# Introducción

Una **base de datos** es una colección de datos relacionados. Con la palabra **datos** nos referimos a los hechos(datos) conocidos que se pueden grabar y que tienen un significado implícito

* Una base de datos representa algún aspecto del mundo real.
* Una base de datos es una colección de datos lógicamente coherente con algún tipo de significado inherente.
* Una base de datos se diseña, construye y rellena con datos para un propósito especifico.

# Sistema de administración de datos (DBSM, database management system)

Es una colección de programas que permite a los usuarios crear y mantener una base de datos. El DBMS es un sistema de software de propósito general que facilita los procesos de definición, construcción, manipulación y compartición de bases de datos entre varios usuarios y aplicaciones.

* **Definir** una base de datos implica especificar los tipos de datos, estructuras y restricciones de los datos que se almacenaran en la base de datos.

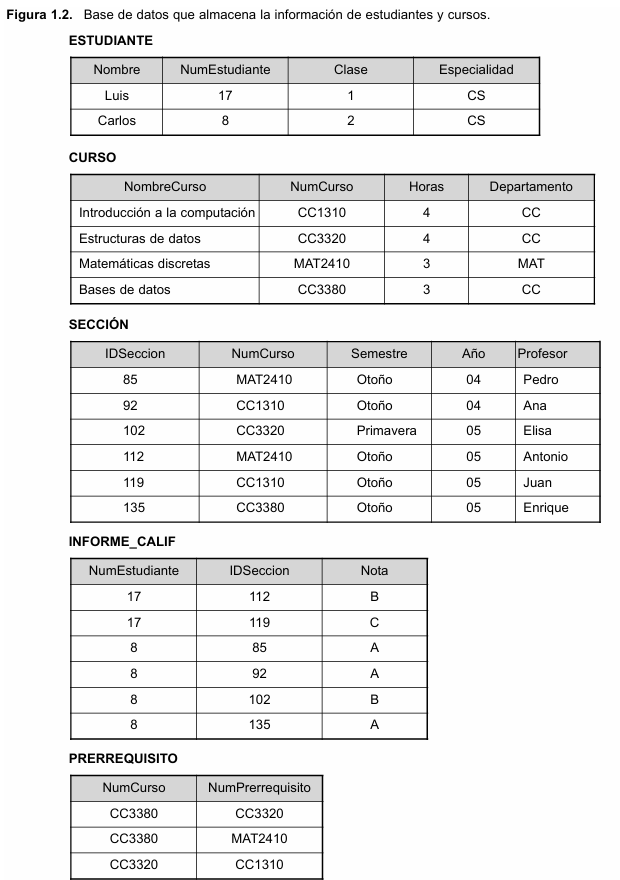
**La definición** o información descriptiva de una base de datos también se almacena en esta ultima forma de catalogo o diccionario de la base de datos, es lo que se conoce como **metadatos.**

* **La construcción** de la base de datos es el proceso consistente en almacenar los datos en algún medio de almacenamiento controlado por el DBMS.
* **La manipulación** de una base de datos incluye funciones como la consulta de la base de datos para recuperar datos específicos, actualizar la base de datos para reflejar los cambios introducidos en el minimundo y generar informes a partir de los datos.
* **Compartir** una base de datos permite que varios usuarios y programas accedan a la base de datos de forma simultánea.

Una **aplicación** accede a la base de datos enviando consultas o solicitudes de dato al DBMS. Una **consulta** normalmente provoca la recuperación de algunos datos una **transacción** puede provocar la lectura o la escritura de algunos datos en la base de datos.

Otras funciones importantes ofrecidas por el DBMS son la protección de la base de datos y su mantenimiento.

* **La protección** incluye la protección del sistema contra el funcionamiento defectuoso del hardware o el software (caídas) y la protección de la seguridad contra el acceso no autorizado o malintencionado.
* El DBMS debe ser capaz de **mantener** el sistema de bases de datos permitiendo que el sistema evolucione según cambian los requisitos con el tiempo.



Para **definir** esta base de datos debemos especificar la estructura de los registros de cada archivo detallando los diferentes tipos de **elementos de datos** que se almacenaran en cada registro. También hay que especificar un **tipo de datos** para cada elemento de datos de un registro. Por ejemplo, podemos especificar que el Nombre de un “ESTUDIANTE” es un **string**, que “NumEstudiante” es un **entero**, etc.

La **construcción** de la base de datos “UNIVERSIDAD” se realiza almacenando los datos que representan a todos los estudiantes, cursos, secciones…, a modo de registro en el archivo adecuado. Los registros de los distintos archivos se pueden relacionar (“ESTUDIANTE” con “INFORME\_CALIF).

* **Manipulación:** 
  + Recuperar el certificado de estudios de “Luis”.
  + Listado de los prerrequisitos del curso “Bases de datos”.
* **Actualización:**
  + Cambiar la clase de “Luis” a estudiante de segundo año.
  + Crear una sección nueva para el curso.

El diseño de una aplicación nueva para una base de datos existente o el diseño de una base de datos nueva empieza con una fase denominada **definición de requisitos y análisis**. Estos requisitos son documentados en detalle y transformados en un **diseño conceptual** que se puede representar y manipular mediante algunas herramientas computarizadas, de modo que en una implementación de base de datos puedan mantenerse, modificarse y transformarse fácilmente. El diseño después se convierte en un **diseño lógico** que se puede expresar en un modelo de datos implementado en un DBMS comercial. La etapa final es el **diseño físico**, durante la que se proporcionan especificaciones suplementarias para el almacenamiento y acceso a la base de datos.

