Clasificación de datos en modelos de inteligencia artificial

Breve descripción:

Este componente tiene como objetivo que los aprendices desarrollen habilidades técnicas para gestionar y transformar datos en conocimiento aplicable, preparándolos para un entorno de inteligencia artificial que se encuentra cada vez más presente en diversas áreas y entornos digitales.

Julio de 2025

Tabla de contenido

Introducción

La sociedad actual vive una era caracterizada por una fuerza invisible, pero omnipresente: los datos. Desde el momento en que elementos como despertadores inteligentes sugieren la hora óptima para levantarse, basándose en el tráfico y el calendario, hasta las recomendaciones de series que esperan en los hogares al final del día, los datos constituyen el motor que impulsa la maquinaria de la vida moderna. Se han convertido en el activo más valioso del siglo XXI, el nuevo petróleo, un recurso bruto que, una vez refinado, posee el poder de transformar industrias, redefinir la experiencia humana y acelerar la innovación a un ritmo sin precedentes.

No obstante, referirse a los "datos" como una entidad monolítica constituye una simplificación excesiva y peligrosa. El ámbito de los datos es extenso, diverso y complejo. No todos los datos son iguales en su origen; su naturaleza, estructura y valor intrínseco varían considerablemente. Un registro de ventas en una hoja de cálculo de Excel, una fotografía subida a una red social, un correo electrónico, una transacción financiera y los datos de un sensor del Internet de las Cosas (IoT), representan aspectos completamente distintos de este universo digital.

Comprender esta diversidad constituye el primer paso para su gestión efectiva. La verdadera complejidad, así como el auténtico desafío, reside en la manera en que se captura, almacena, procesa, modela y estructura los sistemas que otorgan significado a este flujo constante de información. La capacidad de una organización para innovar, ya no depende exclusivamente del ingenio de sus ingenieros o de la visión de sus líderes, sino de la solidez y flexibilidad de su arquitectura de datos. Una estrategia de datos bien fundamentada es el andamiaje sobre el cual se construyen la inteligencia artificial (IA),

el aprendizaje automático, el análisis predictivo y las experiencias de usuario personalizadas que caracterizan a las empresas líderes en la actualidad.

La gestión de datos en modelos de inteligencia artificial constituye un elemento fundamental para el éxito de cualquier proyecto de IA. Esta gestión implica la administración estratégica y sistemática de los activos de datos de una organización, utilizando tecnología de IA para mejorar la calidad de los datos, los análisis y la toma de decisiones.

Es por esto que la gestión de datos en modelos con este tipo de tecnología, abarca el ciclo de vida completo de los datos, que incluye la recolección, limpieza, almacenamiento, gobernanza y preparación, con el objetivo de entrenar, validar y desplegar modelos de IA de manera eficaz. La aplicación de técnicas de inteligencia artificial y aprendizaje automático para optimizar estos procesos se denomina gestión de datos de inteligencia artificial.

Esta gestión permite la automatización de tareas como la identificación de anomalías, la integración de fuentes heterogéneas y la mejora de la calidad de los datos, este enfoque ha revolucionado la manera en que empresas y organizaciones procesan y utilizan la información para el entrenamiento de modelos IA permitiendo así tomar decisiones más informadas en el ámbito empresarial.

Este componente es el primer paso para adentrarse en la cultura de innovación. No es un manual técnico para programadores, sino una guía estratégica para innovadores, tecnólogos y curiosos que deseen comprender los cimientos sobre los que se edifica el futuro digital. Se explorará el ADN de la información, comenzando por sus componentes más básicos: los tipos de datos. Después se abordarán las diferencias

fundamentales entre los datos estructurados, semiestructurados y no estructurados, y por qué esta distinción es crucial para cualquier estrategia tecnológica.

A continuación, se explorará el ámbito de las bases de datos, los repositorios donde estos activos adquieren relevancia. Se presentarán las definiciones básicas para clasificar sus diversas formas, desde las tradicionales bases de datos relacionales que han sido el pilar de la informática empresarial durante décadas, hasta la proliferación de modelos NoSQL diseñados para satisfacer las demandas de escala, velocidad y diversidad de la web moderna.

Partiendo de lo anterior, se invita a que acceda al siguiente video, el cual relaciona la temática a tratar durante este componente formativo:

Video 1. Clasificación de datos en modelos de inteligencia artificial

Enlace de reproducción del video

Video 1. Síntesis del video: Clasificación de datos en modelos de inteligencia artificial
Dicen que los datos son el nuevo petróleo. Pero el petróleo en bruto no sirve de nada. El verdadero poder está en saber refinarlo. Por tal razón, en el presente componente formativo se aborda la importancia de la gestión de los datos en los modelos de inteligencia artificial, como herramienta clave para los modelos de datos de aprendizaje automático, dominando los fundamentos de los diferentes tipos de datos, la clasificación y el almacenamiento de los datos en bases de datos, hasta las arquitecturas que escalan y nunca fallan.

Luego, se explica el modelado de datos mediante la definición de estructuras de datos y sus relaciones, creando el puente entre los requisitos de negocios y la implementación técnica.

Finalmente, se presentan los diferentes tipos y técnicas de normalización de los datos en bases de datos, su funcionamiento y la forma en que las arquitecturas que soportan estas bases de datos trabajan.

Este componente permite a los aprendices desarrollar autonomía y pensamiento crítico, logrando fortalecer su capacidad para innovar en el entorno digital.

