida (con los valores sustituidos) y cada fichero normativo o de consulta como una entidad.

- ✓ Asignarle un nombre con contenido semántico a cada una y relacionar sus atributos.

**II. Normalizar las entidades**

- ✓ Aplicar la normalización a cada entidad.

- ✓ Analizar el resultado, fusionando aquellas entidades que tenga sentido hacerlo, eliminando las redundantes, etc.

**III. Determinar las relaciones entre las entidades (DER)**

- ✓ Confeccionar el DER siguiendo las siguientes ideas:

- ✓ Considerar, generalmente, como relación de m:m las entidades cuyas llaves sean combinación de llaves de otras entidades

- ✓ Analizar cada entidad con las restantes para determinar si existe relación y el tipo de ésta.

- ✓ Analizar la existencia de entidades agregadas, generalizadas, especializadas, débiles y sus relaciones con otras.

**IV. Obtener el modelo lógico global de los datos (según 1)**

**V. Obtener el diseño físico de la base de datos**

Hay que tener en cuenta:

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

### 4.1.1 Obtener el DER a partir de las relaciones:

Paciente (numHC, nombreP, edad, sexo, raza, grupoSang)
Cirujano (numId, nombreC, añosExp, gradoCient, codEsp)
Quirófano (idQ, piso, estado)
TipoIntervQ(tipoIntervQ, tiempo)
Operación (identIQ, diagnPreO, diagnOp, tipoIntervQ, numHC, fecha, susp)
Operación Realizada (identIQ, idQ, numId, horaInicio, horaFin, sitFinal, urgente)
OperaciónSuspendida (identIQ, causa)
Especialidad (codEsp, nombre)
Material (codMat, descripción, precioU)
Hospital (identIQ, codMat, cantidad)

- ✓ Si hay alguna entidad cuya llave es compuesta, de modo que resulta que esa llave es la combinación de las de otras entidades, es porque representa una relación entre ellas de m:n.
- ✓ Si hay alguna entidad cuya llave es compuesta y un subconjunto de los atributos que la forma no es llave de otra entidad, es porque es una entidad débil.
- ✓ Si en una entidad aparece, como atributo no llave, la llave de otra entidad es porque existe entre ellas una relación de m:1 (o, quizás, de 1:1).
- ✓ Analizar entidades agregadas, entidades generalizadas/especializadas y sus relaciones con otras.
- ✓ Ir confeccionando el DER.

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

El DER obtenido a partir de las relaciones anteriores es el siguiente:

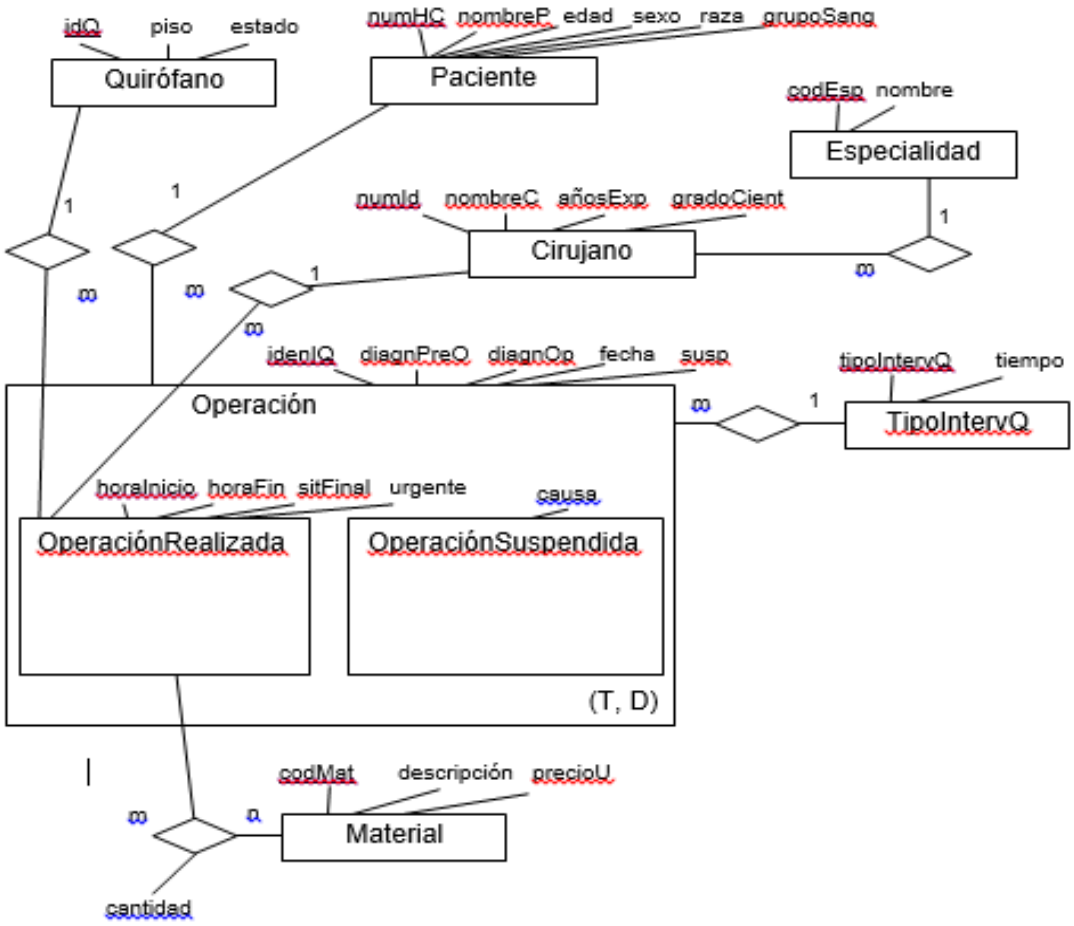



**Figura 20.** DER a Partir de las Relaciones.

## 5. Obtención el diseño lógico global de los datos

Aquí se deben seguir los pasos para llevar del DER al modelo lógico global de los datos.
Si en el paso anterior no se modifica el DER obtenido, entonces las tablas que se obtienen en este paso, en general, son las mismas que las obtenidas antes.

En nuestro caso, pudiera pensarse en agregar la distribución de especialidades por salones en un período de tiempo, por ejemplo, semanalmente.

6. Diseño físico de la BD
En este paso hay que tener en cuenta:

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

Ejemplos de estos dos aspectos a tener en cuenta:

- Considerar la inclusión de campos que sean secundarios debido a la cantidad de veces que habría que calcularlos en una determinada aplicación y cuyos datos primarios (aquellos a partir de los cuales los calculamos varían poco).

- Separar un archivo en varios debido a la cantidad de campos y lo que se accesa cada vez.
- Archivos temporales. En el ejemplo visto se pudiera pensar en tener un fichero con la planificación de las operaciones y que una vez terminado un día se elimina toda la información dado que la información queda registrada en las diferentes tablas que hay para almacenar lo relativo a estas.
- Archivos índices.
- Archivos para reportes.

CONSIDERACIONES FINALES:

- No se tienen en cuenta en las operaciones otros especialistas que no sean cirujanos, como por ejemplo, los anestesistas.
- No hay posibilidades de realizar operaciones en las que intervengan cirujanos de diferentes especialidades, tampoco más de uno de la misma.
- Solo puede existir una causa por la que se suspende una operación. Para hacerlo más general habría que convertir el atributo Causa en entidad.
- No se conoce la fecha en que cada salón no está funcionando y la causa.

CONCLUSIONES

- Recordar pasos para el diseño de la base de datos teniendo en cuenta no solo las salidas, sino también el conocimiento del problema.
- Diferencia entre modelo lógico y físico e implicaciones en el diseño de la base de datos.
- Posibilidades de obtener el DER a partir de las relaciones y viceversa.
- Creatividad además de los diferentes pasos a tener en cuenta.
- Diseños diferentes para una misma situación.

### 4.1.2 Obtención del modelo relacional a partir del DER

✓ Representar cada entidad regular en una tabla relacional.

✓ Representar en una tabla relacional cada entidad agregada con sus correspondientes atributos (entre ellos un identificador, sí fue definido), señalando la llave de la agregación (la llave de cada entidad que participa en la agregación si la relación es de m:m; la llave del extremo m si la relación es de m:1; una de las llaves si la relación es de 1:1).

✓ Representar cada entidad generalizada en una tabla que contendrá sus atributos (sólo los de la generalizada) y, entre ellos, la llave.

✓ Representar cada entidad especializada en una tabla que contendrá la llave de la generalización y los atributos propios sólo de la especialización.

✓ Representar en una tabla cada relación de m:m, incluyendo como llave de la relación, las llaves de las entidades que participan en ella, y los atributos de la relación si los hubiera.

