UNIDAD DIDÁCTICA 1: CONCEPTOS BÁSICOS

En esta unidad el alumno conocerá la necesidad de las empresas de gestionar y almacenar la información y cómo han evolucionado los sistemas de información en la empresa: primero basados en ficheros y actualmente mediante el empleo de bases de datos.

Conocerá también la arquitectura y estructura interna de un sistema de gestión de base de datos, así como a distinguir entre el esquema interno, el esquema conceptual y los esquemas externos.

1. INTRODUCCIÓN

2. LA INFORMACIÓN Y SU ALMACENAMIENTO

3. SISTEMAS GESTORES DE FICHEROS

4. S.G.B.D.

5. ARQUITECTURA DE UNA BASE DE DATOS

6. COMPONENTES DE UN SGBD

1.1 INTRODUCCIÓN

Para empezar con el tema, vamos a plantear unas preguntas para poder introducir los conceptos más importantes de la asignatura:

- ¿Qué entiendes por Bases de Datos?
- ¿Conoces algún Sistema Gestor de Bases de Datos?
- ¿Has trabajado con Bases de Datos o has oído sobre ellas?
- ¿Consideras importante aprender a trabajar con Bases de Datos?

1.2. LA INFORMACIÓN Y SU ALMACENAMIENTO

Existen muchas definiciones para el significado de la palabra informática, unas más elaboradas que otras, pero todas coinciden en que **la informática es la ciencia que estudia el tratamiento automático y racional de la información**.

La información tratada puede ser volátil, esto es, información que una vez tratada no es necesario su almacenamiento y por lo tanto es desechada, se borra. O por el contrario, como ocurre **a menudo, es necesario el almacenamiento de la información para su futura recuperación, consulta y tratamiento.**

Es así como surge la necesidad de buscar cómo almacenar la información de manera que permita un tratamiento eficiente de la misma.

Cualquier organización requiere de sistemas de almacenamiento de la información para la gran cantidad de datos que manipulan diariamente. Dependiendo de la empresa, el volumen de los datos a almacenar será mayor o menor, pero no se nos escapa la necesidad de almacenamiento de información, por ejemplo, de los datos personales de los empleados de dicha empresa u organización.

Inicialmente, esta información se anotaba en tarjetas rectangulares de papel más grueso o cartón, donde podíamos anotar (dependiendo de la empresa) datos correspondientes a los clientes, facturas, y otro tipo de información.

Este volumen de información requería de una buena organización en carpetas o archivadores, cada una con una letra identificativa para que fuese más fácil localizar la información.

En clases, debatiremos cuales eran las ventajas e inconvenientes de este tipo de almacenamiento de información.

Con los avances de la tecnología, se transformó este tipo de almacenamiento en sistemas automatizados por ordenador. En un primer momento se utilizaron ficheros o archivos digitales a los que se trasladaron manualmente la información contenida en las tarjetas de papel, consiguiendo ganar en eficiencia y facilidad de almacenamiento para el trabajo diario. Como evolución al sistema de almacenamiento y organización de la

información en ficheros también surgieron los sistemas de almacenamiento de la información en bases de datos o sistemas gestores de bases de datos.

1.3. SISTEMA GESTORES DE FICHEROS

Un Sistema Gestor de Ficheros es un sistema de almacenamiento de la información en el cual, los datos se almacenan en ficheros o archivos con una estructura particular. A continuación, vamos a ver un ejemplo para clarificar conceptos.

Un ejemplo de uso de archivos sería el fichero con todos los datos de los clientes de un banco, es decir, el conjunto de registros de clientes de un banco almacenados en un dispositivo de memoria secundaria. Hemos de resaltar que los datos están almacenados de tal forma que se puedan añadir, suprimir, actualizar o consultar datos individuales en cualquier momento.