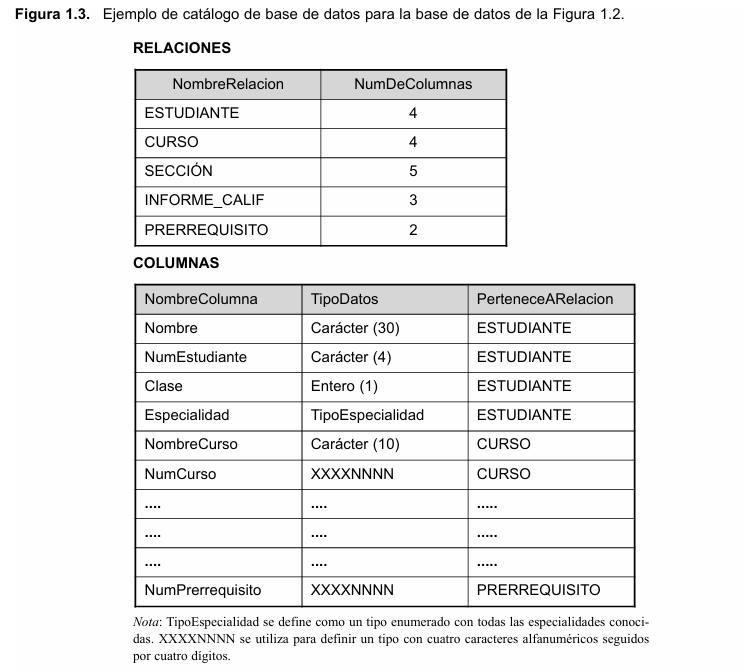
En la metodología de bases de datos se mantiene un único almacén de datos, que se define una sola vez, y al que acceden varios usuarios. En los sistemas de archivos cada aplicación tiene libertad para asignar un nombre independientemente a los elementos de datos. Por el contrario, en una base de datos, los nombres o etiquetas de los datos se definen una vez, y son utilizados por consultas, transacciones y aplicaciones.

**Características de la metodología de bases de datos**

* **Naturaleza autodescriptiva de un sistema de bases de datos**

Según la metodología de bases de datos, el sistema de bases de datos contiene una completa definición o descripción de la estructura de la base de datos y sus restricciones, esta definición se almacena en el catálogo DBMS el cual contiene información como la estructura de cada archivo, el tipo y el formato de almacenamiento de cada elemento de datos y sus restricciones. La información guardada en el catálogo se denomina **metadatos** y describe a la estructura de la base de datos.

En el procesamiento de **archivos** tradicional, normalmente la definición de datos forma parte de los programas de aplicación. Esas aplicaciones están restringidas a trabajar solo con una base de datos especifica, cuya estructura esta declarada en dichas aplicaciones.



El diseñador de la base de datos especifica estas definiciones antes de crear la base de datos y se almacenan en el catálogo. Siempre que se crea una solicitud para acceder, por ejemplo, al Nombre de un registro ESTUDIANTE, el DBMS recurre al catálogo para determinar la estructura del archivo ESTUDIANTE y la posición y tamaño del elemento de datos Nombre dentro de un registro ESTUDIANTE.

En una aplicación de procesamiento de archivos típica, la estructura del archivo y la ubicación exacta de Nombre dentro de un registro ESTUDIANTE también están codificadas dentro de cada programa que accede a dicho elemento.

* **Aislamiento entre programas y datos, y abstracción de datos**

En el procesamiento de archivos tradicional los cambios que se introducen en la estructura de un archivo pueden obligar a realizar cambios en todos los programas que acceden a ese archivo. Por el contrario, Los archivos que acceden a un DBMS no necesitan esos cambios. La estructura de los archivos de datos se almacena en el catalogo DBMS independientemente de los programas de acceso. A esto lo llamamos **independencia programa-datos**.

Una **operación** (o **función o método**) se especifica de dos formas:

* **La interfaz** (o **firma**) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros).
* **La implementación** (o **método**) de la operación se especifica separadamente y puede modificarse sin que la interfaz se vea afectada.

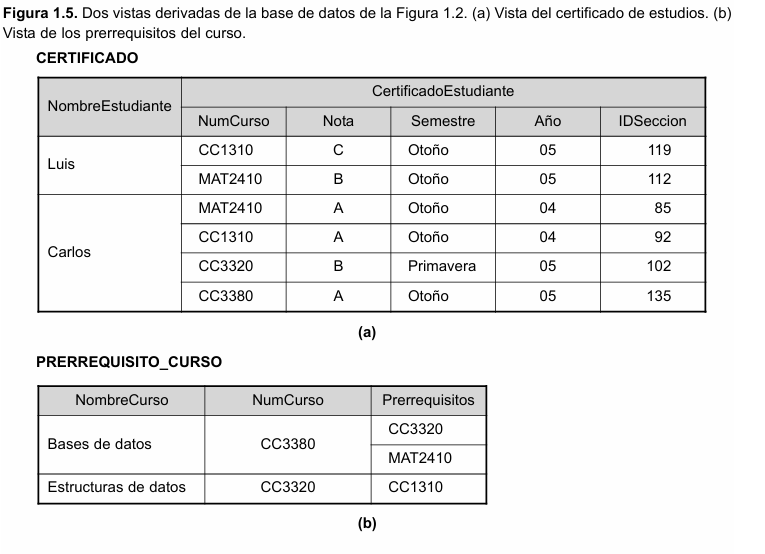
Las apps de usuarios pueden operar sobre los datos invocando estas operaciones por sus nombres y argumentos, independientemente de como estén implementadas las operaciones. A esto se le llama **independencia programa-operación**.

La característica que permite la independencia programa-datos y la independencia programa-operación se denomina **abstracción de datos.** El DBMS proporciona a los usuarios una **representación conceptual** de los datos que no incluye muchos detalles. Un **modelo de datos** es un tipo de abstracción de datos que se utiliza para proporcionar esa representación conceptual,

En las bases de datos orientadas a objetos y de objetos relacionales, el proceso de abstracción no solo incluye la estructura de datos, sino también las operaciones sobre los datos. Por ejemplo, se puede aplicar una operación CALCULAR\_CM a un objeto ESTUDIANTE para calcular la calificación media. En este sentido una abstracción de la actividad del minimundo queda a disposición de los usuarios como una **operación abstracta**.

* **Soporte de varias vistas de los datos**

Una **vista** puede ser un subconjunto de la base de datos o puede contener **datos virtuales** derivados de los archivos de la base de datos pero que no están explícitamente almacenados.



