



## Introducción a las Bases de Datos

### *Contenido*

<b>Introducción a Bases de Datos .....</b>	<b>2</b>
Diferencia entre Datos e Información.....	2
¿Qué es una Base de Datos?.....	2
<b>Organización de una Base de Datos.....</b>	<b>4</b>
Tipos de Bases de Datos.....	5
<b>Bases de Datos Relacional.....</b>	<b>8</b>
Ventajas de una Base de Datos Relacional .....	8
Tablas, Filas y Columnas .....	9
Modelo de Datos .....	9
<b>Modelo Entidad-Relación .....</b>	<b>10</b>
Cardinalidad de las relaciones.....	11
Símbolos y notaciones de Diagrama ER.....	12
Caso de aplicación de un DER. ....	12
Elementos adicionales de un DER.....	13
Cómo hacer un Diagrama Entidad-Relación o DER.....	15
Tipos de Datos .....	16
Índice.....	17
Clave Primaria.....	18
Clave Foránea (FK) .....	18
<b>Bases de Datos SQL.....</b>	<b>18</b>
¿Qué es SQL?.....	19
Comandos en SQL .....	19
Crear una Base de Datos .....	19
Crear una tabla.....	19
Crear Clave Primaria (PK) .....	20
Crear Clave Foránea (FK) .....	20
Selección de datos .....	21
Insertar datos.....	21
Actualizando registros .....	22
Borrar datos.....	22
Operadores .....	23
Ordenación de datos .....	23
Funciones de Agregación .....	24
Subconsultas.....	24

## Introducción a Bases de Datos

Las bases de datos surgen de la necesidad de las empresas en almacenar grandes cantidades de información de una forma rápida, sencilla y fiable, y que a su vez pudieran acceder a ella en cualquier momento sin necesidad de desplazarse a salas dedicadas a archivar documentación, como se hacía antiguamente cuando la tecnología de la información empezaba a emerger.

Cuando comenzó el despegue de los programas informáticos se empezaron a almacenar datos en los archivos de los programas, lo cual era más cómodo, pero aun así tenían grandes dificultades a la hora de querer modificar registros, estructuras o simplemente buscar información. Eran métodos lentos, propensos a errores de archivo e incómodos de implementar en grandes estructuras. Recordemos que estamos hablando de épocas donde no existía internet.

A finales de los años sesenta nacen las bases de datos. En estos sistemas se guardan los datos utilizados por los usuarios. Y los programas que consumían esos datos no se tenían que preocupar de su mantenimiento ni almacenaje, por lo que un cambio en la base de datos no tiene por qué afectar en principio a los programas que la utilizan.

## Diferencia entre Datos e Información

Todo sistema informático tiene como objetivo principal administrar y controlar los datos de manera eficiente para la toma de decisiones, por ello, dependiendo de lo que consideremos como dato deberíamos tenerlo en cuenta para nuestro sistema. Esto nos lleva a definir a los datos como todo aquello que se puede registrar en nuestra base de datos, que siendo procesado nos devolverá información relevante.

Podemos decir que, si nosotros tenemos un dato de temperatura, (por ejemplo 32º C), no nos dice nada por sí sólo, pero si obtenemos ese dato luego de consultar la temperatura de una ciudad determinada en determinado momento se convierte en información para conocer el clima de esa ciudad, a ese procesamiento de datos le llamamos información.

Un dato puede ser una letra, número, símbolo o palabra que por sí solo no tiene ningún significado, mientras que la información se define como un conjunto de datos procesados que tienen significado para el usuario.



## ¿Qué es una Base de Datos?

Como definición de base de datos entendemos que se trata de un “conjunto de datos interrelacionados, almacenados sin redundancias innecesarias y que tienen un significado implícito, los cuales sirven a las aplicaciones sin estar relacionados de una manera directa

entre ellos.” Por otro lado, cuando hablamos de los datos, queremos decir: hechos conocidos que pueden registrarse y que tienen un significado implícito.

Cada día nos encontramos con actividades que requieren algún tipo de interacción con una base de datos (ingreso en un banco, reserva de una entrada para el teatro, solicitud de una suscripción a una revista, compra de productos, ...). Estas interacciones son ejemplos de lo que se llama aplicaciones tradicionales de bases de datos que básicamente almacenan información numérica o de texto, aunque los avances tecnológicos han permitido que también existan: bases de datos multimedia, sistemas de información geográfica (GIS), sistemas de proceso analítico on-line. Y ni hablar si tocamos el tema Big Data.

Hay dos grandes categorías o grupos de datos:

1. **Los datos de usuarios:** Datos usados por las aplicaciones
2. **Los datos de sistema:** Datos que la base de datos utiliza para su propia gestión como los usuarios registrados para operar la base, los privilegios de estos usuarios, configuraciones, etc.

En otras palabras, una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un sistema de gestión de bases de datos (SGBD o DBMS por sus siglas en inglés). En conjunto, los datos y el SGBD, junto con las aplicaciones asociadas a ellos, reciben el nombre de sistema de bases de datos, abreviado normalmente a simplemente base de datos.

#### Sistemas gestores de bases de datos

Un sistema gestor de base de datos (SGBD), es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos. Los usuarios acceden a la información con herramientas específicas de consulta y de generación de informes. Los SGBD, proporcionan métodos para la integridad de los datos, administrar el acceso a usuarios y recuperar la información en caso de un fallo.

#### ¿Cuáles son los componentes de un SGBD?

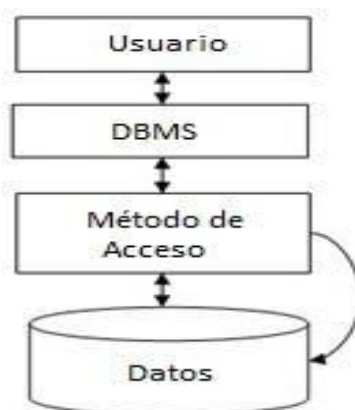
Un sistema de gestión de base de datos consta de varios componentes, todos ellos contribuyen al buen funcionamiento del software. Los elementos básicos que lo conforman son tres: el diccionario de datos, el lenguaje de definición de datos y el lenguaje de manipulación de datos.