1. Tipos de datos

En su nivel más básico, un dato se define como una representación simbólica (numérica, alfabética, algorítmica, etc.) de un atributo o variable de una entidad. Constituye un hecho bruto, una unidad discreta de información objetiva sobre un evento. Aislado, un dato puede carecer de significado. Por ejemplo, el número "21" es simplemente un número. Sin embargo, al situarlo en un contexto como la edad de un cliente, la cantidad de productos en un carrito de compras o la temperatura en grados Celsius, adquiere significado y se transforma en información. La gestión eficaz de los datos comienza con el reconocimiento de sus diversas formas y estructuras.

Por otra parte, la inteligencia artificial (IA) y el aprendizaje automático han transformado significativamente la interacción con la tecnología y los datos. No obstante, el rendimiento y la eficacia de cualquier modelo de IA dependen de manera crítica de la calidad, estructura y gestión de los datos que lo sustentan. Un modelo sofisticado con datos mal gestionados generará resultados deficientes, un principio conocido como garbage in, garbage out o GIGO por sus siglas en inglés.

El principio GIGO es un concepto esencial en la gestión de datos que indica una conexión directa entre la calidad de los datos introducidos y la calidad de los resultados obtenidos. En términos sencillos, esto significa que, si se ingresan datos de baja calidad o incorrectos en un sistema, inevitablemente se obtendrán resultados igualmente deficientes o erróneos. Surge en los inicios de la computación, cuando los programadores advirtieron que los computadores, por muy avanzadas que fueran, solo podían procesar la información que recibían. Si dicha información era incorrecta,

incompleta o inconsistente, los resultados reflejarían esas deficiencias, independientemente de la calidad del procesamiento o de los algoritmos empleados.

Las consecuencias de no tener en cuenta este principio pueden ser serias y costosas. En el mundo empresarial, tomar decisiones basadas en datos incorrectos puede llevar a estrategias equivocadas, pérdidas económicas y oportunidades extraviadas. Este principio es un argumento claro que evidencia la importancia de tener buenos datos que serán utilizados en el entrenamiento de un modelo de IA.

1.1 Estructurados

Los datos estructurados son la forma más ordenada de información. Se distinguen por seguir un esquema fijo y preestablecido, donde cada componente ocupa una posición específica y tiene un tipo de dato claramente definido. Esta disposición en forma de tabla, facilita en gran medida su procesamiento, búsqueda y análisis. De forma breve, se podría afirmar que son aquellos que se adhieren a un modelo de datos predefinido y altamente organizado. Los datos estructurados son, sin duda, el tipo de datos más desarrollado y tradicionalmente más sencillo de gestionar y analizar por las máquinas. La característica principal de los datos estructurados es que su formato, tipo y longitud están predefinidos en una estructura rígida, conocida como "esquema". Un ejemplo de esto es una hoja de cálculo: cada columna está diseñada para contener un tipo específico de información (nombre, fecha, cantidad, etc.) y cada fila representa un registro único que se ajusta a dicha estructura.

Dentro de las características principales de este tipo de dato se tiene (Silberschatz, et al., 2011):

- **Esquema predefinido**

  La estructura (campos, tipos de datos) se define antes de que se introduzcan los datos. Esto se conoce como schema-on-write.

- **Formato tabular**

  Generalmente se organizan en tablas con filas y columnas.

- **Tipos de datos definidos**

  Cada columna está restringida a tipos de datos específicos, como INTEGER, VARCHAR, DATE, BOOLEAN, etc.

- **Facilidad de consulta**

  Su naturaleza organizada los hace ideales para ser consultados, procesados y analizados, mediante lenguajes de consulta estructurados, siendo SQL (Structured Query Language) el estándar de facto.

Utilizar datos estructurados tiene ventajas como:

- **Eficiencia en consultas**

  Facilitan búsquedas rápidas y precisas mediante el uso de lenguajes como SQL.

- **Integridad de datos**

  Los esquemas rígidos aseguran la consistencia y minimizan los errores.

- **Análisis estadístico**

  Son ideales para realizar cálculos matemáticos y análisis cuantitativos.

- **Compatibilidad**

  Ofrecen amplia compatibilidad con herramientas de análisis tradicionales.

Algunos ejemplos de este tipo de datos son las bases de datos relacionales, las hojas de cálculo, archivos CSV, sistemas ERP, datos de transacciones financieras. Sin embargo, estos datos presentan algunas limitaciones como:

- **Rigidez**

  La modificación de la estructura requiere cambios significativos en el sistema.

- **Escalabilidad limitada**

  Pueden volverse ineficientes con grandes volúmenes de datos.

- **Falta de flexibilidad**

  Dificultan el almacenamiento de información que no se ajusta al esquema predefinido.

Ejemplo: Tabla de Empleados ID | Nombre | Edad | Departamento | Salario 1 | Ana López | 28 | Marketing | 45000 2 | Juan Pérez| 35 | IT | 55000 3 | María García| 42 | Ventas | 48000

1.2 Semiestructurados

Los datos semiestructurados representan un punto medio entre la rigidez de los datos estructurados y la flexibilidad de los no estructurados. Estos datos incorporan elementos organizativos como etiquetas, metadatos o jerarquías, pero no requieren un esquema fijo para todos los elementos. Por consiguiente, estos datos no se ajustan a un

esquema formal y rígido como el de una base de datos relacional, pero incorporan etiquetas, metadatos o marcadores que separan los elementos semánticos y permiten una organización jerárquica. Esto les otorga un grado de auto-descripción y flexibilidad que los datos estructurados no poseen. La estructura no está definida en un esquema externo, sino que se encuentra contenida dentro de los propios datos. Considerando un documento que contiene información sobre una persona, este, en lugar de tener columnas fijas, podría disponer de etiquetas que describen cada pieza de información.