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

- ✓ Representar cada entidad débil en una tabla que contendrá la llave de la entidad regular determinante y el identificador de la entidad débil (como llave) y el resto de sus atributos.

- ✓ Asegurarse de que están adecuadamente normalizadas; si no, aplicar el proceso de normalización.

Para nuestras explicaciones, utilizaremos un ejemplo que iremos desarrollando.

La situación a partir de la cual se discutir la metodología para diseñar la base de datos es la sgte.:

En un hospital Clínico Quirúrgico se desea automatizar el control de la actividad quirúrgica. En el hospital se tienen varios quirófanos y se realizan intervenciones quirúrgicas a los pacientes en diferentes especialidades.

- De manera general, cada día de la semana, un salón es destinado a las intervenciones quirúrgicas de una especialidad dada, dejándose uno para las urgencias.
- De cada quirófano se conoce su identificador, piso en que se encuentra, especialidades que lo utilizan y estado (funcionando o cerrado).
- De los pacientes, se conoce el número de historia clínica, el nombre, la edad, el sexo, la raza y el grupo sanguíneo.
- Las intervenciones quirúrgicas pueden ser electivas (planificadas) o urgentes (no planificadas). Cualquiera sea el tipo de intervención, se debe controlar el gasto de materiales en que se incurre, reportando, código del material, su descripción, cantidad de unidades utilizadas y costo unitario.
- En el caso de las planificadas se conoce el número de historia clínica del paciente, fecha, diagnóstico preoperatorio, intervención indicada, cirujano y especialidad, pudiendo planificarse varias operaciones para un mismo día.
- Un cirujano pertenece a una especialidad y una especialidad está conformada por varios cirujanos.
- Los cirujanos se caracterizan por su número de identidad, nombre, años de experiencia y grado científico.
- De la especialidad se conoce su identificador, nombre, cantidad de especialistas y salón de operaciones asignado.
- Se tiene la planificación de las operaciones quirúrgicas a realizar cada día. Esta incluye los mismos datos de las operaciones planificadas y además, el salón y el cirujano. Un paciente solo tiene una operación planificada en un día, sin embargo, aunque se evita, es posible realizar más de una operación no planificada a un paciente.
- Las operaciones planificadas pueden ser suspendidas por distintas causas.
- Para cada operación realizada (haya sido planificada o no), se confecciona un informe operatorio.

**Tabla 6.** Las salidas o reportes que se quieren obtener con el sistema automatizado son las siguientes.

**Salida 1:**

| PROGRAMACIÓN QUIRÚRGICA | | | | | Para el día: _____ de _____ de 200 __ | | | |
|---|---|---|---|---|---|---|---|---|
| Paciente | Hist clín. | Edad | Sexo | Diagnóstico pre-oper. | Intervenc. indicada | Hora | Salón | Cirujano |
| | | | | | | | | |

**Salida 2:**

| OPERACIONES SUSPENDIDAS | | | En el día: _____ de _____ de 200____ |
|---|---|---|---|
| Paciente | Historia clínica | Edad | Causa |
| | | | |

**Salida 3:**

| INFORME OPERATORIO | Fecha de operación: | |
|---|---|---|
| | Hora comienzo | Hora terminación |
| Paciente | Historia clínica | Edad |
| Diagnóstico clínico: _____ | | |
| Operación realizada: _____ | | |
| Diagnóstico operatorio: _____ | | |
| Situación final del paciente: ☐ Satisfactoria ☐ No satisfactoria ☐ Fallecido | | |
| Especialidad | Tipo de operación realizada ☐ Electiva ☐ Urgente | |
| Cirujano _____ | | |

**Salida 4:**

| RESUMEN DE ACTIVIDADES QUIRÚRGICAS | Mes: _____ |
|---|---|
| Cantidad de operaciones electivas: _____<br>Cantidad de operaciones urgentes: _____ | |
| Especialidad | Cantidad de operaciones |
| | |

**Salida 5:**

| GASTO DE MATERIALES DE OPERACIÓN | | | | | En el día: ____ de _____ de 200___ | | | |
|---|---|---|---|---|---|---|---|---|
| Paciente | Hist. clín. | Edad | Hora com. | Patología | Cód. mat. | Descr. mater. | Cantidad | Costo |
| | | | | | | | | |

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

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

1. Determinación de entidades, relaciones y atributos (información obtenida del problema)

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

Entidades:
Quirófano (identQ, piso, estado)
Intervención quirúrgica (planificada)
Especialidad(identEsp, cantEspecialistas)
Paciente (numHC, nombre, edad, sexo, raza, grupo sanguíneo)
Material (codMat, descripción, precio)
Intervención quirúrgica planificada

Cirujano (numId, nombre, años Exp, grado científico)
Tipo de operación
Intervención quirúrgica suspendida (fecha, causa)
Intervención quirúrgica realizada (fecha, horaInicio, horaFin, situaciónPaciente)
Relaciones:
Quirófano – Especialidad
Intervención quirúrgica – Material(cantMat)
Intervención quirúrgica – Paciente (diagnóstico PreOp, cirujano, especialidad)
Tipo de operación – Especialidad
Cirujano - Especialidad
Intervención quirúrgica – Cirujano
Intervención quirúrgica – Quirófano

2. Obtener listado de dependencias funcionales.

idQ → idQ, piso, estado
numHC → numHC, nombre, edad, sexo, raza, grupoSang
intervQPlanif → identIQ, idQ, fecha, horaInicio, numHC, codEsp, diagnPreO, diagnOp, tipoIntervQ, numId, horaFin, sitFinal
intervQNoPlanif → identIQ, idQ, fecha, horaInicio, numHC, codEsp, horaFin, diagnClínico, tipoIntervQ, sitFinal, numId
codMat → codMat, descripción, precioU
numId → numId, nombreC, añosExp, gradoCient, codEsp
codEsp → codEsp, nombreEsp, cantEsp
idQ, fecha, horaInicio → idQ, piso, estado, fecha, horaInicio, numHC, numId, identIQ
intervQSusp → identIQ, causa
identIQ, codMat → identIQ, codMat, cantidad
tipoIntervQ → tipoIntervQ, tiempo
identIQ → identIQ, tipoIntervQ

2a. Completar lista de dependencias funcionales a partir de las salidas

Son los atributos en rojo en las dependencias anteriores.

2b. Tratamiento para atributos secundarios o calculables

- El costo en la salida 5 se obtiene de multiplicar cantidad x precio
- Los atributos Diagnóstico pre-operatorio (salida 2), diagnóstico clínico (salida 3) y Patología (salida 5), son lo mismo. Se trata de diferentes alias.
- La hora de fin de una intervención planificada puede ser calculada a partir de conocer el tiempo promedio que demora una de su tipo, sin embargo, dado que este dato es el real, se mantiene. Sin embargo, esto indica que debe existir un atributo para cada tipo de operación que sea el tiempo promedio de esta.
- En la salida 4, todos los atributos son calculables a partir de los informes operatorios (salida3).
- La cantidad de especialistas que pertenecen a una especialidad es calculable a partir de contar para cada médico su especialidad.

2c. Proponer datos en ficheros nomencladores

En este caso aparece lo concerniente al tiempo promedio de cada tipo de operación y los materiales.

3. Normalizar las relaciones

3a. Obtener llaves candidatas.

- idQ, fecha, horaInicio, codMat
- identIQ, codMat
- fecha, horaInicio, numId, codMat
- fecha, horaInicio, numHC, codMat (en una situación realista, tener en cuenta que en el caso de las operaciones urgentes, no siempre se tiene historia clínica)

3b. Normalizar

1FN:

Hospital(idQ, piso, estado, numHC, nombreP, edad, sexo, raza, grupoSang, identIQ, fecha, horaInicio, codEsp, nombreE, diagnPreO, diagnOp, tipoIntervQ, numId, horaFin, sitFinal, codMat, descripción, precioU, nombreC, añosExp, gradoCient, causa, cantidad, tiempo, urgente)

2FN:

Operación(identIQ, piso, estado,nombreP, edad, sexo, raza, grupoSang, codEsp, nombreE, diagnPreO, diagnOp, tipoIntervQ, horaFin, sitFinal, nombreC, añosExp, gradoCient, causa, urgente, tiempo)

Material(codMat, descripción, precioU)

Hospital(identIQ, codMat, idQ, numHC, fecha, horaInicio, numId, cantidad)

3FN:

Operación(identIQ, piso, estado,nombreP, edad, sexo, raza, grupoSang, codEsp, diagnPreO, diagnOp, tipoIntervQ, horaFin, sitFinal, nombreC, añosExp, gradoCient, causa, urgente, tiempo)

Especialidad(codEsp, nombre)

