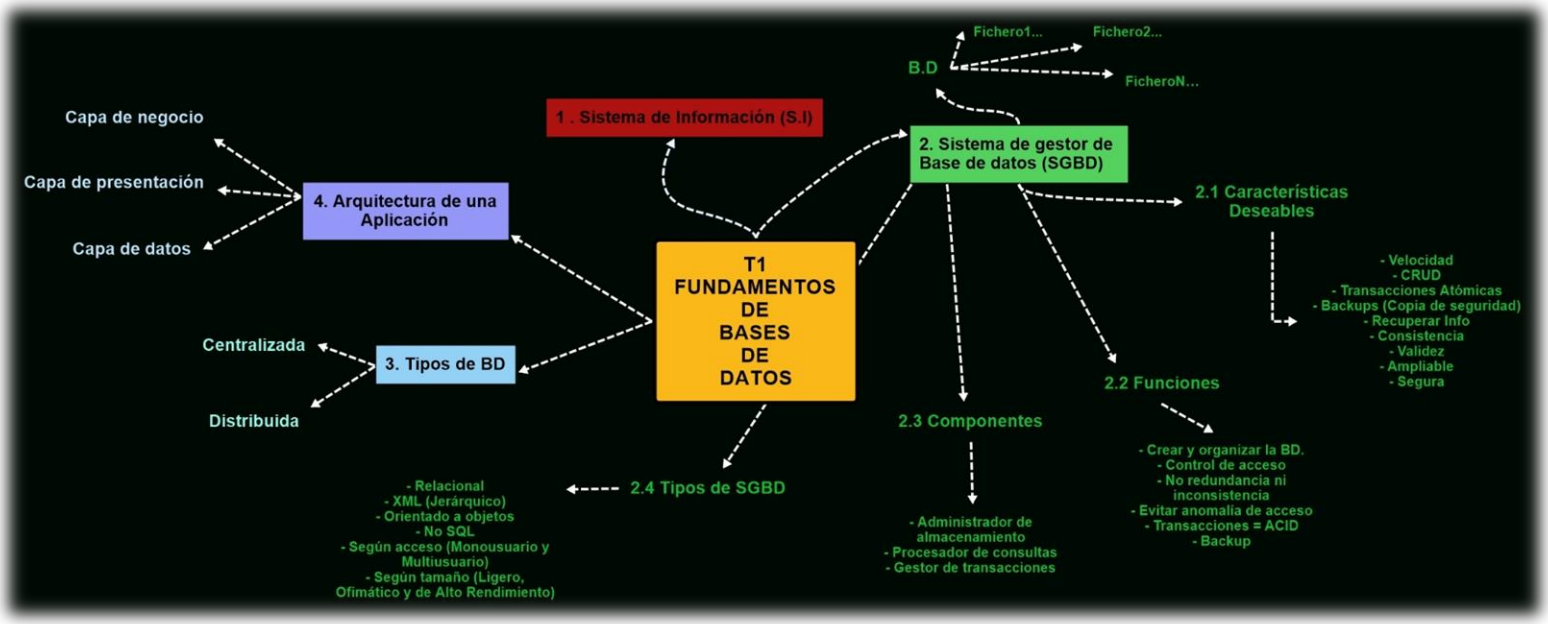
# Bases de Datos

## *T.1 Fundamentos de Bases de Datos*

### *1. Sistema de Información*




Un Sistema de Información es un conjunto de elementos diseñados para procesar y administrar datos e información con el propósito de satisfacer una necesidad específica. Esta definición engloba sistemas de todo tipo, independientemente de si incluyen componentes informáticos. En este curso, nos enfocamos en los Sistemas de Información informáticos, específicamente en los Sistemas Gestores de Bases de Datos (SGBD), que se definirán más adelante.

Es importante notar que en el campo de la informática existen diversos tipos de Sistemas de Información además de los SGBD. Por ejemplo, cualquier aplicación informática que maneje grandes cantidades de información para cualquier propósito se considera un Sistema de Información. Es probable que muchas de estas aplicaciones utilicen un SGBD como plataforma de almacenamiento de datos.