Al igual que los datos estructurados, los datos semiestructurados presentan unas características que ayudan a diferenciarlos de los otros tipos de datos, entre las cuales se encuentran:

- **Auto-descriptivos**

  Incorporan información sobre su propia estructura mediante metadatos.

- **Flexibilidad controlada**

  Permiten variaciones en la estructura sin comprometer la organización.

- **Intercambiabilidad**

  Facilitan el intercambio de datos entre diversos sistemas y plataformas.

- **Escalabilidad**

  Se adaptan de manera más eficiente a los cambios en los requisitos de datos.

Los datos semiestructurados suelen ser aplicados en tareas y actividades como:

- **Desarrollo web**

  En el ámbito del desarrollo web, JSON es esencial para las APIs REST y la comunicación entre servicios.

- **Configuración de sistemas**

  Configuración de sistemas

  En cuanto a la configuración de sistemas, XML se utiliza para gestionar archivos de configuración complejos. Para el intercambio de datos, se emplean formatos estándar que facilitan la comunicación entre aplicaciones.

- **Almacenamiento de logs**

  En el almacenamiento de logs, se requiere una estructura que permita un análisis adecuado, pero que también ofrezca flexibilidad para diferentes tipos de eventos.

Algunos ejemplos de este tipo de datos son:

- Archivos XML.
- JSON.
- HTML.
- Correos electrónicos.
- Logs de servidores.
- Documentos con metadatos.

A continuación, se presentan ejemplos típicos de dos de estos tipos de datos (Abiteboul, et al., 2000):

- **JSON (JavaScript Object Notation)**

  El formato predominante para la comunicación entre servicios web (APIs) y para bases de datos de documentos es el JSON. Un objeto JSON que describe a un usuario podría incluir campos básicos como "nombre" y "email". Sin embargo, otro usuario dentro de la misma colección podría contener campos adicionales, como "perfiles_sociales" (que es un objeto anidado), sin necesidad de modificar ninguna estructura predefinida.

  Ejemplo:

  {

  "empleado": { "id": 1,

  "nombre": "Ana López",

  "edad": 28,

  "departamento": "Marketing",

  "habilidades": ["SEO", "Redes Sociales", "Analytics"] }

  }

- **XML (eXtensible Markup Language)**

  Durante un período prolongado, fue considerado el estándar para el intercambio de datos. Emplea etiquetas personalizadas para definir tanto la estructura como los elementos de los datos. Su uso es ampliamente reconocido en archivos de configuración, documentos de ofimática (como .docx) y en la industria editorial.

  Ejemplo:

  <usuario id="usr123">

  <nombre>Ana Pérez</nombre>

  <email>ana.perez@example.com</email>

<activa>true</activa>

<pedidos>

<pedido id="ped001">

<total>49.99</total>

</pedido>

<pedido id="ped002">

<total>120.50</total>

</pedido>

</pedidos>

</usuario>

Los datos semiestructurados son el lenguaje de la web moderna y de las aplicaciones que necesitan interactuar entre sí de forma flexible y escalable.

1.3 No estructurados

Los datos no estructurados constituyen la mayor parte de la información generada digitalmente. Estos datos carecen de un formato predefinido y requieren técnicas avanzadas de procesamiento para extraer información valiosa.

Debido a su naturaleza este tipo de datos presenta complejidad al momento de procesarlos. Dentro de las complejidades que se generan se pueden encontrar:

- **Análisis de texto**

  Requiere el uso de procesamiento de lenguaje natural (NLP) para la extracción de significado.

- **Reconocimiento de patrones**

  Las imágenes y videos requieren la aplicación de algoritmos de visión computacional.

- **Minería de datos**

  Se emplean técnicas de aprendizaje automático para descubrir patrones ocultos.

- **Contextualización**

  La interpretación está fuertemente influenciada por el contexto y el dominio.

El manejo de los datos no estructurados, aunque implica las complejidades señaladas previamente, tiene un valor empresarial:

- **Perspectivas profundas**

  Ofrecen información detallada sobre comportamientos, opiniones y tendencias.

- **Ventaja competitiva**

  Las empresas que procesan estos datos de manera efectiva obtienen ventajas significativas.

- **Innovación**

  Facilitan nuevos tipos de análisis y aplicaciones previamente imposibles.

Casos comunes de este tipo de datos se pueden encontrar en: documentos de texto, imágenes, videos, audio, redes sociales, presentaciones, PDF.

Ejemplo aplicado:

Reseña de producto "Compré este producto la semana pasada y estoy muy satisfecho. La calidad es excelente y el envío fue rápido. Lo recomiendo totalmente. El precio es un poco alto, pero vale la pena por la durabilidad."

- **Archivos de texto**

  El cuerpo de correos electrónicos, documentos de Word, archivos PDF, publicaciones en redes sociales, comentarios en blogs, respuestas de encuestas abiertas.

- **Contenido multimedia**

  Imágenes (JPEG, PNG), archivos de audio (MP3, WAV) y archivos de video (MP4, AVI). Cada uno de estos archivos es un blob binario de datos desde la perspectiva de una base de datos tradicional.

- **Datos de sensores y logs**

  Los registros (logs) generados por servidores, aplicaciones y redes, así como los datos crudos provenientes de sensores del Internet de las Cosas (IoT), a menudo se presentan como un flujo continuo de texto sin formato.