Material(codMat, descripción, precioU)

Hospital(identIQ, codMat, idQ, numHC, fecha, horaInicio, numId, cantidad)

FNBC (Se cumplen las tres condiciones)

Determinantes

identIQ → numHC

identIQ → numId

identIQ → idQ

identIQ → fecha, horaInicio, tiempo

idQ, fecha, horaInicio → identIQ

⇓

Operación(identIQ, piso, estado,nombreP, edad, sexo, raza, grupoSang, codEsp, diagnPreO, diagnOp, tipoIntervQ, horaFin, sitFinal, nombreC, añosExp, gradoCient, causa, urgente, tiempo, numHC, numId, idQ, fecha, horaInicio)

⇓

Paciente (numHC, nombreP, edad, sexo, raza, grupoSang)

Cirujano (numId, nombreC, añosExp, gradoCient, codEsp)
Quirófano(idQ, piso, estado)
TipoIntervQ(tipoIntervQ, tiempo)
Operación(identIQ, diagnPreO, diagnOp, tipoIntervQ, horaFin, sitFinal, causa, numHC, numId, idQ, fecha, horaInicio, urgente)
Especialidad(codEsp, nombre)
Material(codMat, descripción, precioU)
Hospital(identIQ, codMat, cantidad)

REFINAMIENTO DE LAS RELACIONES
En la relación Operación aparece el atributo causa que solo es válido para las operaciones suspendidas, lo cual nos hace pensar en separar las operaciones realizadas de las suspendidas.

Tampoco tiene mucho sentido poner la hora de fin en las operaciones suspendidas, por lo que se deja en las realizadas. Un análisis similar ocurre con el gasto de materiales, la situación final del paciente y las relaciones con quirófano y cirujano.

Dentro de las operaciones realizadas, pudiera distinguirse entre las planificadas y las urgentes.

Como existen atributos comunes a ambos tipos, lo ideal es tener una entidad generalizada y dos especializadas.

Operación(identIQ, diagnPreO, diagnOp, tipoIntervQ, horaFin, sitFinal, causa, numHC, numId, idQ, fecha, horaInicio, tiempo)
Operación (identIQ, diagnPreO, diagnOp, tipoIntervQ, numHC, fecha, susp)
OperaciónRealizada(identIQ, idQ, numId, horaInicio, horaFin, sitFinal, urgente)
OperaciónSuspendida(identIQ, causa)

También se pudiera pensar en separar la entidad Paciente de manera de distinguir a los hospitalizados de los que no lo están.

### 4.1.3 Ejemplos de estos dos aspectos a tener en cuenta:

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

# CAPITULO 5: LENGUAJE SQL

Contenido

- Características del lenguaje.
- Las consultas simples y multitablas.

El SQL (Structured query language), lenguaje de consulta estructurado, es un lenguaje surgido de un proyecto de investigación de IBM para el acceso a bases de datos relacionales, a principios de los años 70. Evolucionó mucho desde aquellos tiempos y cambió su nombre desde SEQUEL hasta el SQL actual. Actualmente se ha convertido en un estándar de lenguaje de bases de datos, y la mayoría de los sistemas de bases de datos lo soportan, desde sistemas para ordenadores personales, hasta grandes ordenadores.

Por supuesto, a partir del estándar cada sistema ha desarrollado su propio SQL que puede variar de un sistema a otro, pero con cambios que no suponen ninguna complicación para alguien que conozca un SQL concreto.

Como su nombre indica, el SQL nos permite realizar consultas a la base de datos. Pero el nombre se queda corto ya que SQL además realiza funciones de definición, control y gestión de la base de datos. Las sentencias SQL se clasifican según su finalidad dando origen a tres sublenguajes:

- DDL (Data Description Language), lenguaje de definición de datos, incluye órdenes para definir, modificar o borrar las tablas en las que se almacenan los datos y de las relaciones entre estas. (Es el que más varia de un sistema a otro)
- DCL (Data Control Language), lenguaje de control de datos, contiene elementos útiles para trabajar en un entorno multiusuario, en el que es importante la protección de los datos, la seguridad de las tablas y el establecimiento de restricciones en el acceso, así como elementos para coordinar la compartición de datos por parte de usuarios concurrentes, asegurando que no interfieren unos con otros.
- DML (Data Manipulation Language), lenguaje de manipulación de datos, nos permite recuperar los datos almacenados en la base de datos y también incluye órdenes para permitir al usuario actualizar la base de datos añadiendo nuevos datos, suprimiendo datos antiguos o modificando datos previamente almacenados.

**Como componentes del DDL están los comandos:**

- CREATE (crear tablas)

CREATE TABLE tablabase (definición_de_columna [,definición_de_columna]…[, definición_de_clave_primaria] [, definición_de_clave_ajena [,definición_de_clave_ajena]]…)

- ALTER (modificar estructuras de tablas)

ALTER TABLE tablabase ADD columna tipodedato…

- DROP (eliminar tablas)

DROP TABLE tablabase

**Como componentes del DCL están los comandos:**

- GRANT…Otorgar permisos de acceso a la BD.
- REVOKE…Remover (quitar) privilegios de acceso.

**Como componentes del DML se encuentran los comandos encargados de realizar las consultas:**

- SELECT…Realizar consultas (recuperaciones
- INSERT…Insertar registros.
- UPDATE…Modificar registros.
- DELETE…Eliminar registros.

## 5.1 Características del lenguaje

Una sentencia SQL es como una frase (escrita en inglés) con la que decimos lo que queremos obtener y de donde obtenerlo.

Todas las sentencias empiezan con un verbo (palabra reservada que indica la acción a realizar), seguido del resto de cláusulas, algunas obligatorias y otras opcionales que completan la frase. Todas las sentencias siguen una sintaxis para que se puedan ejecutar correctamente.

## 5.2 Las Consultas Simples

Las consultas de selección se utilizan para indicar al motor de datos que devuelva información de las bases de datos, esta información es devuelta en forma de conjunto de registros.
El resultado de la consulta es una tabla lógica, porque no se guarda en el disco sino que está en memoria y cada vez que ejecutamos la consulta se vuelve a calcular.

La sintaxis básica de una consulta de selección es la siguiente:

```
SELECT campos FROM tabla;
```

En donde campos es la lista de campos que se deseen recuperar y tabla es el origen de los mismos, por ejemplo:

```
SELECT Nombre, Telefono FROM Clientes;
```

Esta consulta devuelve un subconjunto de la tabla Clientes con los campos nombre y teléfono.

**Ordenar los registros**

Adicionalmente se puede especificar el orden en que se desean recuperar los registros de las tablas mediante la cláusula del SELECT

```
ORDER BY Lista de Campos.
```

Lista de campos representa los campos a ordenar.

**Ejemplo:**

SELECT CodigoPostal, Nombre, Teléfono FROM Clientes ORDER BY Nombre;

Esta consulta devuelve los campos CodigoPostal, Nombre, Teléfono de la tabla Clientes ordenados por el campo Nombre.

Se pueden ordenar los registros por más de un campo, como por ejemplo:

SELECT CodigoPostal, Nombre, Telefono FROM Clientes ORDER BY CodigoPostal, Nombre;

Se puede especificar el orden de los registros: ascendente o descendente mediante la cláusula ASC o DESC

SELECT CodigoPostal, Nombre, Teléfono FROMClientes ORDER BY CodigoPostal DESC, Nombre ASC;

Consultas con Predicado

El predicado se incluye entre la cláusula y el primer nombre del campo a recuperar, los posibles predicados son:

**Tabla 7.** Consultas con Predicado.

| Predicado | Descripción |
|---|---|
| ALL | Devuelve todos los campos de la tabla |
| TOP | Devuelve un determinado número de registros de la tabla |
| DISTINCT | Omite los registros cuyos campos seleccionados coincidan totalmente |
| DISTINCTROW | Omite los registros duplicados basandose en la totalidad del registro y no sólo en los campos seleccionados. |

ALL Si no se incluye ninguno de los predicados se asume ALL. El Motor de base de datos selecciona todos los registros que cumplen las condiciones de la instrucción SQL. No se conveniente abusar de este predicado ya que obligamos al motor de la base de datos a analizar la estructura de la tabla para averiguar los campos que contiene, es mucho más rápido indicar el listado de campos deseados.

SELECT ALL FROM Empleados;
SELECT * FROM Empleados;

TOP Devuelve un cierto número de registros que entran entre al principio o al final de un rango especificado por una cláusula ORDER BY.
Ej Supongamos que queremos recuperar los nombres de los 25 primeros estudiantes del curso 2014:

```
SELECT TOP 25 Nombre, Apellido  FROM Estudiantes ORDER BY Nota DESC;
```

Si no se incluye la cláusula ORDER BY, la consulta devolverá un conjunto arbitrario de 25 registros de la tabla Estudiantes.
Se puede utilizar la palabra reservada PERCENT para devolver un cierto porcentaje de registros que caen al principio o al final de un rango especificado por la cláusula ORDER BY.