Nombre del archivo: CLIENTES
Campo clave: NIF
Formato del registro:

Campo	Nombre	Tipo de datos	Longitud
1	NIF	Alfanumérico	10
2	APELLIDOS	Alfanumérico	20
3	NOMBRE	Alfanumérico	15
4	NACIMIENTO	Fecha	8
.....			

Introducimos mediante ejemplos las diferencias en los modos de trabajar entre un sistema de gestión de ficheros y un sistema de bases de datos.

Ejemplo 1:

En un centro educativo se quieren mecanizar mediante ordenador los siguientes procesos: (a) Emisión mensual de recibos y (b) Actas de notas

En una concepción clásica de la solución del problema se crearían dos ficheros. La estructura de registro de cada uno sería:

Fichero RECIBOS
Nº Matrícula
Nombre alumno
Nombre asignatura 1
Precio asignatura 1
.....
Nombre asignatura n
Precio asignatura n
Deducciones

Fichero NOTAS
Nº Matrícula
Nombre alumno
Nombre asignatura 1
Nota asignatura 1
.....
Nombre asignatura n
Nota asignatura n

¿Qué defectos se observan en esta organización de los datos?

1º. Campos repetidos (se multiplican así las necesidades de espacio)

Nombre del alumno (está duplicado: aparece en ambos ficheros)
Precio de la asignatura (aparece repetido en cada registro del 1º fichero)
Nombre de la asignatura (aparece repetido en cada registro, en ambos ficheros)

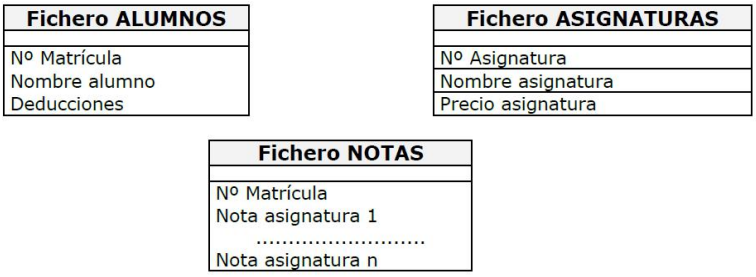
2º. Dificultades para la actualización de los datos (se invierte más tiempo).

Un mismo dato que figura en muchos ficheros deberá actualizarse en todos y cada uno de esos ficheros.

3º. Inconsistencia de datos.

Ocurre cuando, por error, se actualiza un dato en unos ficheros sí y en otros no.

¿Cómo organizaríamos los ficheros para evitar estos problemas?



Reducimos así **el número de campos repetidos** a los estrictamente imprescindibles para mantener el enlace entre los ficheros, necesario para los distintos tratamientos.

Si queremos modificar el nombre de un alumno o de una asignatura **solo necesitamos acceder a un fichero**.

Este diseño se ha hecho teniendo en cuenta las relaciones entre los datos y no la utilización de ellos en un caso particular. Se sigue así un principio básico de los Sistemas de Gestión de Bases de Datos que consiste en crear estructuras independientes de los programas en que se vayan a usar los datos.

Ejemplo 2:

Considérese una empresa bancaria que guarda la información referente a sus clientes y sus respectivas cuentas de ahorro en archivos.

Además, el sistema tiene diversos programas de aplicación que permiten, tanto a los empleados del banco como a los clientes, manejar los archivos, incluyendo:

· Un programa para hacer cargos o abonos a una cuenta.
· Un programa para añadir una nueva cuenta.
· Un programa para obtener el saldo de una cuenta.
· Un programa para generar estados mensuales.

Estos programas de aplicación los han escrito programadores en respuesta a las necesidades de la organización bancaria.

Los registros se almacenan en varios archivos, y se escribe un número de diferentes programas de aplicación para consultar, añadir, borrar o modificar registros de los archivos apropiados. Este sistema tiene un número de **desventajas** importante.

1º. Redundancia e inconsistencia de los datos.