- **Imágenes médicas**

  Radiografías, resonancias magnéticas y tomografías, son datos no estructurados de vital importancia en el sector de la salud.

En la siguiente tabla se presenta un comparativo de los datos tratado en esta sección:

Tabla 1. Comparativo tipos de datos

Aspecto	Estructurados	Semiestructurados	No estructurados
Organización.	Altamente organizado, formato rígido.	Parcialmente organizado, flexible.	Sin organización formal.
Almacenamiento.	Bases de datos relacionales.	NoSQL, almacenes de documentos.	Data lakes, sistemas de archivos.
Procesamiento.	SQL, herramientas tradicionales.	API, parsers especializados.	IA, ML, procesamiento de lenguaje natural.
Velocidad de análisis.	Rápido y eficiente.	Moderado.	Lento, requiere preprocesamiento.
Escalabilidad.	Limitada por esquema fijo.	Alta flexibilidad.	Muy alta, pero compleja.
Casos de uso.	Transacciones, reportes financieros.	APIs web, configuraciones.	Análisis de sentimientos, reconocimiento.

Se puede concluir que, la gestión moderna de datos requiere un enfoque híbrido que combine los tres tipos de datos. Las arquitecturas de datos actuales, como los data lakes, permiten almacenar y procesar todos estos tipos de datos de manera integrada.

Las tecnologías de inteligencia artificial y el aprendizaje automático están revolucionando la capacidad de obtener valor de datos no estructurados, mientras que los sistemas NoSQL han mejorado considerablemente el manejo de datos semiestructurados.

La elección del tipo de dato y las tecnologías relacionadas, debe estar basada en los objetivos específicos del negocio, los recursos disponibles, así como en los requisitos de rendimiento y escalabilidad de cada organización.

2. Bases de datos

Si los datos constituyen el recurso bruto, las bases de datos actúan como las refinerías y los almacenes. Estos son sistemas organizados diseñados para contener, gestionar y proporcionar acceso eficiente a la información. La selección de la base de datos adecuada representa una de las decisiones arquitectónicas más críticas en el diseño de cualquier sistema de software, con implicaciones directas en el rendimiento, la escalabilidad, la fiabilidad y el costo. Es por eso que el manejo de bases de datos siempre ha sido un tema crucial en el ámbito de los sistemas de información. No obstante, en años recientes, la expansión de internet y el rápido avance de la tecnología relacionada con Internet han convertido el conocimiento sobre tecnología de bases de datos en una de las carreras más emocionantes. (Kroenke, 2003)

A medida que las empresas comenzaron a reconocer el valor de la información y el inmenso potencial que los sistemas computacionales ofrecían para organizar y gestionar estos recursos, surgió una fuerte demanda de sistemas de información y un creciente reconocimiento de que la información, como recurso valioso, necesita ser organizada y administrada. (Gómez, et al., 2017)

2.1 Concepto

Los sistemas de bases de datos están diseñados para manejar grandes volúmenes de información. La gestión de datos, abarca tanto la creación de estructuras para el almacenamiento de información, como la implementación de mecanismos para su manipulación. Estos sistemas deben garantizar la integridad de la información almacenada, frente a fallos del sistema o intentos de acceso no autorizado. (Gómez, et al., 2017)

Un Sistema de Gestión de Bases de Datos (SGBD), se compone de un conjunto de datos interconectados y una serie de programas que permiten el acceso a esos datos. Esta colección de datos, comúnmente llamada base de datos, contiene información importante para una empresa. (Silberschatz, et al., 2020)

Para que el sistema sea efectivo, es esencial que recupere los datos de manera eficiente. Esta necesidad ha llevado al desarrollo de estructuras de datos complejas para representar la información en la base de datos. Normalmente, la arquitectura de un SGBD se divide en tres niveles generales (Gómez, et al., 2017):

- **Nivel físico (interno)**

  Este es el nivel más básico de abstracción, que detalla cómo se almacenan los datos en realidad. En el nivel físico, se describen minuciosamente las estructuras de datos complejas de bajo nivel.

- **Nivel lógico (conceptual)**

  El siguiente nivel de abstracción más elevado detalla qué datos se guardan en la base de datos y las relaciones que existen entre ellos. De esta manera, la base de datos completa se describe en términos de unas pocas estructuras relativamente simples.

- **Nivel de vistas (externo)**

  En el nivel más elevado de abstracción, se representa solo una parte de la base de datos completa. Esto es lo que el usuario final puede ver del sistema terminado, mostrando únicamente una sección de la base de datos al usuario autorizado para acceder a ella. El sistema tiene la capacidad de ofrecer múltiples vistas para la misma base de datos.