Supongamos que en lugar de los 25 primeros estudiantes deseamos el 10 por ciento del curso:

```
SELECT TOP 10 PERCENT Nombre, Apellido FROM Estudiantes ORDER BY Nota DESC;
```

DISTINCT Omite los registros que contienen datos duplicados en los campos seleccionados. Para que los valores de cada campo listado en la instrucción SELECT se incluyan en la consulta deben ser únicos.

Por ejemplo, varios empleados listados en la tabla Empleados pueden tener el mismo apellido. Si dos registros contienen López en el campo Apellido, la siguiente instrucción SQL devuelve un único registro:

```
SELECT DISTINCT Apellido FROM Empleados;
```

Con otras palabras el predicado DISTINCT devuelve aquellos registros cuyos campos indicados en la cláusula SELECT posean un contenido diferente.

DISTINCTROW Devuelve los registros diferentes de una tabla; a diferencia del predicado anterior que sólo se fijaba en el contenido de los campos seleccionados, éste lo hace en el contenido del registro completo independientemente de los campos indicados en la cláusula SELECT.

```
SELECT DISTINCTROW Apellido FROM Empleados
```

Si la tabla Empleados contiene dos registros: Antonio López y Marta López el ejemplo del predicado DISTINCT devuelve un único registro con el valor López en el campo Apellido ya que busca no duplicados en dicho campo. Este último ejemplo devuelve dos registros con el valor López en el apellido ya que se buscan no duplicados en el registro completo.

Alias

En determinadas circunstancias es necesario asignar un nombre a alguna columna determinada de un conjunto devuelto. Para resolver todas ellas tenemos la palabra reservada AS que se encarga de asignar el nombre que deseamos a la columna deseada.

Tomado como referencia el ejemplo anterior podemos hacer que la columna devuelta por la consulta, en lugar de llamarse apellido (igual que el campo devuelto) se llame Empleado.

```
SELECT DISTINCTROW Apellido AS Empleado  FROM Empleados;
```

Selección de filas o registros

Las consultas SQL que recuperan todas las filas de una tabla no son frecuentes en las aplicaciones. Generalmente se desea seleccionar parte de las filas de una tabla.
La cláusula WHERE se emplea para especificar las filas que se desean recuperar. Esta cláusula consta de la palabra clave WHERE seguida de una condición de búsqueda que especifica las filas a recuperar. Dicha condición en general es una expresión lógica

**Ejemplos:**

1. Mostrar todas las oficinas donde las ventas excedan al objetivo.

   SELECT ciudad, ventas, objetivo FROM oficinas WHERE ventas > objetivo

2. Muestra el nombre, las ventas y la cuota del empleado número 105.

   SELECT nombre, ventas, cuota FROM vendedores WHERE num_empl = 105

SQL recorre cada fila de la tabla, una a una, y aplica la condición. Se pueden producir tres casos:
- Si la condición de búsqueda es TRUE (cierta), la fila en cuestión se incluye en los resultados de la consulta.
- Si la condición de búsqueda es FALSE (falsa), la fila se excluye de los resultados de la consulta.
- Si la condición de búsqueda tiene un valor NULL (desconocido), la fila se excluye de los resultados de la consulta.

La condición de búsqueda actúa como un filtro para la tabla en cuestión.

Condiciones de selección

Las condiciones de selección son las condiciones que pueden aparecer en la cláusula WHERE.
Podemos tener cinco condiciones básicas:

1. Comparación
2. Rango
3. Pertenencia a un conjunto
4. Valor nulo
5. Correspondencia con un patrón.

Comparación: Compara el valor de una expresión con el valor de otra.

= igual que
<> Distinto de
< Menor que
<= menor o igual
> Mayor que
>= mayor o igual

SELECT numemp, nombre FROM Empleados WHERE ventas > cuota

Lista los empleados cuyas ventas superan su cuota

SELECT numemp, nombre FROM empleados WHERE contrato < #01/01/1988#

Lista los empleados contratados antes del año 88 (cuya fecha de contrato sea anterior al 1 de enero de 1988).

Rango (BETWEEN): Examina si el valor de la expresión está comprendido entre los dos valores definidos por exp1 y exp2.

SELECT numemp, nombre FROM Empleados WHERE ventas BETWEEN 100000 AND 500000

Lista los empleados cuyas ventas estén comprendidas entre 100.000 y 500.00

SELECT numemp, nombre FROM Empleados WHERE (ventas >= 100000) AND (ventas <= 500000)

Obtenemos lo mismo que en el ejemplo anterior. Los paréntesis son opcionales.

Pertenencia a conjunto (IN) Examina si el valor de la expresión es uno de los valores incluidos en la lista de valores.

SELECT numemp, nombre, oficina FROM Empleados WHERE oficina IN (12, 14,16)

Lista los empleados de las oficinas 12, 14 y 16

SELECT numemp, nombre FROM Empleados WHERE (oficina = 12) OR (oficina = 14) OR (oficina = 16)

Obtenemos lo mismo que en el ejemplo anterior. Los paréntesis son opcionales.

Valor nulo (IS NULL) Una condición de selección puede dar como resultado el valor verdadero TRUE, falso FALSE o nulo NULL.

Cuando una columna que interviene en una condición de selección contiene el valor nulo, el resultado de la condición no es verdadero ni falso, sino nulo, sea cual sea el test que se haya utilizado.
Por eso si queremos listar las filas que tienen valor en una determinada columna, no podemos utilizar el test de comparación, la condición oficina = null devuelve el valor nulo sea cual sea el valor contenido en oficina.

**Ejemplos:**

SELECT oficina, ciudad FROM Oficinas WHERE dir IS NULL

Lista las oficinas que no tienen director.

SELECT numemp, nombre FROM Empleados WHERE oficina IS NOT NULL

Lista los empleados asignados a alguna oficina (los que tienen un valor en la columna oficina).

Correspondencia con patrón (LIKE) Se utiliza cuando queremos utilizar caracteres comodines para formar el valor con el comparar.

Los comodines más usados son los siguientes:

? Representa un carácter cualquiera
* Representa cero o más caracteres
# Representa un dígito cualquiera (0-9)

**Ejemplos:**

SELECT numemp, nombre FROM Empleados WHERE nombre LIKE 'Luis*'

Lista los empleados cuyo nombre empiece por Luis (Luis seguido de cero o más caracteres).

SELECT numemp, nombre FROM Empleados WHERE nombre LIKE '*Luis*'

Lista los empleados cuyo nombre contiene Luis, en este caso también saldrían los empleados José Luis (cero o más caracteres seguidos de LUIS y seguido de cero o más caracteres).

SELECT numemp, nombre FROM Empleados WHERE nombre LIKE '??a*'

Lista los empleados cuyo nombre contenga una a como tercera letra (dos caracteres, la letra a, y cero o más caracteres.

**Condiciones de búsqueda compuestas (and, or y not)**

Las condiciones de búsqueda simples, descritas anteriormente devuelven un valor TRUE, FALSE o NULL cuando se aplican a una fila de datos. Utilizando las reglas de la lógica se pueden combinar estar condiciones de búsquedas simples para formar otras más complejas

**Ejemplo:**

Hallar los vendedores que están por debajo de la cuota o con ventas inferiores a 30000.

SELECT nombre, cuota, ventas FROM Vendedores WHERE (ventas < cuota) OR (ventas < 30000)

Los paréntesis en este caso son opcionales, solo se han puesto para claridad de la expresión.

**Agrupamiento de Registros**

GROUP BY Combina los registros con valores idénticos, en la lista de campos especificados, en un único registro. Para cada registro se crea un valor resumen si se incluye una función SQL agregada, como por ejemplo Sum o Count, en la instrucción SELECT.

SELECT campos FROM tabla WHERE criterio GROUP BY campos del grupo

GROUP BY es opcional. Los valores de resumen se omiten si no existe una función SQL agregada

en la instrucción SELECT.

```
SELECT  Id_Familia, Sum (Stock) FROM  Productos GROUP BY Id_Familia;
```

Una vez que GROUP BY ha combinado los registros, HAVING muestra cualquier registro agrupado por la cláusula GROUP BY que satisfaga las condiciones de la cláusula HAVING.