Puesto que los archivos y los programas de aplicación son creados por distintos programadores durante un periodo de tiempo, es probable que los archivos tengan diferentes formatos y los datos pueden estar duplicados en varios sitios (archivos). Por ejemplo, *la dirección y el nº de teléfono* de un cliente determinado pueden aparecer en un archivo que contiene registros de "cuentas nómina" y en un archivo que contiene registros de cuentas de "planes de pensiones". **Esta redundancia aumenta los costes de almacenamiento y acceso**. Además, puede llevar a **inconsistencia de los datos**, esto es, las diversas copias de los mismos datos no concuerdan entre sí. Por ejemplo, una dirección cambiada de un cliente puede estar reflejada en los registros de cuentas de ahorros pero no en otro sitio del sistema.

2º. Dificultad para tener acceso a los datos.

Supóngase que uno de los gerentes del banco necesita averiguar los nombres de todos los clientes cuyo código postal sea 11567 y pide al departamento de procesamiento de datos que genere la lista correspondiente.

Puesto que esta solicitud no fue prevista cuando se diseño el sistema original, no hay ningún programa de aplicación a mano que la satisfaga, aunque sí existe uno que genera la lista de todos los clientes. El gerente tiene ahora dos elecciones: O bien coger la lista de clientes y extraer la información necesaria manualmente, o pedir al departamento de procesamiento de datos que ponga a un programador a escribir el programa de aplicación necesario.

Obviamente, ninguna de las dos alternativas es satisfactoria. Imaginemos que se escribe un programa semejante y que, varios días después, el mismo gerente necesita arreglar esa lista para incluir sólo aquellos clientes con un saldo de 10.000 euros o más. Como se esperaba, no existe un programa que genere esa lista. De nuevo, el gerente tiene las dos opciones anteriores, ninguna de las cuales es satisfactoria.

3º. Aislamiento de los datos.

Puesto que los datos están repartidos en varios archivos, y éstos pueden tener diferentes formatos, es difícil escribir nuevos programas de aplicación para obtener los datos apropiados. Además, si se han utilizado diferentes lenguajes de programación para crear esos archivos, lo más probable es que sus formatos sean incompatibles.

4º. Anomalías del acceso concurrente.

Para mejorar el funcionamiento global del sistema y obtener un tiempo de respuesta más rápido, muchos sistemas permiten que múltiples usuarios actualicen los datos simultáneamente. En un entorno así, la interacción de actualizaciones concurrentes puede dar por resultado datos inconsistentes.

Considérese una cuenta bancaria con 500 euros. Si dos clientes autorizados a manejar esa cuenta retiran fondos (digamos 50 y 100 euros, respectivamente) casi al mismo tiempo, el resultado de las ejecuciones concurrentes puede dejar la cuenta en un estado incorrecto (o inconsistente).

En particular, la cuenta puede contener 450 ó 400 euros, en vez de 350 euros. Para prevenir esta posibilidad, debe mantenerse alguna forma de supervisión en el sistema. Cuando se da una situación similar a la expuesta en el anterior ejemplo, donde se puede acceder a los datos por medio de diversos programas de aplicación que no han sido previamente coordinados, esta verificación es muy difícil de proporcionar.

5º. Problemas de seguridad.

No todos los usuarios del sistema de gestión informático deben poder acceder a todos los datos. Por ejemplo, en un sistema de gestión de un centro educativo cada profesor sólo debe poder acceder a las notas de sus alumnos, y por otra parte, cada alumno podrá consultar sus notas pero no podrá modificarlas. Puesto que los programas de aplicación se añaden al sistema de una forma imprecisa, es difícil implantar tales restricciones de seguridad.

6º. Problemas de integridad.