- **Diccionario de datos:** consiste en una lista de metadatos que reflejan las características de los diversos tipos de datos incluidos en la base de datos. Además, estos metadatos informan sobre los permisos de uso de cada registro y su representación física. De esta manera, el diccionario proporciona toda la información relevante sobre los datos almacenados.
- **Lenguaje de definición de datos:** el lenguaje de definición de datos, también llamado lenguaje de base de datos o DDL (Data Definition Language), sirve para estructurar el contenido de la base de datos. Gracias a este lenguaje, es posible crear, modificar y eliminar objetos individuales, como referencias, relaciones o derechos de usuario.

- Lenguaje de manipulación de datos: mediante el lenguaje de manipulación de datos o DML (Data Manipulation Language), se pueden introducir nuevos registros en la base de datos, así como eliminar, modificar y consultar los que ya contiene. Este lenguaje también permite comprimir y extraer los datos.

Para manipular y gestionar las bases de datos existen herramientas (software) denominados sistemas gestores de bases de datos o SGBD. Es decir, nos permite realizar las funciones de modificar, extraer y almacenar información de una base de datos, además de poseer herramientas con funciones de eliminar, modificar, analizar, etc... datos de estas. Realiza la función concreta de interfaz entre la base de datos y los usuarios finales o los programas correspondientes, organizando los datos y permitiendo su acceso. En internet existen versiones gratuitas, de software libre y software propietario.

En el siguiente esquema podemos ver cómo el usuario interactúa con los datos a través del DBMS o SGBD:



### Organización de una Base de Datos

Toda base de datos organizada debe cumplir los siguientes objetivos:

- **Tiene que ser versátil:** esto quiere decir que, dependiendo de los usuarios o las aplicaciones, puedan hacer diferentes cosas o traten a los datos de formas distintas.
- **Tiene que atender con la rapidez adecuada.**
- **Debe tener un índice de redundancia lo más bajo posible.**
- **Tiene que disponer de una alta capacidad de acceso** para ganar el mayor tiempo posible en la realización de consultas.
- **Tener un alto índice de integridad,** esto significa que al tener muchos usuarios consultando y escribiendo en una misma base de datos no puede haber fallos en la inserción de datos, errores por redundancia o lenta actualización.

## Tipos de Bases de Datos

Hay varios tipos y enfoques de cómo organizar y transaccionar los datos. Nos enfocaremos en las bases de datos Relacionales, pero a los fines informativos dejamos una breve descripción de las otras opciones que existen:

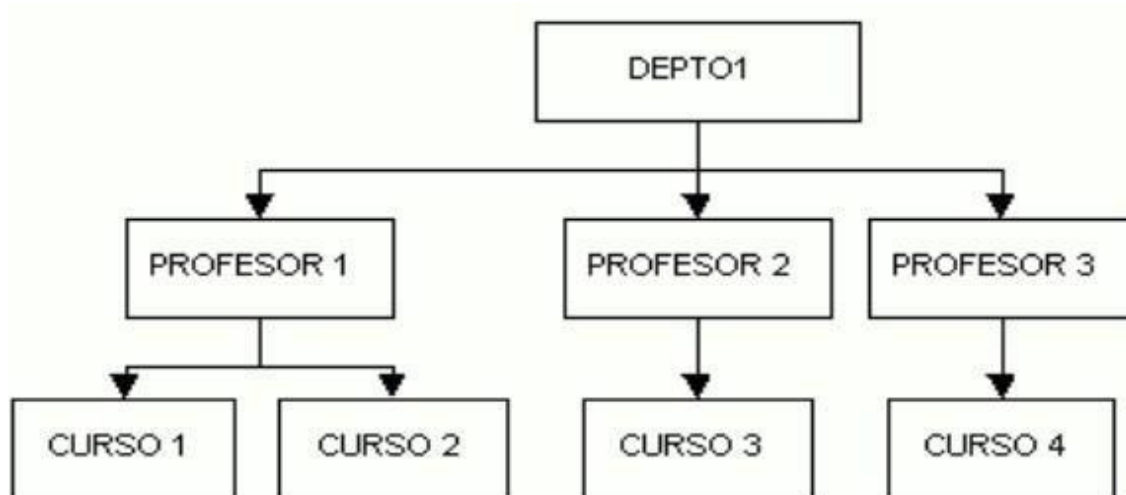
- **Según su variabilidad**

Se encuentran las bases de datos estáticas que son aquellas que sus datos no pueden modificarse, están diseñadas especialmente para la lectura de sus datos. Por lo general se suelen utilizar para almacenar datos históricos, que no cambiarán, para realizar proyecciones estadísticas y ayudar a la toma de decisiones. Por otro lado, las bases de datos dinámicas son aquellas que sus datos se pueden actualizar, ya sea agregando, modificando o eliminando los mismos durante el transcurso del tiempo.