HAVING es similar a WHERE, determina qué registros se seleccionan.

```
SELECT  Id_Familia Sum(Stock) FROM  Productos GROUP BY Id_Familia HAVING
Sum(Stock) > 100 AND NombreProducto Like BOS*;
```

**AVG**

Calcula la media aritmética de un conjunto de valores contenidos en un campo especificado de una consulta.

```
Avg(expr)
```

expr representa el campo que contiene los datos numéricos para los que se desea calcular la media o una expresión que realiza un cálculo utilizando los datos de dicho campo.
La media calculada por Avg es la media aritmética (la suma de los valores dividido por el número de valores).

```
SELECT  Avg(Gastos) AS Promedio FROM Pedidos WHERE Gastos > 100;
```

**Count**

Calcula el número de registros devueltos por una consulta.

```
Count(expr)
```

expr contiene el nombre del campo que desea contar.
Los operandos de expr pueden incluir el nombre de un campo de una tabla, una constante o una función. Puede contar cualquier tipo de datos incluso texto.
Count simplemente cuenta el número de registros sin tener en cuenta qué valores se almacenan en los registros. La función  Count no cuenta los registros que tienen campos null a menos que expr sea el carácter comodín asterisco (*)

```
SELECT  Count(*) AS Total FROM Pedidos;
```

**Sum**

Devuelve la suma del conjunto de valores contenido en un campo específico de una consulta.

```
Sum(expr)
```

expr representa el nombre del campo que contiene los datos que desean sumarse o una expresión que realiza un cálculo utilizando los datos de dichos campos.
Los  operandos de expr pueden incluir el nombre de un campo de una tabla, una constante o una función

```
SELECT  Sum (PrecioUnidad * Cantidad) AS Total FROM  DetallePedido;
```

**Consultas Multitablas**

Las vinculaciones entre tablas se realizan mediante la cláusula INNER que combina registros de dos tablas siempre que haya concordancia de valores en un campo común.

SELECT campos FROM tb1 INNER JOIN tb2 ON tb1.campo1 comp tb2.campo2

tb1, tb2 Son los nombres de las tablas desde las que se combinan los registros.
campo1, campo2 Son los nombres de los campos que se combinan. Si no son numéricos, los campos deben ser del mismo tipo de datos y contener el mismo tipo de datos, pero no tienen que tener el mismo nombre.
Comp Es cualquier operador de comparación relacional: =, <, >, <=, >=, o <>.

El ejemplo siguiente muestra cómo podría combinar las tablas Categorías y Productos basándose en el campo IDCategoria:

SELECT Nombre_Categoría, NombreProducto FROM Categorias INNER JOIN Productos ON Categorias.IDCategoria = Productos.IDCategoria;

En el ejemplo anterior, IDCategoria es el campo combinado.

Se pueden enlazar varias cláusulas ON en una instrucción JOIN, utilizando la sintaxis siguiente:

SELECT campos FROM tabla1 INNER JOIN tabla2 ON tb1.campo1 comp tb2.campo1 AND ON tb1.campo2 comp tb2.campo2) OR ON tb1.campo3 comp tb2.campo3)];

También se pueden anidar instrucciones JOIN utilizando la siguiente sintaxis:

SELECT campos FROM tb1 INNER JOIN (tb2 INNER JOIN [( ]tb3 [INNER JOIN [( ] tablax [INNER JOIN ...)] ON tb3.campo3 comp tbx.campox)] ON tb2.campo2 comp tb3.campo3) ON tb1.campo1 comp tb2.campo2;

**ESTUDIO INDEPENDIENTE**

Considérese la base de datos de seguros:

Persona (id-conductor, nombre, dirección)
Coche (matrícula, año, modelo)
Accidente (número-informe, fecha, lugar)
es-dueño (id-conductor, matrícula)
Participó (id-conductor, coche, número-informe, importe-daños) donde las claves primarias se han subrayado.

Realizar las siguientes consultas SQL para esta base de datos relacional:

- Buscar el número total de las personas cuyos coches se han visto involucrados en un accidente en 1989.

- Buscar el número de accidentes en los cuales se ha visto involucrado un coche perteneciente a «Santos».
- Añadir un nuevo accidente a la base de datos; supóngase cualquier valor para los atributos necesarios.
- Borrar el Lance de «Santos».
- Actualizar el importe de daños del coche de matrícula «2002BCD» en el accidente con número de informe «AR2197» a $ 3000.00

## 5.3 Comandos para la modificación de la BD

Las operaciones de actualización modifican los datos almacenados en las tablas pero no su estructura, ni su definición.

Empezaremos por ver cómo insertar nuevas filas (con la sentencia INSERT INTO), veremos una variante (la sentencia SELECT... INTO), después veremos cómo borrar filas de una tabla (con la sentencia DELETE) y por último cómo modificar el contenido de las filas de una tabla (con la sentencia UPDATE). Si trabajamos en un entorno multiusuario, todas estas operaciones se podrán realizar siempre que tengamos los permisos correspondientes.

**Insertar una fila INSERT INTO...VALUES**

La inserción de nuevos datos en una tabla se realiza añadiendo filas enteras a la tabla, la sentencia SQL que lo permite es la orden INSERT INTO.

La inserción se puede realizar de una fila o de varias filas de golpe, veremos las dos opciones por separado y empezaremos por la inserción de una fila.

La sintaxis es la siguiente:

INSERT INTO Destino VALUES valores (,)

Esta sintaxis se utiliza para insertar una sola fila cuyos valores indicamos después de la palabra reservada VALUES.

- ✓ Los registros se agregan siempre al final de la tabla.
- ✓ Destino es el nombre de la tabla donde vamos a insertar la fila
- ✓ A continuación de la palabra VALUES, entre paréntesis se escriben los valores que queremos añadir. Estos valores se tienen que escribir de acuerdo al tipo de dato de la columna donde se van a insertar, la asignación de valores se realiza por posición, el primer valor lo asigna a la primera columna, el segundo valor a la segunda columna, así sucesivamente...
- ✓ Cuando no se indica ninguna lista de columnas después del destino, se asume por defecto todas las columnas de la tabla, en este caso, los valores se tienen que especificar en el mismo orden en que aparecen las columnas en la ventana de diseño de dicha tabla, y se tiene que utilizar el valor NULL para rellenar las columnas de las cuales no tenemos valores.

**Ejemplo:**

INSERT INTO empleados VALUES (200, 'Juan López', 30, NULL, 'rep ventas', #06/23/01#, NULL, 350000, 0)

El ejemplo anterior se podría escribir de la siguiente forma:

INSERT INTO empleados (numemp, oficina, nombre, titulo, cuota, contrato, ventas) VALUES (200, 30, 'Juan López', 'rep ventas',350000, #06/23/01#,0)

Observar que ahora hemos variado el orden de los valores y los nombres de columna no siguen el mismo orden que en la tabla origen, no importa, lo importante es poner los valores en el mismo orden que las columnas que enunciamos. Como no enunciamos las columnas oficina y director se rellenarán con el valor nulo (porque es el valor que tienen esas columnas como valor predeterminado).

**Insertar varias filas INSERT INTO...SELECT**

Podemos insertar en una tabla varias filas con una sola sentencia SELECT INTO si los valores a insertar se pueden obtener como resultado de una consulta, en este caso sustituimos la cláusula VALUES lista de valores por una sentencia SELECT como las que hemos visto hasta ahora. Cada fila resultado de la SELECT forma una lista de valores que son los que se insertan en una nueva fila de la tabla destino. Es como si tuviésemos una INSERT...VALUES por cada fila resultado de la sentencia SELECT.

**La sintaxis es la siguiente:**

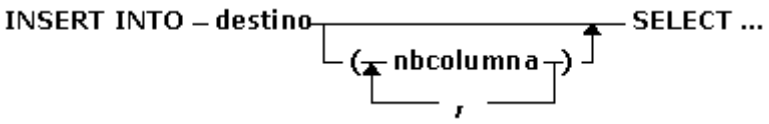



**Figura 21.** Insertar varias filas INSERT INTO...SELECT.

- El origen de SELECT puede ser el nombre de una consulta guardada, un nombre de tabla o una composición de varias tablas
- Las columnas de la SELECT no tienen por qué llamarse igual que en la tabla destino ya que el sistema sólo se fija en los valores devueltos por la SELECT.

**Ejemplo:** Supongamos que tenemos una tabla llamada repres con la misma estructura que la tabla empleados, y queremos insertar en esa tabla los empleados que tengan como título rep ventas.

INSERT INTO repres SELECT * FROM empleados WHERE título = 'rep ventas'

Con la SELECT obtenemos las filas correspondientes a los empleados con título rep ventas y las insertamos en la tabla repres. Como las tablas tienen la misma estructura no hace falta poner la lista de columnas y podemos emplear * en la lista de selección de la SELECT.