Los valores de datos almacenados en la base de datos deben satisfacer ciertos tipos de restricciones de consistencia. Por ejemplo, el saldo de una cuenta bancaria nunca debe caer por debajo de una cantidad prescrita (digamos, 25 euros). Estas restricciones se hacen cumplir en el sistema añadiendo códigos apropiados en los diversos programas de aplicación. Sin embargo, cuando se añaden restricciones nuevas es difícil cambiar los programas para hacerlas cumplir. El problema se complica aún más cuando las restricciones implican varios elementos de información de distintos archivos.

7º. Dependencia de los datos física-lógica.

La estructura física de los datos (definición de archivos y registros) se encuentra codificada en los programas de aplicación. Cualquier cambio en esa estructura implica que el programador deberá identificar, modificar y probar todos los programas que manipulan esos archivos.

1.4. SISTEMAS GESTORES DE BASES DE DATOS

Una Base de Datos es una colección de datos interrelacionados, almacenados en conjuntos sin redundancias perjudiciales o innecesarias, cuya finalidad es la de servir a una o más aplicaciones de la mejor manera posible.

Un Sistema de BD es un sistema mecanizado por ordenador para manejar datos. Este manejo se hace a través de paquetes de Software, que reciben distintos nombres según la traducción realizada del nombre en inglés (puede ser DBMS, SGBD o SMBD).

El SGBD se puede definir como una aplicación que permite a los usuarios definir, crear y mantener la base de datos y proporciona acceso controlado a la misma. Es una herramienta que sirve de interfaz entre el usuario y la base de datos.

La función del SBD es almacenar la información y tenerla disponible para la empresa y para los usuarios que la usan (cada empresa almacena su propia información de la manera más conveniente a sus intereses).

El SBD estaría formado por:

I) LOS DATOS

- Datos Persistentes o Fijos: Aquellos que están de manera indefinida en la BD.
- Datos de Entrada: Aquellos que introducimos por primera vez en la BD, bien desde el teclado, transportándolos desde un fichero, ...
- Datos de Salida: Aquellos que sacamos bien por la terminal, por la impresora, a un fichero,... y que pueden ser: de la propia Base de Datos o Estadísticos.

II) EL HARDWARE

Incluye todos los dispositivos físicos usados.

III) EL SOFTWARE

Está compuesto por un conjunto de programas (SGBD), que nos ayudan como interfaces para el tratamiento de datos. Éste se encarga de atender nuestras peticiones y consultas, intentar eliminar fallos, etc.

IV) LOS USUARIOS

Se pueden clasificar en los siguientes:

- Programadores de las aplicaciones: Conjunto de personas que hacen aplicaciones y se dedican a su mantenimiento.
- Usuarios finales: Personas que usan la BD bien directamente (con un lenguaje de consulta) o bien mediante programas. Sólo hacen preguntas o modificaciones en los datos.
- Administrador de Datos (DA): Donde existe una BD hay una persona o grupo de personas que lleva el control de los datos. No tiene por que saber nada de informática, y normalmente suele tener un cargo alto en la empresa, y toma la decisión de 'Que datos se van a almacenar'.
- Administrador de la BD (DBA): Puede ser una o varias personas. Se encarga del control y la administración de la BD. Decide como almacenar los datos que le diga el Administrador de Datos

Ventajas del enfoque de la BD:

1. Independencia de los datos respecto de las aplicaciones

El usuario tiene una visión abstracta de los datos, sin necesidad de ningún conocimiento sobre la implementación de los ficheros de datos, índices, etc. Esto supone un gran ahorro en los costes de programación, de forma que la modificación de la estructura de los datos no suponga un cambio en los programas y viceversa. Sin ella, el mantenimiento de la base de datos ocuparía el 50% de los recursos humanos dedicados al desarrollo de cualquier aplicación.

2. Disminuye la redundancia de los datos.

Seguramente pensáis que el problema de la redundancia es el espacio perdido. Antiguamente, cuando el precio del byte de disco era muy elevado, esto era un problema grave, pero actualmente prácticamente nunca lo es. ¿Qué problema hay entonces? Simplemente, lo que todos hemos sufrido más de una vez: si tenemos algo apuntado en dos lugares diferentes no pasará demasiado tiempo hasta que las dos anotaciones dejen de ser coherentes, porque habremos modificado la anotación en uno de los lugares y nos habremos olvidado de hacerlo en el otro.