- **Según su contenido**

Según el contenido que almacenan las bases de datos se pueden clasificar de la siguiente forma:

**Bases de datos jerárquicas:** La información se organiza en nodos jerárquicos. Desde un nodo raíz, cada nodo puede contener información de varios hijos y así sucesivamente. Son utilizadas en sistemas con gran volumen de datos y datos compartidos. Un ejemplo de cómo se organizan los datos se muestra en la siguiente imagen:

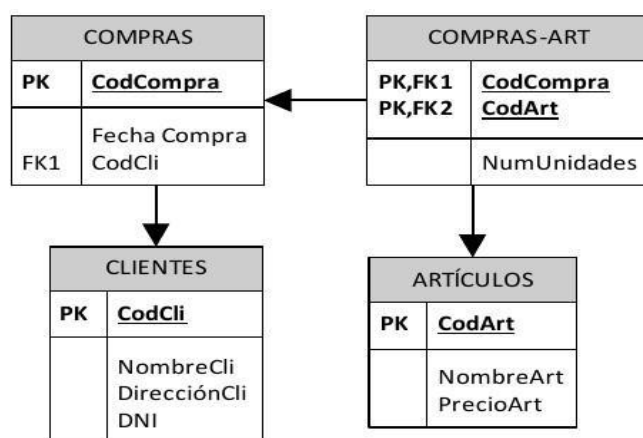


*Base de Datos Jerárquicas*

**Transaccionales:** El diseño de la base de datos está encaminado a recoger las diferentes transacciones que se producen en un sistema. El diseño e implementación están orientados a la rapidez y seguridad de las transacciones, y a la consistencia y durabilidad de los datos. Se deben crear sistemas totalmente fiables, que permitan la recuperación de los errores

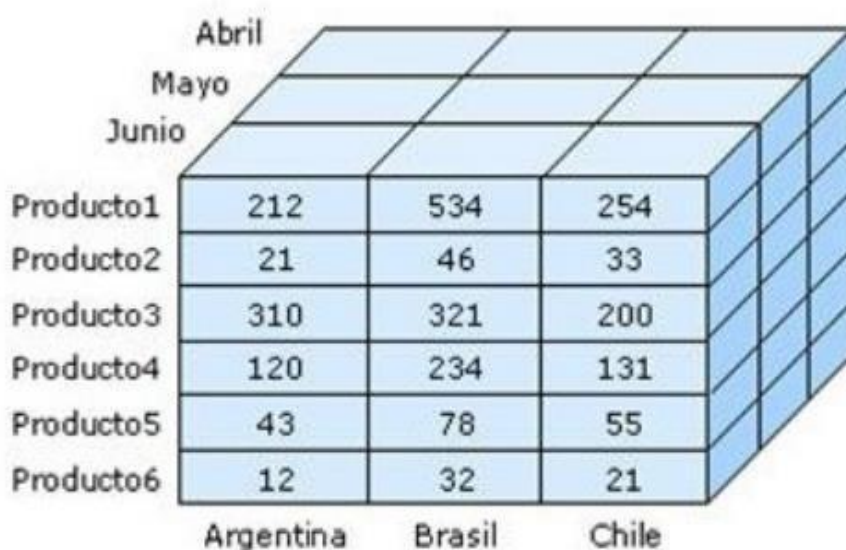
(revirtiendo los procesos que sean necesarios) y que aíslen las diferentes fuentes que acceden al programa para evitar errores e incoherencias.

**Relacionales:** Se basan en la relación de datos estructurados, fiables y homogéneos. Los datos están relacionados conceptualmente, no por su utilización y su implementación en máquina. Las diferentes entidades del sistema son accesibles en tiempo real y compartidas por los usuarios. El centro de los modelos relacionales son las entidades y las relaciones entre ellas, pudiendo haber relaciones normales, de herencia, composición, etc. Todas ellas basadas en las relaciones que se producen en el mundo real de las entidades lógicas.



*Base de Datos relacional*

**Multidimensionales (BDMD):** La información de la base de datos puede verse contenida en una sola tabla multivaluada donde se almacenan registros referidos a las dimensiones o métricas que se van a analizar. Estas tablas se asimilan a un hipercubo, las dimensiones de los cubos se corresponden con la tabla y el valor almacenado en cada celda equivale al de la métrica.





### *Base de datos Multidimensional*

**Orientadas a objetos:** Están basadas en el concepto de objeto de programación. Tradicionalmente, las bases de datos almacenaban los datos y sus relaciones y los procedimientos se almacenaban en los programas de aplicación. En estas bases de datos, sin embargo, se combinan los procedimientos de una entidad con sus datos.



### *Base de datos documental*

**Clave-valor:** Basadas en el concepto de las tablas hash. Se basan en almacenar una serie de registros clave-valor. La clave es un string convencional mientras que los valores pueden ser desde un string hasta una lista o un conjunto. Está diseñada para almacenar grandes cantidades de información, que puede almacenarse en sesión o caché, para ser compartida por varios servidores. Un ejemplo de este tipo de bases de datos es Rendis.

**Orientadas a columnas:** La principal diferencia es que los registros no se organizan en filas sino en columnas. Todo el dominio de valores de un caso de uso, por ejemplo, «Nombre de persona», se pueden acceder como si fueran una única unidad. Cambia totalmente el paradigma de la consulta. Tradicionalmente se recorrían todas las filas de una o varias tablas para devolver los campos seleccionados que cumplieran cierta selección, ahora se seleccionan unos pocos registros de los que nos traemos todo el dominio de valores que cumpla las condiciones. Esto tiene sentido en consultas analíticas.

**Orientada a documentos o NoSQL:** Cada registro corresponde a un documento. Estos documentos se diferencian de los registros de las bases de datos SQL en que se autodefinen ellos mismos, es decir, cada documento define el formato que va a tener, son libres de esquemas (diferente número de campos, campos de longitud variable, campos multivalor, etc.). Los documentos comparten entre sí una parte de información similar y otra muy diferente y se identifican por una clave única, que puede ser desde un string hasta una URI. Las consultas suelen ser dinámicas y muy dispares, se dispone de una API o lenguaje de interrogación para las consultas de documentos según el contenido de los mismos. Este tipo de consultas son muy útiles de cara a los análisis estadísticos. Ejemplos de bases de datos documentales son MongoDB, CouchDB, ArangoDB, etc.