En este curso, se enseñará cómo desarrollar aplicaciones que se conectan con Bases de Datos para gestionar información, lo que equivale a crear nuestros propios Sistemas de Información. Esto se logrará mediante el uso de herramientas de desarrollo y la utilización de un SGBD como respaldo para nuestros datos.

# *2. Sistema gestor de Bases de Datos*

Un Sistema Gestor de Bases de Datos (SGBD) es una herramienta de software que permite crear y administrar bases de datos de manera eficiente. El SGBD se encarga de organizar los archivos que componen la base de datos para asegurar que la información sea siempre accesible de forma eficiente en términos de espacio y velocidad.

Al utilizar un SGBD, se añade un nivel de abstracción, lo que significa que los usuarios no necesitan preocuparse por los detalles técnicos de cómo se almacenan, dónde se almacenan o cuánto espacio ocupan los datos. La herramienta se encarga de mantener la consistencia de la base de datos, garantizar su disponibilidad y realizar otras tareas, que pueden variar en complejidad según el SGBD específico.

Es importante destacar que existen diferentes tipos de SGBD, algunos requieren conocimientos técnicos avanzados y están diseñados para respaldar aplicaciones web y sistemas de información que manejan grandes cantidades de datos, como MySQL. Por otro lado, existen SGBD como Microsoft Access o LibreOffice Base que son más accesibles para usuarios menos técnicos y están diseñados para funcionar de forma independiente como aplicaciones finales de usuario, siendo incluidos en suites ofimáticas populares.

## 2.1 Características deseables de un SGBD.

En cualquier Sistema de Información, se esperan una serie de características, que se exponen a continuación:

- CRUD (Crear, Leer, Actualizar, Eliminar): El SGBD debe permitir estas cuatro operaciones básicas para gestionar los datos en la base de datos.

- Recuperación de la información: Debe posibilitar la búsqueda y recuperación de datos de diversas formas, permitiendo consultas basadas en múltiples criterios.

- Consistencia: Los resultados de las consultas deben ser coherentes, garantizando que una búsqueda repetida produzca el mismo resultado, siempre que los datos no hayan cambiado.

- Validez: Debe ofrecer mecanismos para validar la información ingresada, asegurando que los datos cumplan con el formato correcto.

- Velocidad: Debe ser eficiente en la creación, modificación, lectura y eliminación de datos, influyendo en gran medida el diseño de la base de datos.

- Transacciones atómicas: Debe admitir operaciones como transacciones indivisibles, ejecutándose completamente o no ejecutándose en absoluto.

- Persistencia y soporte para Backups: La información debe ser persistente y protegida contra pérdidas, con la capacidad de respaldar y recuperar datos en caso de fallas o accidentes.

- Capacidad de extensión: Debe permitir la incorporación de nuevas características a medida que evoluciona el tiempo, manteniendo el rendimiento mediante un diseño adecuado.

- Seguridad: Debe ofrecer mecanismos de seguridad para proteger la información almacenada, incluyendo la gestión de cuentas de usuario y privilegios de acceso. La seguridad debe ser proporcional a la sensibilidad de los datos.

## 2.2 Funciones

Las funciones de todo SGBD son las siguientes:

- Creación y organización de la Base de Datos: El SGBD permite crear y gestionar la base de datos sin que los usuarios necesiten conocer su estructura interna.

- Control de acceso: La mayoría de los SGBD brindan control de acceso mediante cuentas de usuario y privilegios, permitiendo al administrador restringir el acceso y las acciones de los usuarios en objetos específicos.

- Evitar redundancia e inconsistencia de datos: El SGBD previene la duplicación de datos y la inconsistencia al asegurar que los datos no se actualicen de manera incorrecta cuando existen copias duplicadas.

- Evitar anomalías en el acceso concurrente: Los SGBD con soporte para múltiples usuarios resuelven problemas relacionados con la modificación simultánea de datos por varios usuarios.

- Garantizar la correcta ejecución de las transacciones: Los SGBD transaccionales siguen el estándar **ACID**, que garantiza que las transacciones sean atómicas (completas o nulas), consistentes (cumplen las reglas de integridad), aisladas (independientes entre sí) y duraderas (persisten en caso de fallo).