Así pues, el verdadero problema es el grave riesgo de inconsistencia o incoherencia de los datos; es decir, la pérdida de integridad que las actualizaciones pueden provocar cuando existe redundancia.

Por lo tanto, convendría evitar la redundancia. En principio, nos conviene hacer que un dato figure una vez en la BD. Sin embargo, esto no siempre será cierto. Por ejemplo, el disponer de réplicas de los datos por razones de fiabilidad, disponibilidad o coste de comunicaciones.

3. Mayor integridad de datos

Aparte de por la redundancia, se puede perder la corrección o consistencia de los datos por muchas otras razones: errores de programas, errores de operación humana, avería de disco, transacciones incompletas por corte de alimentación eléctrica, etc.

El SGBD tiene reglas de integridad para asegurar la consistencia de los datos.

En caso de errores o desastres, también podríamos perder la integridad de los datos. El SGBD nos debe dar las herramientas para reconstruir o restaurar los datos estropeados.

Cuando el SGBD detecte que un programa quiere hacer una operación que va contra las reglas establecidas al definir la BD, no se lo deberá permitir, y le tendrá que devolver un estado de error.

4. Mayor disponibilidad de datos: Permite recuperar las bases de datos en caso de que ocurra algún suceso imprevisto que afecte o destruya la base de datos.

5. Compartición de datos: Los datos deben poder ser accedidos por varios usuarios simultáneamente, teniendo previstos procedimientos para salvaguardar la integridad de los mismos.

Actualmente ya no son raros los Sistemas de Información que tienen, en un instante determinado, miles de sesiones de usuario abiertas simultáneamente. Solo hace falta pensar en los típicos sistemas de los consorcios de compañías aéreas o casos similares, e incluso en los servidores de páginas web.

6. Mayor seguridad de los datos

Los SGBD permiten definir autorizaciones o derechos de acceso a diferentes niveles. Estos mecanismos de seguridad requieren que el usuario se pueda identificar. Se acostumbra a utilizar códigos de usuarios acompañados de contraseñas, pero también se usan tarjetas magnéticas, reconocimiento de voz, etc.

Desventajas del enfoque de la BD:

Bueno, viendo el punto anterior podríamos hacernos la siguiente pregunta:

Con todas las ventajas que tiene el usar sistemas de bases de datos frente a sistemas de archivos, ¿por qué utilizar estos últimos?

Pues la contestación es que hasta ahora no existe nada perfecto en informática, y el uso de los sistemas de bases de datos tiene también sus desventajas que pasamos a analizar.

Podríamos destacar las siguientes:

■ Desventajas relativas a la implantación:

- Instalación costosa en equipos y software.
- Instalación larga y difícil.
- Falta de rentabilidad a corto plazo.

■ Desventajas relativas a los usuarios:

- Necesidad de formación de un personal especializado.

Entonces, ¿cuándo usar un sistema de archivos y cuando un sistema de bases de datos a la hora de plantearnos una aplicación?

Generalizando, usaremos el sistema de archivos cuando la cantidad de datos a guardar sea tan reducida que no justifique las ventajas del uso de los sistemas de bases de datos.

Un ejemplo típico sería realizar una aplicación de agenda en la cual sólo hay que guardar campos de texto ordenados por hora, día, mes y año. Aquí no sería recomendable el uso de un SGBD.

1.5. ARQUITECTURA DE UNA BASE DE DATOS

Uno de los objetivos de un sistema de base de datos es *proporcionar una visión lo más abstracta posible de la información*, es decir, ocultar detalles referentes a la forma en que los datos están organizados y almacenados, proporcionar un fácil acceso a los datos por parte de los usuarios sin tener en cuenta el formato en que se encuentren los datos. El otro objetivo, que será explicado a posteriori, es la independencia de datos con respecto de las aplicaciones.