* **Compartición de datos y procesamiento de transacciones multiusuario**

Un DBMS multiusuario debe permitir que varios usuarios puedan acceder a la base de datos al mismo tiempo. Esto es esencial si los datos destinados a varias aplicaciones serán integrados y mantenidos en una sola base de datos. El DBMS debe incluir el software de **control de la concurrencia** para que esos varios usuarios que intentan actualizar los mismos datos, lo hagan de un modo controlado para que el resultado de la actualización sea correcto. Estos tipos de aplicaciones se denominan aplicaciones de **procesamiento de transacciones en línea** (**OLTP, online transaction processing**).

Una **transacción** es un programa en ejecución o proceso que incluye uno o mas accesos a la base de datos. El DBMS debe implementar varias propiedades de transacción:

* La propiedad **aislamiento** garantiza que parezca que cada transacción se ejecuta de forma aislada de otras transacciones, aunque puedan estar ejecutándose cientos de transacciones al mismo tiempo.
* La propiedad **atomicidad** garantiza que se ejecuten o todas o ninguna de las operaciones de bases de datos de una transacción.

**Actores de la escena (duki y khea bro)**

En esta sección identificamos las personas cutos trabajos implican el uso diario de una base de datos grande; denominados **actores de la escena**.

* **Administradores de las bases de datos**

En un entorno de bases de datos, el recurso principal es la base de datos en si misma, mientras que el recurso secundario es el DBMS y el software relacionado. La administración de estos recursos es responsabilidad del **administrador de la base de datos (DBA, database administrator).**

* **Diseñadores de las bases de datos**

Son los responsables de identificar los datos que se almacenaran en la base de datos y de elegir las estructuras apropiadas para representar y almacenar esos datos. Deben comunicarse con todos los usuarios de la base de datos para conocer sus requisitos, a fin de crear un diseño que satisfaga sus necesidades. Estos desarrollan **vistas** de la base de datos que satisfacen los requisitos de datos y procesamiento de esos grupos de usuarios.

* **Usuarios finales**

Son las personas cutos trabajos requieren acceso a la base de datos para realizar consultas, actualizaciones e informes. Se clasifican en:

* + Los **usuarios finales casuales** acceden ocasionalmente a la base de datos, pero pueden necesitar una información diferente en cada momento.
  + Los **usuarios finales principiantes**, su labor principal gira entorno a la consulta y actualización constantes de la base de datos utilizando tipos de consultas y actualizaciones estándar (**transacciones enlatadas**) que se han programado y probado cuidadosamente.
  + Entre los **usuarios finales sofisticados** se encuentran los ingenieros, los científicos, los analistas comerciales y otros muchos que están completamente familiarizados con el DBMS.
  + Los **usuarios finales independientes** mantienen bases de datos personales utilizando paquetes de programas confeccionados que proporcionan unas interfaces fáciles de usar y basadas en menús o gráficos.
* **Analistas de sistemas y programadores de aplicaciones (ingenieros de software)**

Los analistas de sistemas determinan los requisitos de los usuarios finales, así como las especificaciones de desarrollo para las transacciones enlatadas que satisfacen esos requisitos. Los **programadores de aplicaciones** implementan esas especificaciones como programas después, verifican, depuran, documentan y mantienen esas transacciones enlatadas. Dichos analistas y programadores (o **desarrolladores de software** o **ingenieros de software**) deben familiarizarse con todas las posibilidades proporcionadas por el DBMS al objeto de desempeñar sus tareas.

**Trabajadores entre bambalinas**

Hay otros usuarios que están asociados con el diseño, el desarrollo y el funcionamiento de un entorno de software y sistema DBMS:

* **Diseñadores e implementadores de sistemas DBMS:** Diseñan e implementan los módulos y las interfaces DBMS como un paquete software.
* **Desarrolladores de herramientas:** Diseñan e implementan **herramientas** (paquetes de software que facilitan el modelado y el diseño de la base de datos, el diseño del sistema de bases de datos y la mejora del rendimiento). Las herramientas son paquetes opcionales que a menudo se compran por separado.
* **Operadores y personal de mantenimiento:** Son los responsables de la ejecución y el mantenimiento real del entorno hardware y software para el sistema de bases de datos.

**Ventajas de utilizas una metodología DBMS**

* **Control de la redundancia:**

En el desarrollo tradicional de software que hace uso del procesamiento de archivos, cada grupo de usuarios mantiene sus propios archivos para manipular sus aplicaciones de procesamiento de datos. La **redundancia** resultante de almacenar los mismos datos varias veces conduce a serios problemas.

* + En primer lugar, las actualizaciones lógicas sencillas (introducción de datos, por ejemplo) hay que hacerlas varias veces (una por cada archivo donde se almacenen los datos de los estudiantes), esto lleva a una **duplicación del esfuerzo**.
  + En segundo lugar, se **derrocha espacio de almacenamiento** al guardar repetidamente los mismos datos, y este problema puede llegar a ser muy serio en las bases de datos grandes.
  + En tercer lugar, los archivos que representan los mismos datos pueden acabar siendo **incoherentes**, lo que puede ocurrir cuando una determinada actualización se aplica a unos archivos y a otros no. Incluso si una actualización se aplica a todos los archivos adecuados.
* **Restricción del acceso no autorizado:**

Cuando varios usuarios comparten una base de datos grande, la mayoría de los mismos no tienen autorización para acceder a toda la información de la base de datos seguramente. Normalmente los usuarios o grupos de usuarios tienen números de cuenta protegidos mediante contraseñas, que pueden utilizar para tener acceso a la base de datos.

* **Almacenamiento persistente para los objetos del programa:**

Las bases de datos se pueden utilizar para proporcionar **almacenamiento persistente** a los objetos de programa y las estructuras de datos. Un objeto complejo de C++ se puede almacenar de forma permanente en un DBMS orientado a objetos. Se dice que dicho objeto es **persistente**, porque sobrevive a la terminación de la ejecución del programa y otro programa C++ lo puede recuperar más tarde.