- Respaldo y recuperación: La mayoría de los SGBD ofrecen herramientas para realizar copias de respaldo de la base de datos, lo que permite recuperar datos en caso de fallos.

Estas funciones son esenciales para la gestión eficiente y segura de bases de datos en entornos informáticos.

No todos los Sistemas Gestores de Bases de Datos (SGBD) incorporan todas las funcionalidades mencionadas anteriormente, ya que su implementación depende de las necesidades específicas. Por ejemplo, Microsoft Access no ofrece soporte para acceso concurrente ni soluciona ciertos problemas de seguridad y control de acceso, ya que no está diseñado para entornos complejos, sino como parte de un paquete de oficina. En contraste, SGBD como Oracle, SQL Server y MySQL proporcionan sólido soporte para acceso concurrente, seguridad y control de acceso, dirigidos a entornos más complejos.

Algunos SGBD solo implementan algunas de estas características, diseñados para entornos con poca capacidad de cómputo, como dispositivos móviles. Por ejemplo, Android utiliza SQLite debido a su tamaño reducido y bajos requisitos de procesamiento.

## 2.3 Componentes

En la mayoría de los Sistemas Gestores de Bases de Datos (SGBD), existe una clara separación entre los componentes principales, siguiendo una arquitectura Cliente-Servidor. La mayor parte de la funcionalidad reside en el lado del servidor, donde se almacenan los datos. El lado cliente actúa como una herramienta de comunicación entre el usuario y el Motor (Engine) del SGBD, que forma la parte del servidor.

Dentro del motor de un Sistema Gestor de Bases de Datos (SGBD), se encuentran tres componentes clave:

- Administrador de Almacenamiento: Controla el acceso a la información almacenada en el disco, gestionando los buffers y los archivos de datos.

- Procesador de consultas: Recibe solicitudes de consultas y determina la forma más eficiente de ejecutarlas mediante un Plan de Ejecución. Coordina la obtención de datos con el Administrador de Almacenamiento.

- Gestor de transacciones: Garantiza la integridad de la Base de Datos después de la ejecución de transacciones, ya sean exitosas o fallidas.

## 2.4 Tipos de SGDB

Los SGBD se clasifican en diferentes tipos según el modelo de datos que utilizan. Algunos ejemplos incluyen SGBD orientados a objetos y SGBD NoSQL. Es importante notar que una Base de Datos puede pertenecer a múltiples categorías al mismo tiempo. Por ejemplo, db4o es un SGBD que es tanto orientado a objetos como NoSQL, ya que no emplea SQL para acceder a sus datos almacenados.

| *Tipo* | *Definición* | *Funcionalidades:* | *Apropiadas para:* | *Inapropiadas para:* | *Ejemplos:* |
|---|---|---|---|---|---|
| **Relacional** | Siguen el modelo de datos relacional, que es ampliamente utilizado y se basa en tablas y relaciones entre ellas. | • **Tipos de datos:** Cada columna en una tabla tiene un tipo de dato definido, lo que impide que se almacenen valores de otro tipo en esa columna.<br>• **Restricciones:** Es posible establecer restricciones en las columnas para imponer requisitos específicos a los valores almacenados en ellas.<br>• **Integridad referencial:** El SGBD garantiza que, al registrar nuevos datos relacionados con otros, exista una referencia válida en la tabla relacionada antes de permitir el registro.<br>• **Consultas complejas:** Los SGBD relacionales permiten realizar consultas complejas, incluso aquellas que no se habían considerado inicialmente durante el diseño de la Base de Datos. | Cuando se requieren consultas complejas entre tablas, validación de datos entre tablas, datos con múltiples valores y la necesidad de crear nuevas consultas no planificadas previamente. | Cuando se necesita una topología específica o en situaciones de alto rendimiento donde las reglas de los SGBD relacionales puedan afectar al rendimiento global. | • ***Ms Access***<br>• ***Oracle***<br>• ***Microsoft SQL Server***<br>• ***MySQL*** |
| **XML** | Es un lenguaje para almacenar información de manera jerárquica. Aunque no ofrece funcionalidades para crear, buscar, modificar o validar datos, se utiliza ampliamente para almacenar y transferir datos jerárquicos. | Se limita a definir cómo estructurar el fichero y organizar la información utilizando etiquetas que indican el significado de los valores almacenados. La jerarquía entre estas etiquetas establece la relación entre los datos. Para trabajar con archivos XML de manera efectiva, es necesario utilizar herramientas específicas que se adapten a las necesidades particulares de gestión y procesamiento de estos archivos. | • Cuando los datos son jerárquicos<br>• Cuando se requieren funcionalidades específicas de XML.<br>• Cuando se utilizan aplicaciones que comprenden formato XML. | Cuando los datos no son jerárquicos, se necesitan validaciones complicadas, cuando se requiera relacionar información o la Base de Datos es demasiado grande para reescribir el archivo completo con cada cambio. | • ***Xbird***<br>• ***XQuery*** |