## Bases de Datos Relacional

En este modelo, el lugar y la forma en que se almacenan los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. Los datos pueden ser recuperados o almacenados mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar los datos.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

### Ventajas de una Base de Datos Relacional

#### ***Sus ventajas:***

1. Menor redundancia. No hace falta tanta repetición de datos. Aunque, sólo los buenos diseños de datos tienen poca redundancia.
2. Menor espacio de almacenamiento. Gracias a una mejor estructuración de los datos.
3. Acceso a los datos más eficiente. La organización de los datos produce un resultado más óptimo en rendimiento.
4. Datos más documentados. Gracias a los metadatos que permiten describir la información de la base de datos.
5. Independencia de los datos y los programas y procesos. Esto permite modificar los datos sin modificar el código de las aplicaciones.
6. Integridad de los datos. Mayor dificultad de perder los datos o de realizar incoherencias con ellos.
7. Mayor seguridad en los datos. Al limitar el acceso a ciertos usuarios.

Como contrapartida encontramos los siguientes inconvenientes:

1. Instalación costosa. El control y administración de bases de datos requiere de un software y hardware potente.
2. Requiere personal cualificado. Debido a la dificultad de manejo de este tipo de sistemas.
3. Implantación larga y difícil. Debido a los puntos anteriores. La adaptación del personal es mucho más complicada y lleva bastante tiempo.



## Tablas, Filas y Columnas

Las Bases de Datos Relacional nos permiten agrupar los datos en forma de tablas que, a su vez, dentro de las tablas, los datos se organizan en filas y columnas. La intersección de un registro (fila) y un campo (columna) nos da el dato. Esto nos recuerda mucho a la forma en que trabajamos con las planillas de cálculo como Excel, LibreOffice, OpenOffice, etc.

## Modelo de Datos

Un modelo de datos es una serie de conceptos que se utilizan para describir un conjunto de datos y las operaciones para manipularlos.

Hay dos tipos de modelos de datos: los modelos conceptuales y los modelos lógicos. Los modelos conceptuales se utilizan para representar la realidad a un alto nivel de abstracción. Mediante los modelos conceptuales se puede construir una descripción de la realidad fácil de entender. En los modelos lógicos, las descripciones de los datos tienen una correspondencia sencilla con la estructura física de la base de datos.

En el diseño de bases de datos se usan primero los modelos conceptuales para lograr una descripción de alto nivel de la realidad, y luego se transforma el esquema conceptual en un esquema lógico. El motivo de realizar estas dos etapas es la dificultad de abstraer la estructura de una base de datos que presente cierta complejidad. Un esquema es un conjunto de representaciones lingüísticas o gráficas que describen la estructura de los datos de interés. Los modelos conceptuales deben ser buenas herramientas para representar la realidad, por lo que deben poseer las siguientes cualidades:

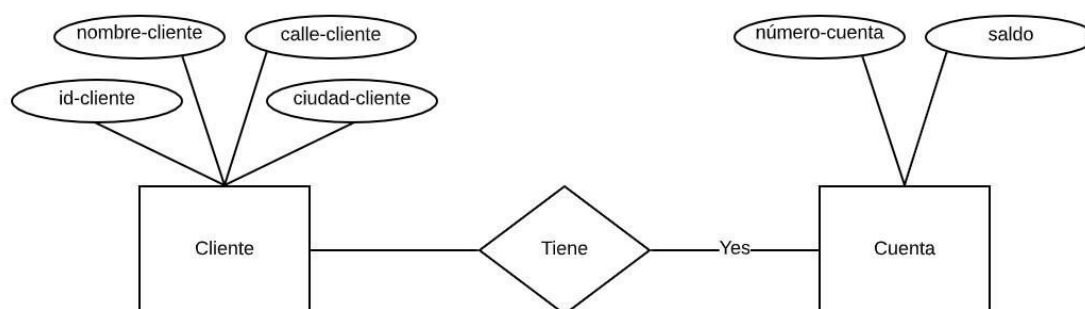
- **Expresividad:** deben tener suficientes conceptos para expresar perfectamente la realidad.
- **Simplicidad:** deben ser simples para que los esquemas sean fáciles de entender.
- **Minimalidad:** cada concepto debe tener un significado distinto.
- **Formalidad:** todos los conceptos deben tener una interpretación única, precisa y bien definida. En general, un modelo no es capaz de expresar todas las propiedades de una realidad determinada, por lo que hay que añadir aserciones que complementen el esquema.

## Modelo Entidad-Relación

Cuando empezamos a pensar o diseñar una base de datos, puede parecer simple en un principio, pero en la medida que nuestra idea de base de datos crece, el diseño también crece y necesitamos una forma de poder conceptualizar y visualizar todos los elementos que la componen, cómo se relacionan y qué características tienen.

Entonces, denominado por sus siglas como: E-R, este modelo representación de la realidad a través de entidades, que son objetos que existen y que se distinguen de otros por sus características, por ejemplo: un alumno se distingue de otro por sus características particulares como lo es el nombre, o el número de control asignado al entrar a una institución educativa, así mismo, un empleado, una materia, etc.

La siguiente imagen ilustra un DER:



*Modelo Entidad Relación*

Hay tres elementos básicos en un modelo ER:

1. Entidad
2. Atributo
3. Relación

**Entidad:** Una entidad representa una "cosa", "objeto" o "concepto" del mundo real.

Algunos ejemplos:

- Una persona: se diferencia de cualquier otra persona, incluso siendo gemelos.
- Un automóvil: aunque sean de la misma marca, el mismo modelo, etc, tendrán atributos diferentes, por ejemplo, el número de chasis.
- Una casa: aunque sea exactamente igual a otra, aún se diferenciará en su dirección.

**Atributos:** Los atributos son las características que definen o identifican a una entidad. como, por ejemplo, para una entidad Persona tenemos: Edad, género, peso, altura, etc.

Veamos otro ejemplo: Consideremos una empresa que requiere controlar a los vendedores y las ventas que ellos realizan; de este problema determinamos que los objetos o entidades principales a estudiar son el empleado (vendedor) y el artículo (que es el producto en venta), y las características que los identifican son:

- Empleado: Nombre, Puesto, Salario, Numero de Empleado
- Artículo: Nombre Descripción Costo

**Conjunto de relaciones:** Consiste en una colección, o conjunto, de relaciones entre las entidades. Por ejemplo: Dadas las entidades "Habitación" y "Huésped", la relación que podemos tener es que uno o más huéspedes estarán alojados en una habitación. Además, a esta relación le podemos dar un título: "Se aloja" ya que los huéspedes se alojan en las habitaciones.