* **Suministro de estructuras de almacenamiento para un procesamiento eficaz de las consultas:**

Los sistemas de bases de datos deben proporcionar capacidades para **ejecutar eficazmente consultas y actualizaciones**. Como la base de datos normalmente se almacena en el disco, el DBMS debe proporcionar estructuras de datos especializados para acelerar la búsqueda en el disco de los registros deseados. Con este fin se utilizan unos archivos auxiliares denominados **índices** usados para la búsqueda en disco.

* **Copia de seguridad y recuperación:**

Un DBMS debe ofrecer la posibilidad de recuperarse ante fallos del hardware o del software. El **subsistema de copia de seguridad recuperación** del DBMS es el responsable de la recuperación.

* **Suministro de varias interfaces de usuario:**

Tanto las interfaces al estilo de los formularios como las basadas en menús se conocen normalmente como **interfaces graficas de usuario (GUI, graphical user inerfaces)**.

* **Representación de relaciones complejas entre los datos:**

Un DBMS debe tener la capacidad de representar las relaciones complejas entre los datos, definir las nuevas relaciones que surgen, y recuperar y actualizar fácil y eficazmente los datos relacionados.

* **Implementación de las restricciones de integridad:**

El tipo de restricción de integridad mas simple consiste en especificar un tipo de datos por cada elemento de datos. Un tipo de restricción mas compleja que se da a menudo implica especificar que un registro de un archivo debe estar relacionado con registros de otros archivos. Otro tipo de restricción especifica la **unicidad** en los valores del elemento de datos. Otras restricciones pueden tener que ser comprobadas por los programas de actualización o en el momento de introducir los datos. En las aplicaciones grandes es costumbre denominar estas restricciones como **reglas de negocio**.

En el modelo Entidad-Relación, una relación debe implicar como mínimo dos entidades. Estas reglas son **reglas inherentes** del modelo de datos y se asumen automáticamente para garantizar la validez del modelo.

* **Inferencia y acciones usando reglas:**

Algunos sistemas de bases de datos ofrecen la posibilidad de definir **reglas de deducción** para **inferir** información nueva a partir de los hechos guardados en la base de datos. Estos sistemas se denominan **sistemas de bases de datos deductivos**.

En los sistemas de bases de datos relacionales actuales es posible asociar **triggers** a las tablas, un **trigger** es una forma de regla que se activa con las actualizaciones de la tabla, lo que conlleva la ejecución de algunas operaciones adicionales sobre otras tablas. Los procedimientos más implicados en la implementación de reglas se conocen popularmente como **procedimientos almacenados**; se convierten en parte de la definición global de la base de datos y se les invoca correctamente cuando se dan ciertas condiciones. Los **sistemas de bases de datos activos** ofrecen la funcionalidad más potente; proporcionan reglas activas que pueden iniciar automáticamente acciones cuando ocurren ciertos eventos y condiciones.

**Unidad 2**

**Modelos de datos, esquemas e instancias**

La **abstracción de datos** se refiere generalmente a la supresión de detalles de la organización y el almacenamiento de datos y a la relevancia de las características fundamentales para un conocimiento mejorado de los datos. Una de las características principales de la metodología de bases de datos es soportar la abstracción de datos para que diferentes usuarios puedan percibir esos datos con el nivel de detalle que prefieren.

Un **modelo de datos** (colección de conceptos que se pueden utilizar para describir la estructura de una base de datos) proporciona lo medios necesarios para conseguir esa abstracción. La mayoría de modelos de datos también incluyen un conjunto de **operaciones básicas** para especificar las recuperaciones y actualizaciones en la base de datos.

Además de las operaciones básicas proporcionadas por el modelo de datos, es cada vez más común incluir conceptos en el modelo de datos para especificar el **aspecto dinámico** o **comportamiento** de una aplicación de base de datos. Esto permite al diseñador de la base de datos especificar un conjunto de operaciones validas definidas por el usuario que son permitidas en los objetos de la base de datos.

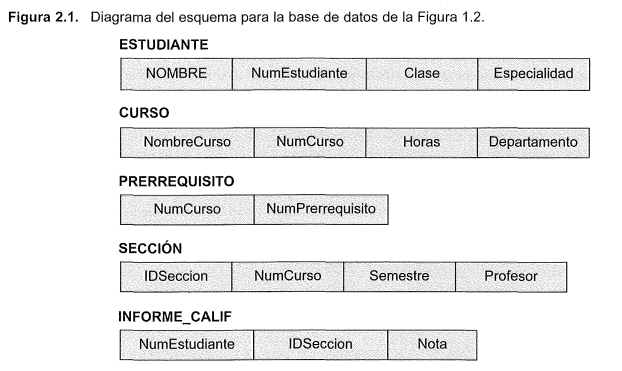
* **Categorías de modelos de datos:**
  + Los **modelos de datos de alto nivel** o **conceptuales** ofrecen conceptos muy cercanos a como muchos usuarios perciben los datos, mientras que los **modelos de datos de bajo nivel** o **físicos** ofrecen conceptos que describen los detalles de cómo se almacenan los datos en el computador. Los conceptos ofrecidos por los modelos de datos de bajo nivel están pensados principalmente para los especialistas en computadoras, no para los usuarios finales normales.
  + Entre estos dos extremos hay una clase de **modelos de datos representativos** (o de **implementación**) que ofrecen conceptos que los usuarios finales pueden entender pero que no están demasiado alejados de cómo se organizan los datos dentro del computador.
  + Los modelos de datos **representativos** ocultan algunos detalles relativos al almacenamiento de los datos, pero pueden implementarse directamente en el computador.
  + Los modelos de datos **conceptuales** utilizan conceptos como entidades, atributos y relaciones
    - **Entidad:** Representa un objeto o concepto del mundo real.
    - **Atributo:** Representa alguna propiedad de interés que describe a una entidad.
    - **Relación:** Representa una asociación entre dos o más entidades.
  + Los modelos de datos **representativos o de implementación** son los más utilizados, incluyen los modelos de datos relacionales ampliamente utilizados, así como los modelos de datos heredados (los modelos de red y **jerárquicos**).
  + Los modelos de datos **representativos** representan los datos mediante estructuras de registro, se los conoce a veces como **modelos de datos basados en registros**.
  + El **grupo de modelos de datos de objetos (ODMG, objetct data model group)** es una nueva familia de modelos de datos de implementación de alto nivel que está más cercana a los modelos de datos conceptuales.
  + Los modelos de datos **físicos** describen como se almacenan los datos en el computador en forma de archivos representando la información como formatos de registro, ordenación de registros y rutas de acceso.
  + Un **índice** es un ejemplo de ruta de acceso que permite el acceso directo a los datos que utilizan un término del índice o palabra clave.
* **Esquemas, instancias y estado de la base de datos:**