Como se había descrito previamente, un SGBD es un programa que se comunica con los usuarios, las aplicaciones y la propia base de datos para capturar, analizar y administrar los datos. Actúa como una capa de abstracción entre los datos físicos; es decir, la forma en que se almacenan en el disco, y la vista lógica que perciben los usuarios y las aplicaciones. El SGBD se encarga de llevar a cabo tareas esenciales como (Date, 2006):

- **Definición de datos**

  Proporcionar un lenguaje (como el DDL - Data Definition Language de SQL) para definir el esquema de la base de datos.
- **Manipulación de datos**

  Permitir a los usuarios insertar, actualizar, eliminar y consultar datos (usando un lenguaje como el DML - Data Manipulation Language de SQL).
- **Control de concurrencia**

  Gestionar el acceso simultáneo a los datos por parte de múltiples usuarios o procesos para evitar conflictos.
- **Seguridad y autorización**

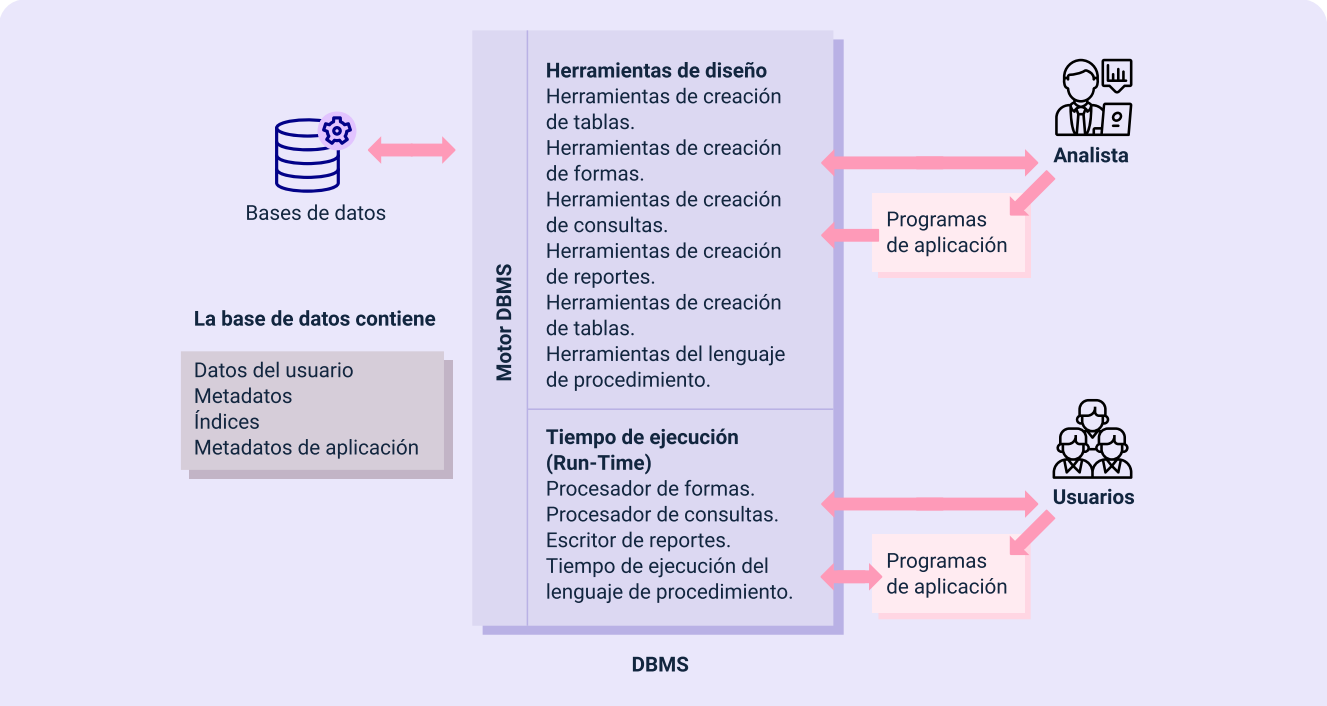
  Controlar quién puede ver y modificar los datos.
- **Integridad de datos**

  Asegurar que los datos cumplan con ciertas reglas y restricciones.
- **Respaldo y recuperación**

Proporcionar mecanismos para proteger los datos contra fallos del sistema.

En la siguiente figura, se presenta cómo se relacionan los componentes de un sistema de base de datos:

Figura 1. Componentes de los sistemas de bases de datos



Fuente: Kroenke (2003, p26)

Componentes de los sistemas de bases de datos

Bases de datos

La base de datos contiene

Datos del usuario.

Metadatos.

Índices.

Metadatos de aplicación.

Motor DBMS

Herramientas de diseño

Tiempo de ejecución (Run-Time)

Herramientas de creación de tablas.	Procesador de formas.
Herramientas de creación de formas.	Procesador de consultas.
Herramientas de creación de consultas.	Escritor de reportes.
Herramientas de creación de reportes.	Tiempo de ejecución del lenguaje de
Herramientas de creación de tablas.	Procedimiento.
Herramientas del lenguaje de procedimiento.	
Programas de aplicación.	**Programas de aplicación.**
Analista.	**Usuario.**

En síntesis, una base de datos se refiere a la información almacenada, mientras que el Sistema de Gestión de Bases de Datos (SGBD), es el sistema encargado de su administración. En el uso cotidiano, es común emplear el término "base de datos" para referirse tanto al SGBD como a los datos que este contiene.

2.2 Clasificación y características

Las bases de datos se pueden categorizar de diversas maneras, por su modelo de datos, su propósito específico o su arquitectura de distribución. Algunos de estas categorías son (Kleppmann, 2017):

- Por modelo de datos
- Por uso previsto

A continuación, se presenta una descripción más completa basada en esta categorización:

Clasificación por modelado de dato

Dentro esta categoría existe dos tipos:

- **Relacionadas (SQL)**

  Basadas en el modelo relacional, organizan los datos en tablas. Son la opción dominante para datos estructurados. Ejemplos: PostgreSQL, MySQL, Oracle.

- **No Relacionadas (NoSQL) (SQL)**

  Una categoría amplia que abarca todos los modelos que no son primariamente relacionales. Están diseñadas para la flexibilidad y la escala.