Las relaciones entre las entidades se pueden agrupar en 3 tipos según su finalidad.

### Cardinalidad de las relaciones

La cardinalidad de un atributo indica el número mínimo y el número máximo de valores que puede tomar para cada ocurrencia de la entidad o relación a la que pertenece. El tipo de cardinalidad se representa mediante una etiqueta en el exterior de la relación, respectivamente: "1:1", "1:N" y "N:M".

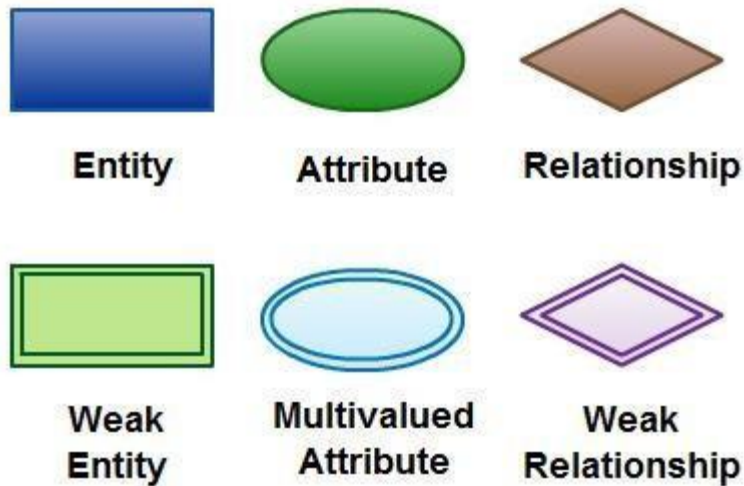
- Uno a Uno (1:1) se da cuando un elemento de una entidad se puede relacionar solamente con un solo registro de otra entidad y viceversa.
- Uno a Muchos (1:M) se da cuando un registro de una entidad A se puede relacionar con cero o muchos registros de otra entidad B y cada registro de la entidad B se relaciona con un sólo registro de la entidad A.
- Muchos a Muchos (N:M) se da cuando un registro de una entidad se relaciona con cero o varios registros de otra entidad.

Si tuviéramos el caso de una base de datos de una escuela podríamos pensar en las siguientes relaciones:

1. Un alumno de una escuela puede estar solamente en un solo curso (1:1)
2. En una materia de la escuela, un alumno tendrá varias notas durante el año (1:N)
3. En la universidad, un alumno puede estar en muchas clases, y las clases pueden tener alumnos de diferentes carreras (N:N)

### Símbolos y notaciones de Diagrama ER

Las entidades, sus atributos y las relaciones se pueden representar de forma gráfica a través de algunos elementos visuales. A continuación, una imagen que nos describe en primer término cómo se visualiza una entidad, un atributo y una relación:

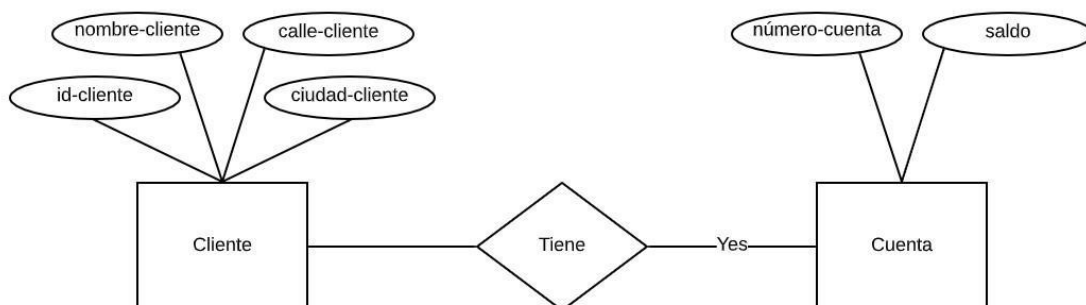


*Símbolos y notaciones de Diagramas ER*

### Caso de aplicación de un DER.




Se nos pide que armemos una base de datos para llevar las cuentas de los clientes. Los datos que necesitamos almacenar de los clientes son el nombre, la ciudad y la calle donde vive, y un número de identificación del cliente. Por otro lado, las cuentas de los clientes tienen un número de cuenta y un saldo.



Si bien en la próxima sección veremos cómo diseñar el DER, al ser ésta una herramienta muy intuitiva, podemos tomar la siguiente imagen y entender las partes ya vistas.













### Elementos adicionales de un DER

Cómo mencionamos dejamos unas tablas adicionales de los diagramas de entidades, relaciones, atributos y cardinalidades a los fines de conocer que existen y se pueden utilizar o encontrar en otros DER más complejos.

Símbolo de entidad	Nombre	Descripción
	Entidad fuerte	Estas figuras son independientes de otras entidades y con frecuencia se les denomina entidades matriz ya que a menudo tienen entidades débiles que dependen de ellas. También tendrán una clave primaria, que distinga a cada suceso de la entidad.
	Entidad débil	Las entidades débiles dependen de algún otro tipo de entidad. No tienen claves primarias y no tienen significado en el diagrama sin su entidad matriz.
	Entidad asociativa	Las entidades asociativas relacionan las instancias de varios tipos de entidades. También contienen atributos que son específicos a la relación entre esas instancias de entidades.

Símbolo de relación	Nombre	Descripción
	Relación	Las relaciones son asociaciones entre dos o más entidades.
	Relación débil	Las relaciones débiles son conexiones entre una entidad débil y su propietario.