La arquitectura más estándar y, por tanto, la más utilizada, es la que hace una división en niveles de la base de datos. *Se consideran tres niveles* según la perspectiva desde la que sea vista la información.

- ***Nivel interno***. Es la representación del nivel más bajo de abstracción. Se especifican los archivos que contienen la información, su organización, la forma de acceder a los registros, el tipo y longitud del registro, los campos que lo componen, los campos clave, etc, es decir, *como la información es almacenada en los dispositivos de almacenamiento.*

- ***Nivel conceptual***. Es la visión o representación del problema tal y como éste se presenta en el mundo real. Describe qué datos son almacenados realmente en la base de datos y las relaciones que existen entre los mismos. Se obtiene a partir de los requerimientos de los usuarios potenciales del sistema de base de datos a implantar, sin importar la forma ni el lugar en el

que se almacenarán y recuperarán los datos. Contiene los datos elementales (campos), los datos compuestos (registros), las relaciones entre campos elementales y compuestos, las reglas que rigen el funcionamiento de la empresa, etc. Esta visión no cambia al menos que cambie la naturaleza del problema.

- ***Niveles externos***. Son el conjunto de percepciones individuales de la base de datos. Cada visión individual se denomina *subesquema* o *vista*. Un subesquema podrá ser compartido por varios usuarios, y cada usuario tendrá la posibilidad de acceder a distintos subesquemas. Al crear un subesquema, es posible mezclar campos de distintos registros, omitir campos, cambiar el orden de los campos, añadir campos que puedan ser calculados a partir de los descritos en el esquema conceptual, etc.

  Estas "visiones particulares" de los usuarios son proporcionadas por los procedimientos o programas de aplicación que sólo manejan parte de la información de la base de datos.

ESQUEMA EXTERNO

Subesquema 1:

ALUMNOS CON UNA EDAD DETERMINADA: AL_DNI, AL_EDAD

Subesquema 2:

ASIGNATURAS APROBADAS POR UN ALUMNO : AL_DNI, ASIG_NUM, NOTA

ESQUEMA CONCEPTUAL

ALUMNOS : AL_DNI, AL_NOM, AL_APELL,...
ASIGNATURAS : ASIG_NUM, ASIG_NOM, ASIG_HORAS
NOTAS : AL_DNI, ASIG_NUM, NOTA, FECHA

Reglas :
1. Un alumno tiene exactamente ocho asignaturas.
2. No puede haber más de treinta alumnos por asignatura
3. .....

ESQUEMA INTERNO

ARCHIV	ORGANIZ.	CLAVE	LONG.REG.	CAMPOS	TIPO_DATOS
Alumnos	Indexada	AL_DNI	70	AL_DNI	X(9)
				AL_NOM	X(15)
				AL_APELL	X(20)
				.....	

Independencia de los datos

Es la capacidad para modificar el esquema de un nivel del sistema de la base de datos sin tener que modificar el esquema del nivel inmediato superior.

Podemos de definir dos tipos de independencia de los datos:

- Independencia lógica
- Independencia física

Independencia lógica de los datos

Entendemos por independencia lógica de datos que:

- Los cambios que hagamos en el esquema lógico no deben afectar a los esquemas externos que no utilicen los datos modificados.

Es decir, es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación.

Esto es, podemos modificar el esquema conceptual por ejemplo, para ampliar la base de datos (añadiendo un nuevo tipo de registro o un elemento de datos) o para reducir la base de datos (eliminado un tipo de registro o un elemento de datos).

Esta modificación en el esquema conceptual no deberá afectar a los esquemas externos que sólo se refieran a los datos restantes.