Las categorías no relacionales, incluyen:

- **Documentales**

  Almacenan datos en documentos (JSON, BSON). Ejemplos: MongoDB, Couchbase.

- **Clave-valor**

  El modelo más simple, almacena datos como un diccionario. Ejemplos: Redis, Amazon DynamoDB.

- **Columnares (Wide-Column)**

  Puerta trasera (backdoor): permite el acceso remoto a un sistema sin autorización, evitando los mecanismos de seguridad. Actúa de forma silenciosa, facilitando el control total por parte del atacante.

- **Grafos**

  Optimizadas para almacenar y navegar relaciones entre entidades. Ejemplos: Neo4j, Amazon Neptune.

- **Series temporales**

  Especializadas en datos indexados por tiempo. Ejemplos: InfluxDB, TimescaleDB.

Clasificación por uso previsto

Esta clasificación tiene también dos tipos:

- **OLTP (Online Transaction Processing)**

  Se refiere a sistemas optimizados para manejar un gran volumen de transacciones breves y rápidas, tales como lecturas, inserciones y actualizaciones. Constituyen la columna vertebral de aplicaciones orientadas al usuario, como el comercio electrónico y la banca. Su diseño prioriza la velocidad de escritura y la integridad de los datos. Las bases de datos relacionales son típicamente ejemplos de sistemas OLTP.

- **OLAP (Online Analytical Processing)**

  Se caracteriza por su optimización para consultas complejas que examinan grandes volúmenes de datos históricos. Constituye la base de los sistemas de inteligencia empresarial y almacenes de datos. Este enfoque prioriza la velocidad de lectura y la capacidad de agregación. Las bases de datos columnares son particularmente adecuadas para OLAP.

Independientemente de su tipo (modelado de datos - uso previsto), existen diversas características que determinan la calidad y capacidad de una base de datos, encontrando:

Propiedades ACID: un acrónimo que asegura la fiabilidad de las transacciones. Constituye el estándar de excelencia para las bases de datos OLTP y se explica a continuación:

- **Atomicidad (Atomicity)**

  Una transacción se ejecuta en su totalidad o no se ejecuta en absoluto, sin estados intermedios.
- **Consistencia (Consistency)**

  Una transacción únicamente puede transformar la base de datos de un estado válido a otro.
- **Aislamiento (Isolation)**

  Las transacciones concurrentes no interfieren entre sí, produciendo un resultado equivalente al de una ejecución en serie.
- **Durabilidad (Durability)**

  Una vez que una transacción ha sido confirmada, su estado persiste, incluso en caso de un fallo del sistema.

Otras características en común que tienen las bases de datos son:

A. Escalabilidad

Es la capacidad del sistema para manejar una carga de trabajo creciente. Dentro de los tipos de escalado que se pueden encontrar están:

Escalado Vertical (Scale-up)

Aumenta los recursos de un único servidor (más CPU, RAM, disco). Tiene un límite físico y es costoso.

Escalado Horizontal (Scale-out)

Distribuye la carga entre múltiples servidores. Es la estrategia de las arquitecturas web modernas y es fundamental en los sistemas NoSQL.

B. Disponibilidad

Hace referencia al porcentaje de tiempo que la base de datos está operativa y disponible para su uso. Los sistemas de alta disponibilidad utilizan técnicas como la replicación para asegurar que no haya un único punto de fallo.

C. Rendimiento

Medido por la latencia (tiempo de respuesta a una consulta) y el throughput (número de operaciones por segundo).

2.3 Modelos de bases de datos

Debajo de la estructura de la base de datos se halla el modelo de datos, que es un conjunto de herramientas conceptuales utilizadas para describir los datos, sus relaciones, la semántica y las restricciones de consistencia (Silberschatz, et al., 2020). Dicha de forma breve, son las estructuras que facilitan la organización, almacenamiento y gestión de la información. A continuación, se describen los principales modelos utilizados en las bases de datos de acuerdo con Robinson, et al. (2015):

- **Modelo relacional**

  Propuesto por Edgar F. Codd en 1970, este modelo transformó la industria. Los datos se organizan en tablas (o "relaciones"), donde cada tabla consiste en una serie de filas (o "tuplas") y columnas (o "atributos"). Las conexiones entre las tablas se logran mediante claves externas (`foreign keys`), que son campos que señalan el ID único (clave primaria o `primary key`) de una fila en otra tabla. SQL es el lenguaje estándar para interactuar con este modelo. Su principal ventaja es la integridad y consistencia de los datos, garantizadas por las propiedades ACID.

- **Modelo de documentos**

  En lugar de utilizar filas y columnas, se guarda la información en documentos. Un documento es una estructura de datos independiente, generalmente en formato JSON o BSON (JSON binario), que puede ser tan complejo y anidado como se requiera. Toda la información de una entidad (como un usuario y todos sus pedidos) puede estar contenida en un solo documento. Esto elimina la necesidad de realizar JOINs complejos y costosos, lo que permite lecturas extremadamente rápidas. Es altamente flexible, no es necesario que todos los documentos de una colección tengan la misma estructura. Es ideal para catálogos de productos, perfiles de usuario y sistemas de gestión de contenido.

- **Modelo clave-valor**

  Este es el modelo NoSQL más básico. Se compone de una extensa tabla hash en la que cada elemento está asociado a una clave única y un valor. Para la base de datos, el valor es un "`blob`" opaco; puede ser cualquier cosa, desde una cadena de texto hasta un objeto JSON o una imagen. Son

extremadamente rápidos y permiten una escalabilidad horizontal. Se emplean para cachés de alta velocidad (como Redis), almacenamiento de sesiones de usuario y aplicaciones en tiempo real.