Es importante distinguir entre la **descripción** de la base de datos y **la misma base de datos**. **La descripción** de una base de datos se denomina **esquema de la base de datos**, que se especifica durante la fase de diseño y no se espera que cambie con frecuencia.

Un esquema visualizado se denomina **diagrama del esquema**. A cada objeto del esquema lo denominamos **estructura de esquema**. Un diagrama del esquema solo muestra algunos **aspectos** de un esquema, como los nombres de los tipos de registros y los elementos de datos, y algunos tipos de restricciones.

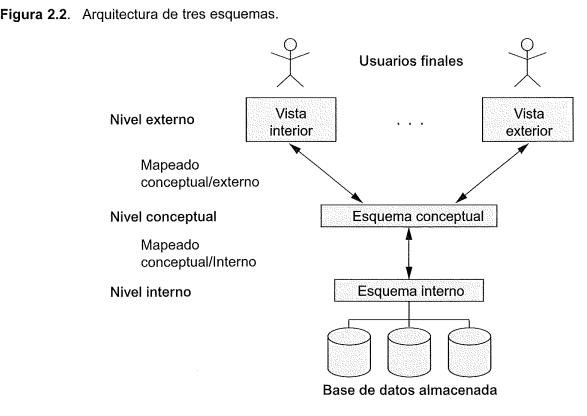
Los datos de la base de datos en un momento concreto se denominan **estado de la base de datos** o **snapshot (captura)** y también reciben el nombre de **conjunto actual** **de ocurrencias o instancias** de la base de datos.

* + Cuando **definimos** una base de datos, solo especificamos su esquema al DBMS.
  + El estado correspondiente de la base de datos es el **estado vacío**.
  + El **estado inicial** se da cuando la base de datos se **rellena o carga** por primera vez con los datos iniciales.
  + El DBMS almacena las descripciones de las construcciones de esquema y las restricciones **(metadatos**) en el catálogo del DBMS.
  + El esquema recibe el nombre de **intención** y el estado de la base de datos **extensión** del esquema.
  + No es raro que ocasionalmente haya que introducir algún cambio en él al cambiar los requisitos de la aplicación, esto se conoce como **evolución del esquema**.



**Arquitectura de tres esquemas e independencia de los datos**

En esta sección se especificará una arquitectura para los sistemas de bases de datos, denominada **arquitectura de tres esquemas**, que se propuso para ayudar a conseguir y visualizar estas características (1\_**Aislamiento de los programas y los datos, 2\_Soporte de varias vistas de usuario y 3\_Uso de un catálogo para almacenar la descripción de la base de datos (esquemas))**

1. El **nivel interno** tiene un **esquema interno** que describe la estructura de almacenamiento físico de la base de datos.
2. El **nivel conceptual** tiene un **esquema conceptual** que describe la estructura de toda la base de datos para una comunidad de usuarios. Este esquema oculta los detalles de las estructuras de almacenamiento físico y se concentra en describir las entidades, los tipos de datos, las relaciones, las operaciones de los usuarios y restricciones.
3. El **nivel de vista** o **externo** describe la parte de la base de datos en la que un grupo de usuarios en particular está interesado y le oculta el resto de la base de datos.

* **Independencia de los datos:**

La arquitectura de **tres esquemas** se puede utilizar para explicar el concepto de **independencia de datos,** que puede definirse como la capacidad de cambiar el esquema en un nivel de sistema de bases de datos sin tener que cambiar el esquema en el siguiente nivel más alto.

1. **Independencia lógica de datos:** Es la capacidad de cambiar el esquema conceptual sin tener que cambiar los esquemas externos o los programas de aplicación. Es posible cambiar el esquema conceptual para expandir la base de datos (añadiendo un tipo de registro o un elemento de datos), para cambiar las restricciones o para reducir la base de datos (eliminando un tipo de registro o un elemento de datos).
2. **Independencia física de datos:** Es la capacidad de cambiar el esquema interno sin que haya que cambiar el esquema conceptual.

**Lenguajes e interfaces de bases de datos**

Se explicará los tipos de lenguajes e interfaces proporcionados por un DBMS y las categorías de usuarios a las que se dirige cada interfaz.

* **Lenguajes DBMS:**

Al terminar el diseño de una base de datos y elegido un DBMS, lo primero es especificar los esquemas conceptual e interno para la base de datos y cualquiera mapeados entre los dos.

* + En DBMSs donde no se mantiene una separación estricta entre niveles, se utiliza un lenguaje denominado **lenguaje de definición de datos (DDL, data definition language)**, para definir los dos esquemas. En los DBMSs donde hay una clara separación entre los niveles conceptual e interno, se utiliza DDL solo para especificar el esquema conceptual.
  + Para especificar el esquema interno se utiliza otro lenguaje, el **lenguaje de definición de almacenamiento (SDL, storage definition language)**.
  + Para conseguir una arquitectura de tres esquemas real se necesita un tercer lenguaje, el **lenguaje de definición de vistas (VDL, view definition language)**, para especificar las vistas de usuario y sus mapeados al esquema conceptual.
  + Una vez compilados los esquemas de la base de datos y rellenada esta con datos, los usuarios deben disponer de algunos medios para manipularla. El DBMS proporciona un conjunto de operaciones o un lenguaje denominado **lenguaje de manipulación de datos (DML, data manipulation language)** para todas estas tareas.