Símbolo de atributo	Nombre	Descripción
	Atributo	Los atributos son las características de una entidad, una relación de muchos a muchos, o una relación de uno a uno.
	Atributo de varios valores	Los atributos de valores múltiples son aquellos que pueden tomar más de un valor.
	Atributo derivado	Los atributos derivados son atributos cuyos valores se pueden calcular a partir de valores de atributos relacionados.
	Relación	Las relaciones son asociaciones entre dos o más entidades.

Símbolo de cardinalidad	nombre
	One
	Many
	One (and only one)
	Zero or one
	One or many
	Zero or many



## Cómo hacer un Diagrama Entidad-Relación o DER

Los DER pueden verse confusos, pero no son tan complejos como parecen. Los diagramas entidad-relación se pueden generar fácilmente, y existen muchas herramientas que permiten facilitarnos la tarea como:

- <https://app.creately.com/>
- <https://www.lucidchart.com/>
- <https://app.diagrams.net/>

Para continuar, crearemos un diagrama ER conceptual de un sistema simple en el cual un estudiante se registra para un curso que es impartido por un profesor.

Identificar los componentes

1. Determina las entidades: Las entidades generalmente son sustantivos como auto, banco, estudiante o producto.

En este ejemplo, las tres entidades son “Estudiante”, “Curso” y “Profesor”.



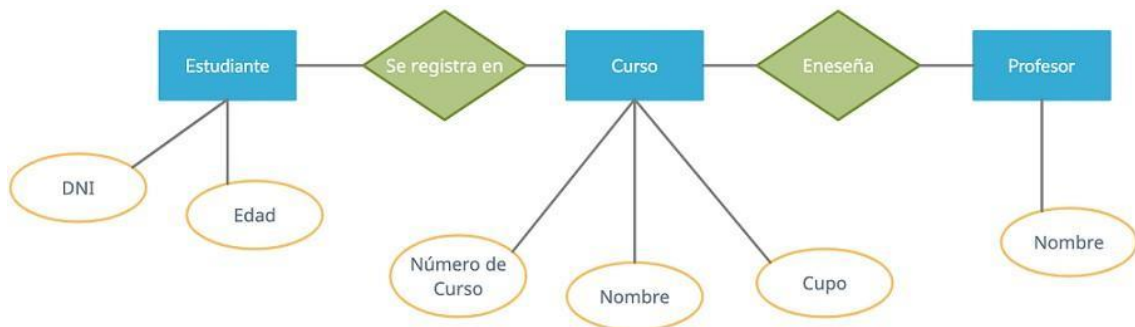
2. Identifica las relaciones: Las relaciones resaltan cómo las entidades interactúan entre sí.

Las relaciones generalmente son verbos como “compra”, “contiene” o “hace”. En nuestro ejemplo, las relaciones “Se registra en” y “Enseña” explican de forma efectiva las interacciones entre las tres entidades.



3. Agrega atributos: Los atributos muestran características específicas de una entidad, detallando qué información es importante para el modelo.

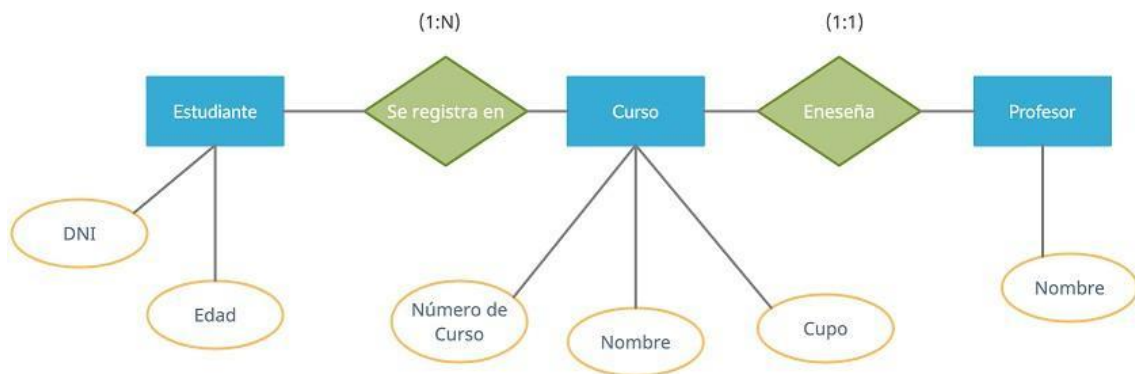
En un diagrama ER, los atributos son necesarios para modelar qué características se incluirán con cada entidad. Siguiendo el ejemplo, los estudiantes tienen un DNI y una edad, los cursos tienen un número de curso, una cantidad de asientos disponibles y el nombre del curso. Los profesores por su parte solamente nos interesa guardar el nombre.



#### 4. Cardinalidad de las relaciones

Para finalizar nos quedaría preguntarnos cómo es la cardinalidad entre los estudiantes.

- En un curso puede haber muchos estudiantes, pero un estudiante solo puede estar en un curso. (1:N)
- En este caso particular, la institución nos exige que solamente pueda haber un profesor por curso, y que cada profesor solamente puede tener asignado un curso.



#### Tipos de Datos

Conocer y definir los tipos de datos en el diseño físico es muy importante porque forman parte de las restricciones que sirven para establecer las reglas de una tabla. Especificando qué tipos de datos se pueden añadir en un registro evita que se inserten datos incorrectos. Es necesario conocer los tipos de datos de SQL. A continuación, una sencilla tabla de tipos de datos que manipulan las bases de datos.

Tipo de Datos	Longitud	Descripción
BIT	1 byte	Valores Si/No - True/False - Verdadero/Falso
DATETIME	8 bytes	Fecha u hora entre los años 100 y 9999.
DOUBLE	8 bytes	Un valor en punto flotante de precisión doble con un rango de - 1.79769313486232*10308 a -4.94065645841247*10-324 para valores <b>negativos</b> siendo para valores <b>positivos</b> entre 4.94065645841247*10-324 a 1.79769313486232*10308 y 0.
LONG	4 bytes	Valor entero largo entre -2,147,483,648 y 2,147,483,647.
LONGTEXT	1 byte por carácter	De cero a un máximo de 1.2 gigabytes.
TEXT	1 byte por carácter	De cero a 255 caracteres.