**Ejemplo:** Supongamos ahora que la tabla repres tuviese las siguientes columnas numemp,

oficinarep, nombrerep. En este caso no podríamos utilizar el asterisco, tendríamos que poner:

```
INSERT INTO repres SELECT numemp, oficina, nombre FROM empleados WHERE título
= 'rep ventas'
```

**O bien:**

```
INSERT INTO repres (numemp, oficinarep, nombrerep) SELECT numemp, oficina, nombre FROM
empleados WHERE título = 'rep ventas'
```

Borrar filas (DELETE)

**La sentencia DELETE elimina filas de una tabla.**

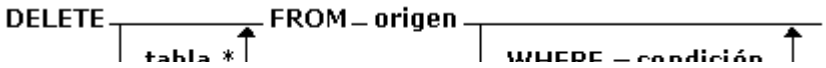



**Figura 22.** Sentencia DELETE elimina filas de una tabla.

- Origen es el nombre de la tabla de donde vamos a borrar, podemos indicar un nombre de tabla, incluir la cláusula IN si la tabla se encuentra en una base de datos externa, también podemos escribir una composición de tablas.
- La opción tabla.* se utiliza cuando el origen está basado en varias tablas, y sirve para indicar en qué tabla vamos a borrar.
- La opción * es opcional y es la que se asume por defecto y se puede poner únicamente cuando el origen es una sola tabla.
- La cláusula WHERE sirve para especificar qué filas queremos borrar. Se eliminaran de la tabla todas las filas que cumplan la condición. Si no se indica la cláusula WHERE, se borran TODAS las filas de la tabla.
- Una vez borrados, los registros no se pueden recuperar.
- Si la tabla donde borramos está relacionada con otras tablas se podrán borrar o no los registros siguiendo las reglas de integridad referencial definidas en las relaciones.

**Ejemplo:**

```
DELETE * FROM pedidos WHERE clie IN (SELECT numclie FROM clientes WHERE
nombre = 'Julian López');
```

**O bien:**

```
DELETE pedidos.* FROM pedidos INNER JOIN clientes ON pedidos.clie = clientes.numclie WHERE
nombre = 'Julián López';
```

Las dos sentencias borran los pedidos del cliente Julián López. En la segunda estamos obligados a poner pedidos.* porque el origen está basado en varias tablas.

```
DELETE * FROM pedidos; o DELETE FROM pedidos;
```

Borra todas las filas de pedidos.

Modificar el contenido de las filas ( UPDATE )

En determinadas situaciones puede ser deseable cambiar un valor dentro de un registro, sin cambiar todos los valores de la misma. Para este tipo de situaciones se utiliza la instrucción update. Al igual que ocurre con insert y delete, se pueden elegir las tuplas que van a ser actualizadas mediante una consulta.

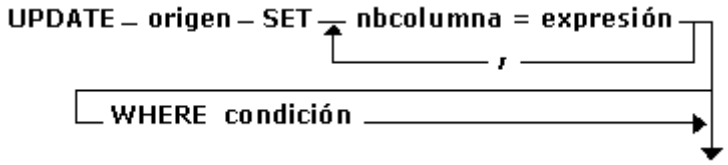



**Figura 23.** Modificar el contenido de las filas ( UPDATE ).

Origen puede ser un nombre de tabla, puede incluir la cláusula IN si la tabla a modificar se encuentra en una base de datos externa.

- La cláusula SET especifica qué columnas van a modificarse y qué valores asignar a esas columnas.
- nbcolumna, es el nombre de la columna a la cual queremos asignar un nuevo valor por lo tanto debe ser una columna de la tabla origen.

La expresión en cada asignación debe generar un valor del tipo de dato apropiado para la columna indicada. La expresión debe ser calculable a partir de los valores de la fila que se está actualizando.

**Ejemplo:**

UPDATE oficinas INNER JOIN empleados ON oficinas.oficina = empleados.oficina SET cuota=objetivo*0.01;

En este ejemplo queremos actualizar las cuotas de nuestros empleados de tal forma que la cuota de un empleado sea el 1% del objetivo de su oficina. La columna a actualizar es la cuota del empleado y el valor a asignar es el 1% del objetivo de la oficina del empleado, luego la cláusula SET será SET cuota = objetivo*0.01 o SET cuota = objetivo/100. El origen debe contener la cuota del empleado y el objetivo de su oficina, luego el origen será el INNER JOIN de empleados con oficinas.

**Ejemplo:** Queremos poner a cero las ventas de los empleados de la oficina 12

UPDATE empleados SET ventas = 0 WHERE oficina = 12;

**Ejemplo:** Queremos poner a cero el límite de crédito de los clientes asignados a empleados de la oficina 12.

UPDATE clientes SET limitecredito = 0 WHERE repclie IN (SELECT numemp FROM empleados

WHERE oficina = 12);

Cuando se ejecuta una sentencia UPDATE primero se genera el origen y se seleccionan las filas según la cláusula WHERE. A continuación se coge una fila de la selección y se le aplica la cláusula SET, se actualizan todas las columnas incluidas en la cláusula SET a la vez por lo que los nombres de columna pueden especificarse en cualquier orden. Después se coge la siguiente fila de la selección y se le aplica del mismo modo la cláusula SET, así sucesivamente con todas las filas de la selección.

**Ejemplo:**

UPDATE oficinas SET ventas=0, objetivo=ventas;

**O bien:**

UPDATE oficinas SET objetivo=ventas, ventas=0;

Los dos ejemplos anteriores son equivalentes ya que el valor de ventas que se asigna a objetivo es el valor antes de la actualización, se deja como objetivo las ventas que ha tenido la oficina hasta el momento y se pone a cero la columna ventas.

Hasta ahora hemos estudiado las sentencias que forman parte del DML (Data Management Language) lenguaje de manipulación de datos, todas esas sentencias sirven para recuperar, insertar, borrar, modificar los datos almacenados en la base de datos; lo que veremos a continuación son las sentencias que afectan a la estructura de los datos.

El DDL (Data Definition Language) lenguaje de definición de datos es la parte del SQL que más varía de un sistema a otro ya que esa área tiene que ver con cómo se organizan internamente los datos y eso, cada sistema lo hace de una manera u otra.

**CREATE TABLE**

La sentencia CREATE TABLE sirve para crear la estructura de una tabla no para rellenarla con datos, nos permite definir las columnas que tiene y ciertas restricciones que deben cumplir esas columnas.

La sintaxis es la siguiente:

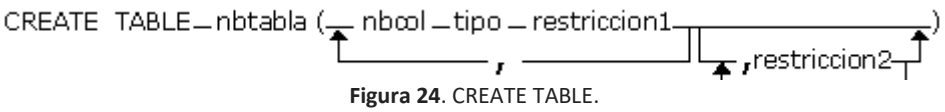



**Figura 24**. CREATE TABLE.

nbtabla: nombre de la tabla que estamos definiendo
nbcol: nombre de la columna que estamos definiendo
tipo: tipo de dato de la columna, todos los datos almacenados en la columna deberán ser de ese tipo.

Para escribir una sentencia CREATE TABLE se empieza por indicar el nombre de la tabla que queremos crear y a continuación entre paréntesis indicamos separadas por comas las definiciones de cada columna de la tabla, la definición de una columna consta de su nombre, el tipo de dato que tiene y podemos añadir si queremos una serie de especificaciones que deberán cumplir los datos almacenados en la columna, después de definir cada una de las columnas que compone la tabla se pueden añadir una serie de restricciones, esas restricciones son las mismas que se pueden indicar para cada columna pero ahora pueden afectar a más de una columna por eso tienen una sintaxis ligeramente diferente.

Una restricción consiste en la definición de una característica adicional que tiene una columna o una combinación de columnas, suelen ser características como valores no nulos (campo requerido), definición de índice sin duplicados, definición de clave principal y definición de clave foránea (clave ajena o externa, campo que sirve para relacionar dos tablas entre sí).