Hay dos tipos de DML:

* + - Se puede utilizar un DML de **alto nivel** o **no procedimental** para especificar de forma concisa las operaciones complejas con las bases de datos.
    - Un DML de **bajo nivel** o **procedimental** debe incrustarse en un lenguaje de programación de propósito general. Normalmente este tipo de DML recupera registros individuales u objetos de la base de datos, y los procesa por separado.
  + Siempre que hay comandos DML, de alto o de bajo nivel, incrustados en un lenguaje de programación de propósito general, ese lenguaje se denomina **lenguaje host** y el DML **sublenguaje de datos.** Por el contrario, un DML de alto nivel utilizado de forma interactiva independiente se conoce como **lenguaje de consulta**.
* **Interfaces de los DBMSs:**
  + **Interfaces basadas en menús para los clientes web o la exploración:** Estas interfaces presentan al usuario listas de opciones (**menús**), estos eliminan la necesidad de memorizar los comandos específicos y la sintaxis de un lenguaje de consulta.
  + **Interfaces basadas en formularios:** Los usuarios pueden rellenar las entradas del **formulario** para insertar datos nuevos, o rellenar únicamente ciertas entradas, en cuyo caso el DBMS recuperara los datos coincidentes para el resto de entradas. Estos se diseñan normalmente para los usuarios principiantes como interfaces para las transacciones enlatadas. Muchos DBMSs tienen **lenguajes de especificación de formularios** que son lenguajes especiales que ayudan a los programadores a especificar dichos formularios.
  + **Interfaces graficas de usuario:** Una GUI muestra un esquema. El usuario puede especificar entonces una consulta manipulándolo el diagrama.
  + **Interfaces de lenguaje natural:** Aceptan consultas escritas en inglés y otro idioma e intentan entenderlas. Normalmente tienen su propio esquema, que es parecido al esquema conceptual de la base de datos, así como un diccionario de palabras importantes. La interfaz de lenguaje natural se refiere a las palabras de su esquema, así como al conjunto de palabras estándar de su diccionario, para interpretar la consulta.
  + **Entrada y salida de lenguaje hablado:** La entrada de lenguaje hablado se detecta mediante una librería de palabras predefinidas que se utilizan para configurar los parámetros que se suministran a las consultas.
  + **Interfaces para el DBA:** La mayoría de los sistemas de bases de datos contienen comandos privilegiados que solo puede utilizar el personal del DBA.

**Entorno de un sistema de bases de datos**

* **Módulos componentes de un DBMS:**

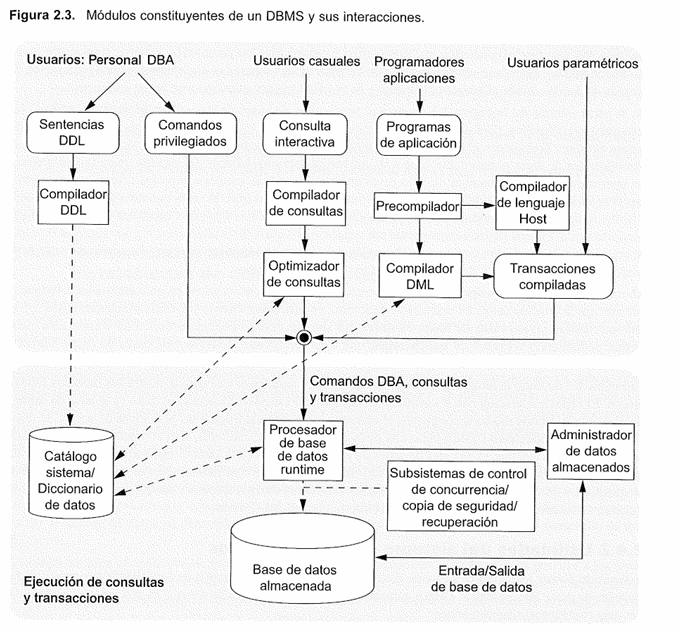
El acceso al disco esta principalmente controlado por el **sistema operativo (SO)** que planifica la E/S del disco. Un módulo **administrador de los datos almacenados** de alto nivel controla el acceso a la información del DBMS almacenada en el disco.

Los usuarios casuales y las personas con una necesidad ocasional de información de la base de datos interactúan utilizando alguna forma de interfaz, que mostramos, como la **interfaz de consulta interactiva**.

Un **compilador de consultas** analiza estas consultas sintácticamente para garantizar la corrección de las operaciones de los modelos, los nombres de los elementos de datos, etcétera, y luego los compila todo en un formato interno. El **optimizador de consultas** se ocupa de la reconfiguración y la posible reordenación de operaciones, eliminación de redundancias y uso de los algoritmos e índices correctos durante la ejecución.

Los programadores de aplicaciones escriben programas en lenguajes **host** como Java, C o COBOL, que son enviados a un **precompilador**. Este extrae los comandos DML de un programa de aplicación escrito en un lenguaje de programación **host**.

Es normal tener el **programa cliente** que accede al DBMS ejecutándose en un computador diferente al que alberga la base de datos. El primero se denomina **computador cliente** y ejecuta un cliente DBMS, y el ultimo se denomina **servidor de bases de datos**. El cliente a veces accede a un computador intermedio, denominado **servidor de aplicaciones**, que a su vez accede al servidor de bases de datos.

****

* **Clasificación de los sistemas de administración de bases de datos:**

El primero es el **modelo de datos** en el que el DBMS está basado. El **modelo de datos relacional** es el modelo de datos principal que se utiliza en muchos de los DBMS’s comerciales actuales. Muchas aplicaciones heredadas todavía se ejecutan en sistemas de bases de datos basados en los **modelos de datos jerárquicos y de red**.