## Índice

Para facilitar la obtención de información de una tabla se utilizan índices. El índice de una tabla desempeña la misma función que el índice de un libro: permite encontrar datos rápidamente. En el caso de las tablas, localiza registros.

Una tabla se indexa por uno o varios campos, y posibilita el acceso directo y rápido haciendo más eficiente las búsquedas. Sin índice, se debe recorrer secuencialmente toda la tabla para encontrar un registro consumiendo más tiempo y recursos. Entonces el objetivo de un índice es acelerar la recuperación de información. La desventaja es que consume espacio en el disco ya que requiere realizar una tarea de indexación. Es una técnica que optimiza el acceso a los datos, mejora el rendimiento acelerando las consultas y otras operaciones. Es útil cuando la tabla contiene miles de registros. Es importante identificar el o los

campos por los que sería útil crear un índice, aquellos campos por los cuales se realizan operaciones de búsqueda con frecuencia.

Hay distintos índices, a saber:

- **Index:** Puede haber varios por tabla. Es un índice común. Los valores no necesariamente son únicos y aceptan valores nulos. Podemos darle un nombre, si no se lo damos, se coloca uno por defecto. "key" es sinónimo de "index".

- **Unique:** Es un índice para las cuales los valores deben ser únicos y diferentes, aparece un mensaje error si intentamos agregar un registro con un valor ya existente. Permite valores nulos y pueden definirse varios por tabla.
- **Clave Primaria (PK).**

### Clave Primaria

Una clave primaria es la columna o colección de columnas que nos permiten identificar de forma única a una fila determinada en una tabla. La clave primaria proporciona una forma importante de distinguir una fila de otra. Subrayar las columnas o la colección de columnas que componen la clave primaria usualmente es la mejor forma de representar la clave primaria de cada tabla de la base de datos.

Veamos un ejemplo, si tenemos una tabla con datos de personas, el número de documento puede establecerse como clave primaria, es un valor que no se repite; puede haber personas con igual apellido y nombre, incluso el mismo domicilio (padre e hijo, por ejemplo), pero su documento será siempre distinto.

Existen tres tipos de claves primarias:

1. **Clave natural** es una clave primaria compuesta de una columna que identifica de forma única a una entidad, por ejemplo el DNI de una persona o la patente de un auto.
2. **Clave artificial** es una columna creada para una entidad con el propósito de servir únicamente como clave primaria y es visible para los usuarios.
3. **Clave subrogada** es una clave primaria generada por el sistema, usualmente un tipo de datos generado automáticamente que suele estar escondido del usuario. Normalmente se define con el nombre de "id" y se define como un valor autoincremental. Esto hace que, al insertar un nuevo registro, este valor aumente.

### Clave Foránea (FK)

Una clave foránea es un campo o colección de campos de una tabla cuyos valores deben coincidir con los valores de la clave primaria de una segunda tabla.

### Bases de Datos SQL

Si queremos crear, insertar o consultar datos desde una base de datos será necesario interactuar con el motor de la base de datos mediante lenguaje de consulta estructurado o SQL. SQL es un lenguaje de computación que se asemeja al inglés, pero que comprende los programas de base de datos.

Si comprendemos el funcionamiento de SQL podremos crear mejores consultas además de facilitar la forma de solucionar una consulta que no devuelve los resultados que queremos.

## ¿Qué es SQL?

SQL es un lenguaje de computación para trabajar con conjuntos de datos y las relaciones entre ellos. Los programas de bases de datos relacionales, como SQL Server, MySQL, MariaDB, etc. usan SQL para trabajar con datos. El lenguaje SQL no es difícil de leer y entender, incluso para un usuario inexperto resulta algo intuitivo. Al igual que muchos lenguajes de computación, SQL es un estándar internacional reconocido por organismos de estándares como ISO y ANSI.

## Comandos en SQL

Muchas de las acciones como crear tabla e insertar datos se pueden hacer de forma visual a través de un Sistema Gestor de Base de Datos, es importante conocer las principales acciones que podemos hacer con SQL para luego poder realizar consultas y desarrollos más complejos. Los siguientes comandos los podremos ir probando en el SQL Server, pero también podemos hacer pruebas más sencillas con intérpretes online de SQL como los siguientes:

- <http://sqlfiddle.com/>
- <https://www.idoodle.com/execute-sql-online/>

Como ocurre con todo lenguaje, existe una documentación en la cual podemos encontrar mucha más información y detalle. Siempre es importante consultar más detalles.

## Crear una Base de Datos

Todo comienza con la siguiente instrucción.

```
CREATE DATABASE MI_BASE_DE_DATOS
```

## Crear una tabla

Una vez que tenemos una base de datos podemos empezar a crear nuestras tablas o entidades. Es importante en este momento conocer los tipos de datos:

```
CREATE TABLE Nombre_Tabla(  
Nombre_Campo tipo_valor opciones,  
Nombre_Campo_2 tipo_valor opciones  
);
```

Si quisiéramos crear una tabla de Clientes sería por ejemplo así:

```
CREATE TABLE Clientes(  
Id_Cliente int NOT NULL,  
Nombre varchar(50) NOT NULL,  
Apellido varchar(50) NOT NULL,  
Id_Localidad int NOT NULL  
);
```

### Crear Clave Primaria (PK)

Para crear una clave primaria, fundamental en todas las tablas ejecutamos el siguiente comando:

```
ALTER TABLE Nombre_Tabla  
ADD PRIMARY KEY (Nombre_campo);
```

Si lo aplicamos a nuestro ejemplo de la tabla de clientes nos quedaría de la siguiente forma:

```
ALTER TABLE Clientes  
ADD PRIMARY KEY (Id_Cliente);
```