| *Tipo* | *Definición* | *Funcionalidades:* | *Apropiadas para:* | *Inapropiadas para:* | *Ejemplos:* |
|---|---|---|---|---|---|
| **Orientado a objetos** | Representan todos los elementos de la Base de Datos como objetos, siguiendo el paradigma de la Programación Orientada a Objetos. | Están diseñadas para trabajar en conjunto con lenguajes de programación orientados a objetos, permitiendo que los objetos de la aplicación se correspondan directamente con los objetos en la Base de Datos, utilizando el mismo modelo de datos. | • Cuando se está desarrollando una aplicación con un lenguaje orientado a objetos<br>• No se necesitan consultas muy complejas. | • Cuando es necesario interactuar con otras herramientas que utilizan modelos más comunes como el relacional.<br>• Cuando se requieren consultas complejas.<br>• Cuando la aplicación no utiliza un lenguaje orientado a objetos.<br>• Cuando se necesitan validaciones que este tipo de bases de datos no ofrece. | • ***db4o***<br>• ***InterSystems Caché.*** |
| **Objeto - Relacional** | Son una extensión del modelo relacional, lo que significa que almacenan datos en tablas. | Permiten el uso de características de la Programación Orientada a Objetos (POO) como la herencia, o tipos complejos como colecciones y campos estructurados. | • Cuando la arquitectura de la aplicación está orientada a objetos.<br>• Cuando se necesitan realizar consultas complejas de datos relacionados entre sí.<br>• Cuando es necesario validaciones de datos.<br>• Cuando se mantiene una separación entre los programadores de la aplicación y la base de datos. | • Cuando la aplicación no utiliza un lenguaje orientado a objetos como parte de su desarrollo. | • ***PostgreSQL*** |
| **NoSQL** | Representan un cambio de tendencia en comparación con los SGBD relacionales tradicionales. No utilizan SQL como su principal lenguaje de consultas y no admiten operaciones JOIN como en los sistemas relacionales y por lo general, no garantizan ACID. | Destacan por su escalabilidad horizontal, lo que significa que pueden mantener el rendimiento al agregar nodos al sistema. | • Cuando se necesita manejar grandes volúmenes de datos.<br>• Cuando es necesario escalar el sistema agregando más nodos. | • No son apropiadas para todos los proyectos, ya que el rendimiento no siempre es la principal consideración y pueden carecer de reglas de validación, coherencia o garantía ACID. | • ***MongoDB***<br>• ***Cassandra*** |