* + Los DBMS se pueden clasificar basándose en el modelo de datos: **relacional, objeto, objeto-relacional, jerárquico, de red y otros.**
  + El segundo criterio que se utiliza es el **número de usuarios** soportado por el sistema. Los **sistemas de un solo usuario** solo soportan un usuario al mismo tiempo y se utilizan principalmente con los PCs. Los **sistemas multiusuarios**, que incluyen la mayoría de los DBMS’s, soportan varios usuarios simultáneamente.
  + El tercer criterio es el **número de sitios** sobre los que se ha distribuido la base de datos.
    - Un DBMS es **centralizado** si los datos están almacenados en un solo computador. Un DBMS centralizado puede soportar varios usuarios, pero el DBMS y la base de datos residen en un solo computador.
    - Un **DBMS distribuido** (**DDBMS**) puede tener la base de datos y el software distribuidos por muchos sitios conectados por una red de computadores.
    - Los **DBMS homogéneos** utilizan el mismo software DBMS en varios sitios.
    - Una tendencia reciente es desarrollar un software para acceder a varias bases de datos autónomas preexistentes almacenadas en un DBMS0s homogéneos. Esto lleva a un **DBMS federado** (o **sistema multibase de datos**) en el que los DBMS’s participantes se acoplan y tienen cierto grado de autonomía local.
  + El cuarto criterio es el coste.
  + También podemos clasificar un DBMS según los **tipos de rutas de acceso** para almacenar archivos.
  + Un DBMS puede ser de **propósito general** o de **propósito especial**.
    - Cuando la principal consideración es el **rendimiento** se puede diseñar y construir un DBMS de **propósito especial** para una aplicación específica.
    - Esto cae en la categoría de los sistemas de **procesamiento de transacciones en línea (OLTP, online transction processing).**
  + El **modelo de datos relacional básico** representa una base de datos como una colección de tablas, donde cada tabla se puede almacenar como un archivo separado.
  + El **modelo de datos de objetos** define una base de datos en termino de objetos, sus propiedades y sus operaciones. Los objetos con la misma estructura y comportamiento pertenecen a una **clase**, y las clases están organizadas en **jerarquías** (o **gráficos acíclicos**). Las operaciones de cada clase son especificas en términos de procedimientos predefinidos denominados **métodos**.
  + Los DBMS’s relacionales han ido ampliando sus modelos para incorporar conceptos de bases de datos de objetos y otras capacidades; estos sistemas se conocen como **sistemas objeto-relacional** o **sistemas relacionales extendidos**.
  + Dos modelos de datos más antiguos e históricamente importantes, ahora conocidos como modelos de datos heredados, son los modelos de red y jerárquico. El **modelo de red** representa los datos como tipos de registros y también representa un tipo limitado de relación 1:N denominado **tipo conjunto**.
  + El **modelo jerárquico** representa los datos como estructuras en forma de árboles jerárquicos.
  + El **modelo de lenguaje de marcado extendido (XML, extended markup language)** también utiliza estructuras en forma de árbol jerárquico.

**UNIDAD 3**

**Modelado de datos con el modelo Entidad-Relación (ER)**

El termino **aplicación de base de datos** se refiere a una base de datos concreta y a los programas asociados encargados de implementar las consultas y actualizaciones de la base de datos. Parte de la aplicación de base de datos requerirá el diseño, la implementación y la comprobación de esos programas de aplicación. El diseño y comprobación de los **programas de aplicación** se han considerado más como parte del dominio de la ingeniería de software que del dominio de las bases de datos.

El **modelo Entidad-Relación (ER)** es un modelo de datos **conceptual de alto nivel**. Este modelo y sus variaciones se utilizan para el diseño conceptual de las aplicaciones de base de datos.

* **Tipos de entidad, conjuntos de entidades, atributos y claves:**
  + **Entidades y atributos:**
    - **Entidades y sus atributos:** El objeto básico representado por el modelo ER es una **entidad**, que es una cosa del mundo real con una existencia independiente. Puede ser un objeto con una existencia física o puede ser un objeto con una existencia conceptual.

Cada entidad tiene **atributos** que son propiedades particulares que la describen.

* + - **Atributos compuestos frente a atributos simples (atómicos):** Los **atributos compuestos** se pueden dividir en subpartes más pequeñas, que representan atributos más básicos con significados independientes.

Los atributos que no son divisibles se denominan **atributos simples** o **atómicos.**

* + - **Atributos monovalor y multivalor:**  La mayoría de los atributos tienen un solo valor para una entidad en particular; dichos atributos reciben el nombre de **monovalor** o **de un solo valor**.

Un atributo puede tener un conjunto de valores para la misma entidad (por ejemplo, “Colores” para un coche), estos atributos se denominan **multivalor**. Estos atributos pueden tener un límite superior y uno inferior para restringir el número de valores permitidos.

* + - **Atributos almacenados y derivados:** En algunos casos dos o más valores de atributo están relacionados (por ejemplo, los atributos “Edad” y “FechaNac” de una persona). El atributo “Edad” se denomina entonces **atributo derivado** y se dice que se ha **derivado** del atributo “FechaNac” que es el denominado **atributo almacenado**.
    - **Valores NULL (nulos):** A veces es posible que una entidad no tenga un valor aplicable para un atributo. Por ejemplo, el atributo “NumApto” de una dirección solo se aplica a las direcciones correspondientes a edificios de apartamentos, y no a otros tipos de residencias, como las casas.
    - **Atributos complejos:** Los atributos compuestos y multivalor se pueden anidar arbitrariamente. Podemos representar el anidamiento arbitrario agrupando componentes de un atributo compuesto entre **paréntesis ()** y separando los componentes con comas, y mostrando los atributos multivalor entre **llaves {}.** Dichos atributos se denominan **atributos complejos**.
* **Tipos de entidades, conjuntos de entidades, claves y conjuntos de valores:**
  + **Tipos de entidades y conjuntos de entidades:** Un **tipo de entidad** define una **colección (o conjunto)** de entidades que tienen los mismos atributos.