- **Modelo columnar (Wide-Column)**

  A diferencia de las bases de datos relacionales que almacenan datos por filas, las bases de datos de modelo columnar los almacenan por columnas. Cada fila puede contener un conjunto distinto de columnas. Están optimizadas para agregar valores de una columna específica, a través de millones de filas, lo que las hace ideales para cargas de trabajo de análisis (OLAP). Por ejemplo, para calcular el promedio de ventas de un producto, es necesario leer únicamente la columna "ventas", ignorando todas las demás, lo que resulta en una mayor eficiencia.

- **Modelo de grafo**

  Específicamente diseñado para datos en los que las relaciones constituyen el aspecto más crucial de la información. Los datos se representan como nodos (o vértices) y las relaciones entre ellos como aristas (o ejes). Tanto los nodos como las aristas pueden poseer propiedades. Son ideales para aplicaciones como redes sociales, sistemas de recomendación, detección de fraudes y logística.

2.4. Técnicas de almacenamiento de datos

Los datos se almacenan físicamente en el disco y en la forma en que se hagan tiene un impacto masivo en el rendimiento. De acuerdo a Elmasri & Navathe (2016), suelen clasificarse en:

- **Almacenamiento orientado a filas**

  Representa el enfoque tradicional de las bases de datos relacionales (OLTP). En este método, todos los valores de una fila se almacenan de manera contigua en el disco. Es altamente eficiente cuando se requiere recuperar una fila completa, como en el caso de la consulta `SELECT * FROM usuarios WHERE id = 123`, ya que todos los datos necesarios se leen en una única operación de entrada/salida de disco. No obstante, si solo se necesitan unas pocas columnas de una tabla con un gran número de columnas, este método resulta ineficiente, dado que se deben leer todos los datos de la fila, aunque la mayoría de ellos no sean necesarios.

- **Almacenamiento orientado a columnas**

  Técnica empleada por las bases de datos analíticas (OLAP). Todos los valores de una misma columna se almacenan de manera contigua en el disco. Presenta dos ventajas significativas para el análisis de datos. En primer lugar, la eficiencia de lectura se ve mejorada. En consultas que solo requieren un subconjunto de columnas, únicamente se accede a los datos de dichas columnas, lo que reduce de manera considerable la entrada/salida de disco. En segundo lugar, se consigue una compresión más eficiente. Debido a que los datos dentro de una misma columna tienden a ser uniformes en cuanto a tipo y rango de valores, pueden comprimirse de manera más eficaz que los datos variados de una fila. Esto no solo reduce el espacio de almacenamiento necesario, sino que también agiliza las consultas, ya que hay menos datos que leer del disco.

Indexación

Un índice es una estructura de datos auxiliar que mejora la velocidad de las operaciones de recuperación de datos en una tabla. Funciona de manera similar al índice de un libro: en lugar de leer todo el libro para encontrar un tema, se va al índice, se encuentra el tema y se dirige directamente a la página correcta. Entre los más usados se encuentran:

- **Árboles B (B-Trees)**

  Son la estructura de índice más prevalente en las bases de datos relacionales. Esta estructura de árbol se auto-balancea, manteniendo los datos en orden y permitiendo realizar búsquedas, inserciones y eliminaciones de manera eficiente en tiempo logarítmico. Es adecuada tanto para búsquedas directas (`WHERE id = 123`) como para búsquedas por rango (`WHERE edad > 30`).

- **Índices Hash**

  Crean una tabla `hash` donde la clave es el valor de la columna indexada y el valor es un puntero a la fila del disco. Son extremadamente rápidos para búsquedas de igualdad exactas, pero no son útiles para búsquedas por rango.

- **Índices invertidos**

  Son la base de los motores de búsqueda de texto completo. Crean un mapa desde el contenido (palabras) a su ubicación en los documentos.

3. Modelado de datos

El proceso de crear una representación visual o un esquema que define las estructuras de datos, sus interrelaciones, sus atributos y las normas que los rigen constituyen una de las fases más críticas en el diseño de un sistema de información. Un modelo de datos bien elaborado asegura la consistencia, la calidad y la usabilidad de la información a largo plazo. Este proceso actúa como un puente entre los requisitos del negocio y la implementación técnica en una base de datos.

3.1. Concepto

El modelado de datos constituye un proceso de abstracción que se inicia con la recopilación de los requisitos de información del mundo real, los cuales se traducen progresivamente en una estructura formal comprensible para una base de datos. Un modelo de datos bien diseñado debe garantizar la integridad de la información.

Asimismo, establece las normas y limitaciones para asegurar que solo se guarden datos correctos. Al minimizar la redundancia se evita la repetición innecesaria de datos, lo que ahorra espacio y previene discrepancias. También, optimiza el rendimiento si se cuenta con una estructura lógica bien diseñada para realizar consultas de manera más rápida y eficiente. Por último, facilita la comunicación ofreciendo un lenguaje común para que los analistas de negocio, desarrolladores y administradores de bases de datos puedan discutir y comprender los requisitos de información. (Simsion & Witt, 2004).

El proceso de modelado generalmente se divide en tres fases:

1. **Modelo conceptual**

   Es el nivel más alto de abstracción. Describe las entidades principales, las relaciones y los atributos de interés para el negocio, sin entrar en detalles

técnicos. Se centra en el "qué". Un diagrama Entidad-Relación (ERD) es el artefacto típico de esta fase.