Una restricción de tipo 1 se utiliza para indicar una característica de la columna que estamos definiendo, tiene la siguiente sintaxis:

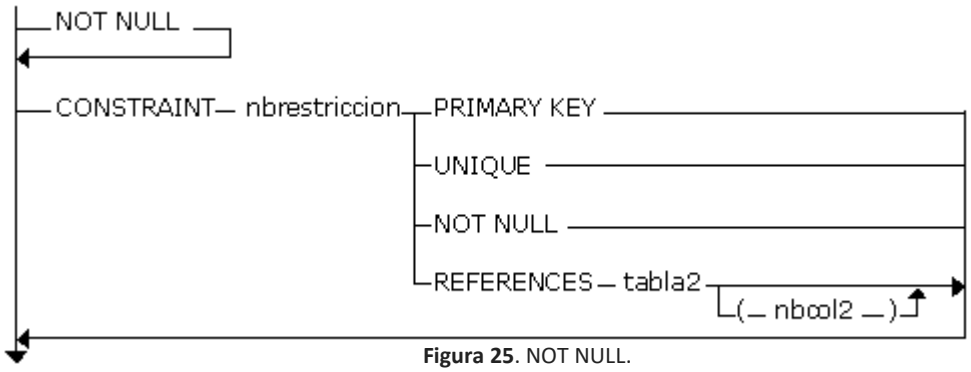



**Figura 25**. NOT NULL.

- La cláusula NOT NULL indica que la columna no podrá contener un valor nulo, es decir que se deberá rellenar obligatoriamente y con un valor válido (equivale a la propiedad requerido Sí de las propiedades del campo).
- La cláusula CONSTRAINT sirve para definir una restricción que se podrá eliminar cuando queramos sin tener que borrar la columna. A cada restricción se le asigna un nombre que se utiliza para identificarla y para poder eliminarla cuando se quiera.
- Como restricciones tenemos la de clave primaria (clave principal), la de índice único (sin duplicados), la de valor no nulo, y la de clave foránea.
- La cláusula PRIMARY KEY se utiliza para definir la columna como clave principal de la tabla. Esto supone que la columna no puede contener valores nulos ni pueden haber valores duplicados en esa columna, es decir que dos filas no pueden tener el mismo valor en esa columna.
- En una tabla no puede haber varias claves principales, por lo que no podemos incluir la cláusula PRIMARY KEY más de una vez, en caso contrario la sentencia da un error. No hay que confundir la definición de varias claves principales con la definición de una clave principal compuesta por varias columnas, esto último sí está permitido y se define con una restricción de tipo 2.

- La cláusula UNIQUE sirve para definir un índice único sobre la columna. Un índice único es un índice que no permite valores duplicados, es decir que si una columna tiene definida un restricción de UNIQUE no podrán haber dos filas con el mismo valor en esa columna. Se suele emplear para que el sistema compruebe el mismo que no se añaden valores que ya existen, por ejemplo si en una tabla de clientes queremos asegurarnos que dos clientes no puedan tener el mismo D.N.I. y la tabla tiene como clave principal un código de cliente, definiremos la columna dni con la restricción de UNIQUE.
- La cláusula NOT NULL indica que la columna no puede contener valores nulos, cuando queremos indicar que una columna no puede contener el valor nulo lo podemos hacer sin poner la cláusula CONSTRAINT, o utilizando una cláusula CONSTRAINT.
- La última restricción que podemos definir sobre una columna es la de clave foránea, una clave foránea es una columna o conjunto de columnas que contiene un valor que hace referencia a una fila de otra tabla, en una restricción de tipo 1 se puede definir con la cláusula REFERENCES, después de la palabra reservada indicamos a qué tabla hace referencia, opcionalmente podemos indicar entre paréntesis el nombre de la columna donde tiene que buscar el valor de referencia, por defecto coge la clave principal de la tabla2, si el valor que tiene que buscar se encuentra en otra columna de tabla2, entonces debemos indicar el nombre de esta columna entre paréntesis, además sólo podemos utilizar una columna que esté definida con una restricción de UNIQUE, si la columna2 que indicamos no está definida sin duplicados, la sentencia CREATE nos dará un error.

**Ejemplo:**

```
CREATE TABLE tab1 (col1 INTEGER CONSTRAINT pk PRIMARY KEY,
col2 CHAR (25) NOT NULL, col3 CHAR (10) CONSTRAINT uni1 UNIQUE,
col4 INTEGER, col5 INT CONSTRAINT fk5 REFERENCES tab2 );
```

Con este ejemplo estamos creando la tabla tab1 compuesta por: una columna llamada col1 de tipo entero definida como clave principal, una columna col2 que puede almacenar hasta 25 caracteres alfanuméricos y no puede contener valores nulos, una columna col3 de hasta 10 caracteres que no podrá contener valores repetidos, una columna col4 de tipo entero sin ninguna restricción, y una columna col5 de tipo entero clave foránea que hace referencia a valores de la clave principal de la tabla tab2.

ALTER TABLE

La sentencia ALTER TABLE sirve para modificar la estructura de una tabla que ya existe. Mediante esta instrucción podemos añadir columnas nuevas, eliminar columnas. Ten cuenta que cuando eliminamos una columna se pierden todos los datos almacenados en ella.

También nos permite crear nuevas restricciones o borrar algunas existentes. La sintaxis puede parecer algo complicada pero sabiendo el significado de las palabras reservadas la sentencia se aclara bastante; ADD (añade), ALTER (modifica), DROP (elimina), COLUMN (columna),

CONSTRAINT (restricción).

La sintaxis es la siguiente:

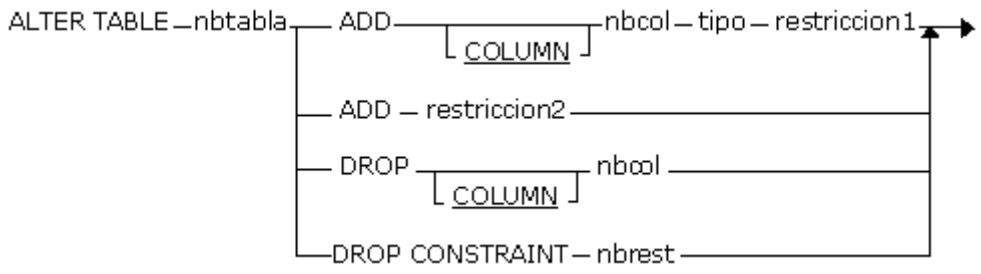



**Figura 26.** ALTER.

La sintaxis de restriccion1 es idéntica a la restricción1 de la sentencia CREATE TABLE, te la describimos a continuación, si tienes alguna duda repasa la sentencia CREATE TABLE.

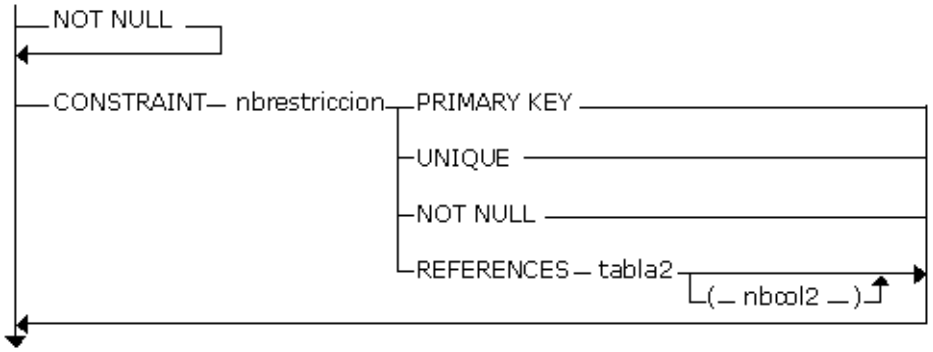



**Figura 27. Sintaxis de restriccion1.**

La cláusula ADD COLUMN (la palabra COLUMN es opcional) permite añadir una columna nueva a la tabla. Como en la creación de tabla, hay que definir la columna indicando su nombre, tipo de datos que puede contener, y si lo queremos alguna restricción de valor no nulo, clave primario, clave foráneo, e índice único, restriccion1 es opcional e indica una restricción de tipo 1 que afecta a la columna que estamos definiendo.

**Ejemplo:**

ALTER TABLE tab1 ADD COLUMN col3 integer NOT NULL CONSTRAINT c1 UNIQUE

Con este ejemplo estamos añadiendo a la tabla tab1 una columna llamada col3 de tipo entero, requerida (no admite nulos) y con un índice sin duplicados llamado c1.

Cuando añadimos una columna lo mínimo que se puede poner sería:

ALTER TABLE tab1 ADD col3 integer

En este caso la nueva columna admite valores nulos y duplicados.

- ✓ Para añadir una nueva restricción en la tabla podemos utilizar la cláusula ADD restriccion2 (ADD CONSTRAINT...).

**Ejemplo:**

ALTER TABLE tab1 ADD CONSTRAINT c1 UNIQUE (col3)
Con este ejemplo estamos añadiendo a la tabla tab1 un índice único (sin duplicados) llamado c1 sobre la columna col3.

- ✓ Para borrar una columna basta con utilizar la cláusula DROP COLUMN (COLUMN es opcional) y el nombre de la columna que queremos borrar, se perderán todos los datos almacenados en la columna.

**Ejemplo:**

ALTER TABLE tab1 DROP COLUMN col3

También podemos escribir:

ALTER TABLE tab1 DROP col3

El resultado es el mismo, la columna col3 desaparece de la tabla tab1.