| *Tipo* | *Nombre* | *Definición* | *Funcionalidades:* | *Ejemplos:* |
|---|---|---|---|---|
| **Según accesos simultáneos** | **Monousuario** | Son sistemas que no permiten que más de un usuario se conecte a la BD al mismo tiempo. | Carecen de controles de usuario y la Base de Datos se reduce a un simple archivo que la aplicación puede acceder como un documento estándar. Estos SGBD no admiten el control de concurrencia ni otras características propias de los sistemas multiusuarios. | • ***Ms Access***<br>• ***LibreOffice Base*** |
| | **Multiusuario** | Son sistemas que permiten múltiples conexiones simultáneas a una misma BD. | Estos sistemas ofrecen soporte para el control de cuentas de usuario y, por lo tanto, incluyen todas las características necesarias para gestionar la concurrencia en un entorno multiusuario. | • ***MySQL,***<br>• ***PostgreSQL,***<br>• ***Microsoft SQL Server***<br>• ***Oracle***<br>• ***MongoDB***<br>• ***Cassandra*** |
| **Según tamaño** | **SGBD ligero** | Es un SGBD de tamaño reducido, que dan soporte para algunas funciones principales de estas herramientas | Son ideales para entornos donde el rendimiento no es una prioridad o donde la capacidad de procesamiento es limitada, como en dispositivos móviles. | • ***SQLite*** |
| | **SGBD Ofimática** | Son un poco más potentes que los SGBD ligeros y ofrecen algunas funciones adicionales. | Son comúnmente utilizados en aplicaciones de ofimática para gestionar bases de datos pequeñas, generalmente ocupando varios megabytes de espacio. | • ***Microsoft Access***<br>• ***LibreOffice Base.*** |
| | **SGBD de alto rendimiento** | Son ideales para entornos dedicados a la gestión de datos y aplicaciones que manejan grandes volúmenes de información. | Ofrecen funciones avanzadas como control de usuarios, manejo de concurrencia, soporte para transacciones y otras operaciones. | • ***MySQL***<br>• ***Microsoft SQL Server***<br>• ***Oracle***<br>• ***PostgreSQL*** |

## *3. Tipos de Bases de Datos*

- **Bases de Datos centralizadas**

Las Bases de Datos centralizadas son comunes en grandes corporaciones, donde todos los datos se almacenan y gestionan en un único equipo con un SGBD. Esto ofrece ventajas como una fácil implementación y un diseño sencillo debido a su amplia adopción.
Sin embargo, una desventaja importante es que cualquier fallo en el equipo central afectará a todos los datos almacenados en la Base de Datos, ya que todo está ubicado en un único lugar físico.

- **Bases de Datos distribuidas**

Una Base de Datos distribuida consta de múltiples bases de datos relacionadas, ubicadas en diferentes lugares físicos. Estas bases de datos pueden operar de manera autónoma o de forma distribuida, lo que permite a los usuarios acceder a toda la información como si estuviera en un solo lugar, incluso a través de una red. Las ventajas incluyen un menor costo al crear una red de máquinas más económicas en comparación con una sola máquina poderosa, mayor disponibilidad ya que un fallo en un nodo no afecta a los demás, y la capacidad de ubicar datos donde más se necesiten.

Sin embargo, las desventajas incluyen la complejidad en la implementación y mantenimiento, desafíos de seguridad debido a múltiples nodos y falta de experiencia en situaciones de fallos poco comunes. También puede requerir más mano de obra especializada en cada nodo.

# *4. Arquitectura de una aplicación*

La posición de un Sistema de Gestión de Bases de Datos (SGBD) y la Base de Datos en una aplicación se ve influenciada por la arquitectura cliente-servidor y la programación por capas. La programación por capas busca separar las diferentes capas lógicas de una aplicación, que suelen ser presentación, negocio y datos. La arquitectura más común es la de 3 capas, donde se busca una separación lógica de estas capas, aunque no siempre requiere una separación física. La separación física dependerá de la naturaleza de la aplicación o las necesidades de rendimiento, como en el caso de aplicaciones web o la distribución de capas en uno o varios equipos para respaldar la capa de negocio o datos.

- **La capa de presentación** es la interfaz que el usuario ve y con la que interactúa en una aplicación. Su función principal es presentar la aplicación al usuario, permitir la interacción y mostrar la información solicitada por el usuario.

- **La capa de negocio** es donde reside la lógica de una aplicación. En esta capa, se llevan a cabo todos los procesos de tratamiento de la información que se obtiene de la capa de datos. Los resultados de estos procesos se presentan al usuario a través de la capa de presentación.

- **La capa de datos** es donde se almacenan los datos y donde se interactúa con los (SGBD). La capa de negocio envía peticiones para acceder a los datos almacenados en esta capa, y los SGBD procesan estas solicitudes y devuelven los resultados a la capa de negocio para su posterior procesamiento y presentación en la capa de presentación.