Por ejemplo, el hecho de suprimir el atributo *fecha de nacimiento* de la entidad *alumno* y añadir otra entidad aula no debería afectar a ninguno de los programas existentes que no utilicen el atributo *fecha de nacimiento.*

Otros ejemplos:

1) El personal administrativo de secretaría podría tener una visión de la entidad *alumno* sin que fuese necesario que existiese el atributo *nota*. Sin embargo, los usuarios profesores (o los programas dirigidos a ellos) podrían tener una visión en la que existiese el atributo *nota* pero no el atributo *fecha de pago*.
2) Decidimos ampliar la longitud del atributo *nombre* y lo aumentamos de treinta a cincuenta caracteres, pero no sería necesario modificar los programas que ya tenemos escritos si no nos importa que los valores obtenidos tengan sólo los primeros treinta caracteres del nombre.

Independencia física de los datos

Por independencia física de los datos entendemos que:

- El esquema lógico no se ve afectado por cambios realizados en el esquema interno, correspondientes a modos de acceso, etc. Es decir, es la capacidad de

modificar el esquema interno sin tener que alterar el esquema conceptual (o los externos).

Un ejemplo:

Puede ser necesario modificar el esquema interno de los datos por la necesidad de reorganizar ciertos ficheros físicos (por ejemplo, al crear estructuras de datos adicionales) con el fin de mejorar el rendimiento de las operaciones de recuperación y actualización. Si la base de datos contiene aún los mismos datos, no será necesario modificar el esquema conceptual.

Para conseguir esta independencia, tanto los usuarios que hacen consultas (o actualizaciones) directas como los profesionales informáticos que escriben programas que las lleven incorporadas, deben poder desconocer las características físicas de la BD con que trabajan. No necesitan saber nada sobre el soporte físico, ni estar al corriente de qué SO se utiliza, qué índices hay, la compresión o no compresión de datos, etc. De este modo se pueden hacer cambios de tecnología y cambios físicos para mejorar el rendimiento sin afectar a nadie.

1.6 COMPONENTES DE UN SGBD

El SGBD está dividido en módulos que llevan a cabo sus funciones asociadas. Se compone del núcleo, lenguaje, utilidades y diccionario de datos.

1) **Núcleo:** Es el conjunto de programas que coordinan y controlan el funcionamiento del SGBD. Son programas transparentes al usuario.

   - Controlan la integridad y seguridad.
   - Implementan las funciones de comunicación entre niveles.
   - Facilitan la independencia de los datos.
   - Gestionan el diccionario de datos.
   - Proporcionan el soporte necesario para los programas de utilidad y los lenguajes.

2) **Lenguajes:** El SGBD proporciona lenguajes que permiten la definición y el manejo de los datos de la base. Cada SGBD tiene una estructura y organización particular, pero todos tienen la característica común de ofrecer al administrador y a los usuarios tres lenguajes:

   - *El lenguaje de descripción de datos* (DDL) se utiliza para definir el esquema conceptual y los distintos subesquemas externos de la base de datos. Además, ofrece la posibilidad de establecer parte de la seguridad de la base de datos, permitiendo asignar derechos sobre las operaciones que pueden realizar los usuarios.
   - *El lenguaje de manipulación de datos* (DML) es el encargado de gestionar la información de la base de datos. Permite añadir, eliminar y modificar registros, y recuperar información de forma

estructurada de la base de datos.

- *El lenguaje de control de datos* (DCL) que se utiliza para conceder o revocar permisos a los distintos usuarios sobre los objetos de la base de datos

3) Utilidades: Son aplicaciones que facilitan el trabajo a los usuarios y programadores. Tiene la característica común de tener un interfaz fácil de entender. Se basan en menús que guían al usuario para conseguir el objetivo final. Por ejemplo, asistentes o generador de formularios.

4) Diccionario de datos: Es un almacén integrado en el que se almacena toda la información referente a la descripción, gestión e implantación de la base de datos. También se conoce como "*catálogo del sistema"*.