- ✓ Para borrar una restricción basta con utilizar la cláusula DROP CONSTRAINT y el nombre de la restricción que queremos borrar, en este caso sólo se elimina la definición de la restricción pero los datos almacenados no se modifican ni se pierden.

**Ejemplo:**

ALTER TABLE tab1 DROP CONSTRAINT c1

Con esta sentencia borramos el índice c1 creado anteriormente pero los datos de la columna col3 no se ven afectados por el cambio.

**DROP TABLE**

La sentencia DROP TABLE sirve para eliminar una tabla. No se puede eliminar una tabla si está abierta, tampoco la podemos eliminar si el borrado infringe las reglas de integridad referencial (si interviene como tabla padre en una relación y tiene registros relacionados).

**La sintaxis es la siguiente:**

DROP TABLE _ nbtabla

**Ejemplo:**

DROP TABLE tab1

Elimina de la base de datos la tabla tab1.

CREATE INDEX

La sentencia CREATE INDEX sirve para crear un índice sobre una o varias columnas de una tabla.

La sintaxis es la siguiente:

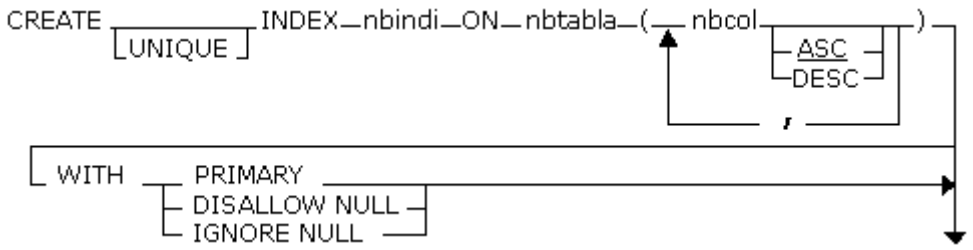



**Figura 28.** Sintaxis CREATE INDEX.

**nbindi**: nombre del índice que estamos definiendo. En una tabla no pueden haber dos índices con el mismo nombre de lo contrario da error.

**nbtabla**: nombre de la tabla donde definimos el índice. A continuación entre paréntesis se indica la composición del índice (las columnas que lo forman).

**nbcol**: nombre de la columna que indexamos. Después del nombre de la columna podemos indicar cómo queremos que se ordenen las filas según el índice mediante las cláusulas ASC/DESC.

**ASC:** la cláusula ASC es la que se asume por defecto e indica que el orden elegido para el índice es ascendente (en orden alfabético si la columna es de tipo texto, de menor a mayor si es de tipo numérico, en orden cronológico si es de tipo fecha).

**DESC:** indica orden descendente, es decir el orden inverso al ascendente.

Podemos formar un índice basado en varias columnas, en este caso después de indicar la primera columna con su orden, se escribe una coma y la segunda columna también con su orden, así sucesivamente hasta indicar todas las columnas que forman el índice.

**Ejemplo:**

CREATE UNIQUE INDEX ind1 ON clientes (provincia, población ASC, fecha nacimiento DESC)

Crea un índice llamado ind1 sobre la tabla cliente formado por las columnas provincia, población y fecha nacimiento. Este índice permite tener ordenadas las filas de la tabla clientes de forma que aparezcan los clientes ordenados por provincia, dentro de la misma provincia por población y dentro de la misma población por edad y del más joven al más mayor.

Al añadir la cláusula UNIQUE el índice no permitirá duplicados por lo que no podría tener dos clientes con la misma fecha de nacimiento en la misma población y misma provincia, para evitar el problema sería mejor utilizar:

CREATE INDEX ind1 ON clientes (provincia, población ASC, fecha nacimiento DESC)

**DROP INDEX**

La sentencia DROP INDEX sirve para eliminar un índice de una tabla. Se elimina el índice pero no las columnas que lo forman.

**La sintaxis es la siguiente:**

DROP INDEX _ nbíndice _ ON _ nbtabla

Ejemplo:

DROP INDEX ind1 ON clientes

Elimina el índice que habíamos creado en el ejemplo anterior.

## 5.4 Conclusiones

El uso de Consultas de Referencias Cruzadas nos permite agilizar la actualización de datos de una Base de Datos. La rapidez es una de las características esenciales de este tipo de consulta.

ESTUDIO INDEPENDIENTE

1- **Se tiene la base de datos de empleados**

Empleado (nombre-empleado, calle, ciudad)
Trabaja (nombre-empleado, nombre-empresa, sueldo)
Empresa (nombre-empresa, ciudad)
Jefe(nombre-empleado, nombre-jefe)

Donde las claves primarias se han subrayado. Plantee una expresión SQL para cada una de las consultas siguientes:

- Buscar los nombres de todos los empleados que trabajan en el Banco BPA.
- Buscar los nombres y ciudades de residencia de todos los empleados que trabajan en el Banco BPA.
- Buscar los nombres, direcciones y ciudades de residencia de todos los empleados que trabajan en el Banco BPA y que ganan más de $ 500.00.
- Buscar todos los empleados que viven en la ciudad de la empresa para la que trabajan.
- Buscar todos los empleados que viven en la misma ciudad y en la misma calle que sus jefes.
- Buscar todos los empleados que no trabajan en el Banco BPA.
- Buscar todos los empleados que ganan más que cualquier empleado del Banco BANDEC.
- Supóngase que las empresas pueden tener sede en varias ciudades. Buscar todas las empresas con sede en todas las ciudades en las que tiene sede el Banco BANDEC.

- Buscar todos los empleados que ganan más que el sueldo medio de los empleados de su empresa.
- Buscar la empresa que tiene el mayor número de empleados.
- Buscar la empresa que tiene el menor sueldo medio.
- Buscar aquellas empresas cuyos empleados ganan un sueldo más alto, en media, que el sueldo medio del Banco BPA.

2. **De la misma base de datos anterior, plantee una expresión en SQL para cada una de las siguientes consultas:**
   - Modificar la base de datos de forma que Santos viva en Holguín.
   - Incrementar en un 10% el sueldo de todos los empleados del Banco BPA.
   - Incrementar en un 10% el sueldo de todos los jefes del Banco BPA.
   - Incrementar en un 10% el sueldo de todos los empleados del Banco BPA, a menos que su sueldo pase a ser mayor de $ 1500.00, en cuyo caso se incrementará su sueldo sólo en un 3%.
   - Borrar todos los registros de la relación trabaja correspondientes a los empleados del Banco BPA.

# BIBLIOGRAFÍA


Abad, R., Medina, M., Careaga, A. (1993). Fundamentos de las estructuras de datos relacionales. Grupo Noriega Editores, Limusa. México.

A. Silberschatz, H. F. Korth y S. Sudarshan, Fundamentos de Bases de Datos, España: MacGraw Hill, 2002.

Camuña, J., (2014). Lenguaje de definición y modificación de datos SQL [Versión electrónica]. (1ra. ed.). IC Editorial, Málaga.

C. J. Date, Introducción a los Sistemas de Bases de Datos, México: Pearson Educación, 2001

Connolly, T., Begg, C. (2005). Sistemas de bases de datos. Un enfoque práctico para diseño, implementación y gestión (4ta. ed.). Pearson, Madrid.

De Miguel, A., Martínez, P., Castro, E., Cavero, J., Cuadra, D., Iglesias, A., Nieto, C. (2005). Diseño de bases de datos, Problemas resueltos. Alfaomega Ra-Ma. México.

Elmasri, R., Navathe, S. (2007). Fundamentos de sistemas de bases de datos (5ta. Ed.). Pearson. Madrid.

Groff, J., Weinberg, P., (2003). SQL manual de referencia. McGraw-Hill. Madrid.

Ibañez, l. (2014). Gestión de bases de datos (2da. ed.). Ra-Ma. Madrid.

Jiménez, M. (2014). Bases de datos relacionales y modelado de datos, [Versión electrónica]. IC Editorial.

Koutchouk, M. (1992). SQL et DB2 le relationnel et sa practique (2da. ed.). Masson. Paris.

M. V. N. Cabello, Introducción a la Base de Datos Relacionales, Madrid: Visión Libros, 2010.

M. Y. Jimenez Capel, IFCT0310: Bases de datos relacionales y modelado de datos., Malaga: IC Editorial, 2015.

Piattini, M., Clavo, J., Cervera, J., Fernández, L., (1996) Análisis y diseño detallado de aplicaciones informáticas de gestión. Ra-Ma, Madrid.

Silberschatz, A., Korth, H., Sudarshan, S. (2014). Fundamentos de bases de datos (6ta. ed.). McGraw-Hill. Madrid.

V. F. Alarcón, Desarrollo de sistemas de información: una metodología basada en el modelado., Catalunya: Edicions UPC, 2010.

Ingeniería y Tecnología