En aplicaciones de tamaño medio, es común tener un modelo lógico de 3 capas, pero en la implementación física, a menudo se distribuyen en solo 2 capas. En estos casos, la capa de presentación suele estar en el equipo del usuario, mientras que las capas de negocio y datos se encuentran en un servidor remoto. Conforme la aplicación se vuelve más exigente, las capas de negocio y datos pueden separarse en diferentes equipos, generalmente físicamente más cercanos, y se pueden agregar más equipos a cualquiera de estas capas para mejorar el rendimiento según sea necesario.

# *ANEXO 1: Las BD en la producción de software*

## Los lenguajes de bases de datos

Se utilizan para facilitar la comunicación entre el usuario y la base de datos, permitiéndole expresar sus necesidades de manera sencilla. Esto se logra mediante la creación de lenguajes que se asemejan al lenguaje natural y que brindan comandos abstractos para interactuar con la base de datos, con el objetivo de que el usuario se enfoque en sus problemas en lugar de preocuparse por la forma de expresarlos. Estos lenguajes buscan ser lo más parecidos posible al lenguaje natural en términos de léxico, sintaxis y semántica, para ofrecer la máxima flexibilidad en la comunicación con la base de datos.

Los lenguajes de bases de datos se dividen en tres categorías según su función:

- **DDL (Lenguaje de Definición de Datos)**: Comandos usados por administradores para crear y modificar la estructura de la base de datos. Ejemplos: *CREATE*, *ALTER*, *DROP*.

- **DML (Lenguaje de Manipulación de Datos)**: Comandos utilizados por usuarios avanzados para realizar operaciones en los datos, como agregar, eliminar, actualizar y consultar. Estos comandos pueden integrarse en otros lenguajes de programación utilizados para desarrollar aplicaciones. Ejemplos: *SELECT*, *INSERT*, *DELETE*, *UPDATE*.

- **DCL (Lenguaje de Control de Datos)**: Comandos usados por administradores para gestionar el acceso y permisos a los datos. Ejemplos: *GRANT* (concesión de permisos) y *REVOKE* (revocación de permisos).

## Las bases de datos en la producción de software

Las bases de datos deben ser parte integral del ciclo de vida de la ingeniería del software, que comprende etapas como ***análisis***, ***diseño***, ***desarrollo*** y ***explotación***. En este proceso, se definen actividades para garantizar coherencia con otros sistemas y establecer controles de gestión del proyecto.

## Ciclo de vida en las bases de datos

El ciclo de vida de las bases de datos en la ingeniería del software se compone de varias fases interrelacionadas:

- **Planificación**: Define qué se realizará, los recursos disponibles, las técnicas de recopilación de datos y el enfoque para diseño e implementación.

- **Definición**: Establece los límites de la aplicación, sistemas relacionados y usuarios involucrados.


- **Recogida y Análisis de Requisitos**: Recolecta y analiza los requisitos a través de entrevistas y análisis de documentos.

- **Diseño de la Base de Datos**: Divide en tres fases:

    - **Diseño Conceptual**: Crea un esquema independiente del sistema que representa la estructura de la base de datos.

    - **Diseño Lógico**: Refina el esquema conceptual (obteniendo un esquema lógico), eliminando elementos que no se pueden utilizar en el modelo de datos elegido (por ejemplo, relacional, orientado a objetos).

    - **Diseño Físico**: Traduce el esquema lógico en una esquema físico, considerando las formas de almacenamiento y métodos de acceso adecuados para el sistema de información y el SGBD elegido.

- **Diseño de la Aplicación**: Diseña programas de aplicación que interactúan con la base de datos y asegura la inclusión de requisitos de usuario.

- **Prototipado**: Crea un modelo de trabajo para que los usuarios evalúen el sistema y detecten problemas.

- **Implementación**: Crea el esquema de la base de datos, el sistema de almacenamiento y los procesos para trabajar con datos.

- **Conversión y Carga de Datos**: Reemplaza datos antiguos por nuevos en diferentes estructuras durante las actualizaciones.

- **Prueba y Mantenimiento**: Valida requisitos, realiza pruebas con datos reales y garantiza la calidad y confiabilidad del software.