### Crear Clave Foránea (FK)

Para crear una clave foránea recordemos que necesitamos tener una segunda tabla y utilizamos un campo que nos permita relacionar ambas tablas:



```
ALTER TABLE Tabla_origen ADD FOREIGN KEY(campo_tabla_origen)
REFERENCES Tabla_destino (campo_clave_primaria_destino)
```

Aplicado a nuestro ejemplo anterior de clientes, supongamos que tenemos una segunda tabla con los datos de las localidades. Para crear la clave foránea sería:

```
ALTER TABLE Clientes ADD FOREIGN KEY(Id_Localidad)
REFERENCES Localidades (Id_Localidad)
```

## Selección de datos

Una de las sentencias más usadas es el SELECT que sirve para obtener los datos de una tabla. Si queremos seleccionar todos los campos de una tabla aplicamos la siguiente sentencia:

```
SELECT * FROM Nombre_tabla
```

Si queremos seleccionar algunos campos, solamente debemos listarlos separados por coma:

```
SELECT campo_1,campo_2,... FROM Nombre_tabla
```

En nuestro caso si quisiéramos obtener la lista de Nombre y Apellidos de la tabla de Clientes nos quedaría así:

```
SELECT Nombre, Apellido FROM Clientes
```

## Insertar datos

El comando para insertar datos en una tabla es INSERT. Deben respetar el orden de las columnas con los datos que se van a guardar y deben estar separados por comas las columnas y es posible ignorar los campos que permiten valores nulos.

```
INSERT INTO nombre_tabla (columna1, columna2, columna3, ...)
VALUES (valor1, valor2, valor3, ...);
```

Siguiendo con el ejemplo de la tabla Clientes, para agregar un nuevo cliente debemos ejecutar la siguiente instrucción:

```
INSERT INTO Clientes (Nombre, Apellido, Id_Localidad) VALUES ('Juan',
'Gomez', 15)
```

### Actualizando registros

La instrucción UPDATE sirve para actualizar los registros existentes. Es necesario definir una condición que permita identificar los registros que se quieren actualizar:

```
UPDATE Nombre_tabla
SET columna1 = valor1, columna2 = valor2, ...
WHERE condicion;
```

Si queremos actualizar un registro de nuestra base de clientes para cambiar el nombre de un cliente sabiendo previamente cuál es el id del mismo, la sentencia sería la siguiente:

```
UPDATE Clientes SET Nombre = 'Juan Carlos', Apellido = 'Gómez' WHERE
Id_Cliente = 10
```

### Borrar datos

Para borrar registros necesitamos la instrucción DELETE. Nuevamente es necesario y muy importante tener una condición que nos permita identificar los registros a eliminar. En caso de que no se defina esta condición es posible que perdamos todos los registros de nuestra tabla o base de datos. La instrucción sería:

```
DELETE FROM Nombre_tabla WHERE condicion;
```

Si queremos borrar al cliente número 10 de nuestra tabla de clientes la instrucción nos quedaría:

```
DELETE FROM Clientes WHERE Id_Cliente = 10
```

## Operadores

además de los operadores lógicos que ya conocemos (=,!=,<,>,<=, >=) tenemos otros como por ejemplo:

- **IS NULL** para seleccionar aquellos campos que sean nulos
- **IS NOT NULL** para seleccionar aquellos campos que no sean nulos
- **IN** para seleccionar aquellos campos que estén dentro de una lista
- **LIKE** para buscar con algún patrón.

También tenemos la posibilidad de unir condiciones bajo los operadores AND, OR y NOT

## Ordenación de datos

Para esa acción tenemos la instrucción ORDER BY. Se pueden especificar una o más columnas. La sentencia es la siguiente:

```
SELECT columna1, columna2, ...  
FROM Nombre_tabla  
ORDER BY columna1, columna2, ... ASC|DESC;
```

Por ejemplo, si quisiéramos obtener todos los registros de la tabla Clientes ordenados por Apellido ascendente, nuestra consulta quedaría de la siguiente manera:

```
SELECT * FROM Clientes ORDER BY Apellido ASC
```

## Funciones de Agregación

Una función de agregación es un cálculo sobre un conjunto de valores, devolviendo un sólo valor, excepto por la función COUNT. Estas funciones ignoran los valores NULL. Por lo general, debemos combinar las funciones de agregado con la cláusula GROUP BY. Algunas de ellas son MAX (Máximo), MIN (Mínimo), AVG (Promedio), SUM (Suma), COUNT (Cantidad) entre otras.

Por ejemplo, si tuviéramos una tabla Transferencias con un campo monto y quisiéramos obtener el monto máximo, nuestra consulta sería la siguiente: SELECT MAX(Monto) FROM Transferencias.

## Subconsultas

Una subconsulta es una instrucción SELECT anidada dentro de otra instrucción SELECT, las cuales pueden contener condiciones como las mencionadas anteriormente. Para generar una subconsulta debemos tener presente ciertas reglas como: escribir la subconsulta entre paréntesis, especificar en ella sólo una columna (si no se utiliza IN, ANY, ALL o EXISTS), que no contenga la cláusula GROUP BY.

Además, que no se puede recuperar datos de la misma tabla a la cual se realice la sentencia UPDATE o DELETE.

Por ejemplo, si quisiéramos obtener los Clientes que tienen en el campo Id\_Localidad cualquier ID que esté en la tabla Provincias siendo Córdoba el nombre de la misma, nuestra consulta sería la siguiente:

```
SELECT * FROM Clientes WHERE Id_Localidad IN(SELECT Id_Localidad FROM Provincias WHERE Nombre = 'Córdoba')
```

La subconsulta obtendrá todos los valores de Id\_Localidad que contenga la provincia Córdoba y la consulta obtendrá los Clientes que tengan el valor de Id\_Localidad que estén en la lista de la subconsulta.