Un tipo de entidad se representa en los diagramas ER como un **rectángulo** con el nombre del tipo de entidad en su interior. Los nombres de los atributos se encierran en **óvalos** y están unidos a su tipo de entidad mediante líneas rectas. Los **atributos compuestos** están unidos a sus atributos componente mediante líneas rectas. Los **atributos** **multivalor** se muestran en **óvalos dobles**.

Un tipo de entidad describe el **esquema** o la **intención** de un **conjunto de entidades** que comparten la misma estructura. La colección de entidades de un tipo de entidad en particular esta agrupada en un conjunto de entidades, que también se denomina **extensión** del tipo de entidad.

* + **Atributos clave de un tipo de entidad:** Una restricción importante en las entidades de un tipo de entidad es la **clave** o **restricción de unicidad** de los atributos. Un tipo de entidad normalmente tiene un atributo cuyos valores son distintos para cada entidad individual del conjunto de entidades. Dicho atributo se denomina **atributo clave**, y sus valores pueden utilizar para identificar cada entidad sin lugar a dudas.

En ocasiones, una clave está formada por varios atributos juntos, lo que da a entender que la **combinación** de los valores de atributo debe ser distinta para cada entidad.

Una clave compuesta debe ser mínima, en el atributo compuesto se deben incluir todos los atributos componente para tener una propiedad de **unicidad**.

En la notación diagramática ER cada atributo clave tiene su nombre **subrayado** dentro del ovalo.

* + **Conjunto de valores (dominós) de atributos:** Cada atributo simple de un tipo de entidad está asociado con un **conjunto de valor (**o **dominio** de valores), que especifica el conjunto de los valores que se pueden asignar a ese atributo por cada entidad individual.

Los conjuntos de valores no se muestran en los diagramas ER; normalmente se especifican mediante los **tipos de datos** básicos disponibles en la mayoría de los lenguajes de programación, como entero, cadenas, booleano, flotante, tipo enumerado, subrango, etc.

Matemáticamente, un atributo A de un tipo de entidad E cuyo conjunto de valores es V se puede definir como una **función** de E al conjunto potencia P (V) de V:

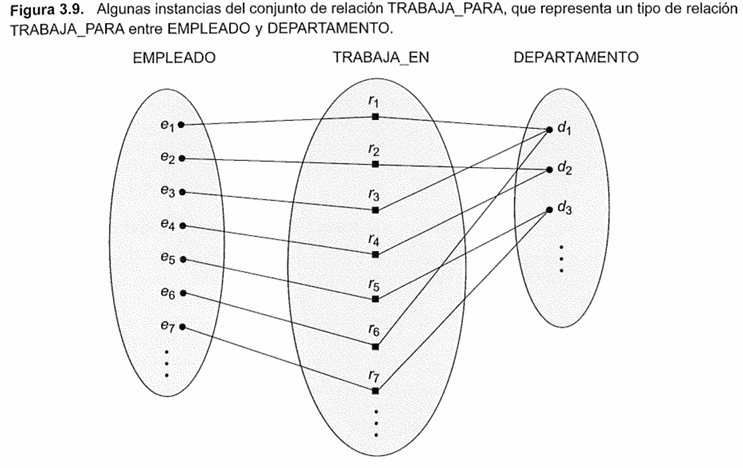
**A : E --> P(V)**

**Tipos de relaciones, conjuntos de relaciones, roles y restricciones estructurales**

* **Grado de relación, nombres de rol y relaciones recursivas:**
  + **Grado de un tipo de relación:** El **grado** de un tipo de relación es el número de tipos de entidades participantes**.**

Un tipo de relación de grado dos se denomina **binario** y uno de grado tres, **ternario**.

* + **Relaciones y atributos:** A veces es conveniente imaginar un tipo de relación en términos de atributos. Considere el tipo de relación TRABAJA\_PARAde la figura 3.9. Uno puede pensar en un atributo denominado “Departamento” del tipo de entidad EMPLEADO donde el valor de “Departamento” por cada entidad EMPLEADO es (una referencia a) la entidad DEPARTAMENTO para la que ese empleado trabaja. Por lo tanto, el conjunto de valores para este atributo “Departamento” es el conjunto de todas las entidades DEPARTAMENTO, que es el conjunto de entidades DEPARTAMENTO.



* + **Nombres de rol y relaciones recursivas:** Cada tipo de entidad que participa en un tipo de relación juega un papel o rol particular en la relación. El **nombre de rol** hace referencia al papel que una entidad participante del tipo de entidad juega en cada instancia de relación y ayuda a explicar el significado de la relación.

Los nombres de rol no son técnicamente necesarios en los tipos de relación donde todos los tipos de entidad participantes son distintos, puesto que cada nombre de tipo de entidad participante se puede utilizar como participación. Dichos tipos de relación se denominan **relaciones recursivas.**

* **Restricciones en los tipos de relaciones:**

Los tipos de relaciones normalmente tienen ciertas restricciones que limitan las posibles combinaciones entre las entidades que pueden participar en el conjunto de relaciones correspondiente. Podemos distinguir dos tipos principales de restricciones de relación:

* + **Razones de cardinalidad para las relaciones binarias:** La **razón de cardinalidad** de una relación binaria especifica el número **máximo** de instancias de relación en las que una entidad puede participar.
  + **Restricciones de participación y dependencias de existencia:** La **restricción de participación** especifica si la existencia de una entidad depende de si está relacionada con otra entidad a través de un tipo de relación. Esta restricción especifica el número **mínimo** de instancias de relación en las que puede participar cada entidad, y en ocasiones recibe el nombre de **restricción de cardinalidad mínima.**
    - Si una política de la empresa dice que **cada** empleado debe trabajar para un departamento, entonces una entidad de empleado solo puede existir si participa en al menos una instancia de relación TRABAJA\_PARA. De este modo la participación de EMPLEADO en TRABAJA\_PARA se denomina **participación total**, es decir cada entidad del **conjunto total** de entidades “empleado” debe estar relacionada con una entidad “departamento” a través de TRABAJA\_PARA.

**Diagramas ER, convenciones de denominación y problemas de diseño**