2. **Modelo lógico**

   Traduce el modelo conceptual a una estructura más detallada, pero todavía independiente del SGBD específico que se utilizará. Define los tipos de datos, las claves primarias y externas, y las relaciones con mayor precisión. Aquí es donde se aplican las técnicas de normalización.

3. **Modelo físico**

   Es la implementación real del modelo lógico en un SGBD concreto. Describe cómo se construirán los datos en la base de datos, incluyendo la definición exacta de las tablas, los nombres de las columnas, los tipos de datos específicos del SGBD, los índices, las particiones y otras características de almacenamiento.

3.2 Principios

Un modelado de datos eficaz se basa en varios principios fundamentales:

- **Claridad y simplicidad**

  El modelo debe ser comprensible para todas las partes interesadas. Es esencial evitar nombres de tablas y columnas que resulten crípticos. La simplicidad no implica una falta de detalle, sino la eliminación de complejidades innecesarias.

- **Completitud**

  El modelo debe capturar todos los requisitos de datos pertinentes al problema que se está abordando, sin omitir entidades o atributos significativos.

- **No redundancia (normalización)**

  La información no debe almacenarse repetidamente sin necesidad. En ocasiones, se introduce deliberadamente redundancia controlada por razones de rendimiento (desnormalización), pero esta debe ser una decisión consciente y justificada.

- **Flexibilidad y extensibilidad**

  El modelo debe poseer la capacidad de evolucionar. Dado que los requisitos empresariales están sujetos a cambios, el modelo de datos debe ser capaz de adaptarse a nuevos tipos de información o a nuevas relaciones sin necesidad de una reestructuración masiva.

- **Integridad**

  El modelo debe establecer las reglas que aseguren la corrección y consistencia de los datos. Esto incluye la definición de claves primarias, claves externas, restricciones de unicidad y reglas de validación.

Los bloques de construcción básicos de un modelo de datos (especialmente en el modelado relacional) son:

Entidad

Se refiere a un objeto o concepto del mundo real del cual deseamos guardar información, como **Cliente, Producto o Pedido.** En el modelo físico, se transforma en una tabla.

Atributo

Es una propiedad o característica de una entidad, por ejemplo, en la entidad **Cliente**, los atributos pueden incluir Nombre, Email y FechaDeNacimiento. Estos se convierten en columnas dentro de una tabla.

Relación

Explica cómo se vinculan dos o más entidades entre sí, como cuando un **Cliente** realiza un **Pedido**. Las relaciones poseen cardinalidad, que puede ser uno a uno, uno a muchos, o muchos a muchos.

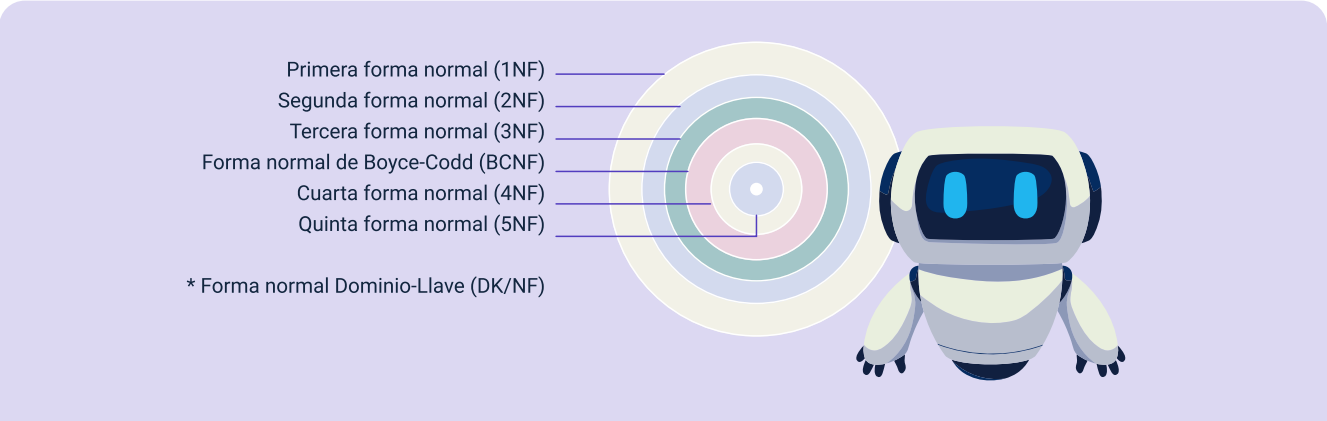
3.3. Tipos y técnicas de normalización

La normalización constituye un proceso sistemático destinado a organizar las columnas y tablas en una base de datos relacional, con el fin de minimizar la redundancia de datos. Su objetivo es descomponer tablas grandes y complejas en tablas más pequeñas y bien estructuradas, además de definir las relaciones entre ellas. Este proceso contribuye a eliminar las anomalías de datos que pueden surgir durante las operaciones de inserción, actualización o eliminación.

Hay diversas "formas normales" (FN o NF), cada una con un conjunto de reglas más riguroso que la anterior. En la práctica, la mayoría de las bases de datos OLTP bien estructuradas logran alcanzar la Tercera Forma Normal (3FN). (Codd, 1970).

En la siguiente figura se presenta la relación de crear formales:

Figura 2. Relación de formales



Fuente: Kroenke (2003, p.128).

Relación de formales

Primera forma normal (1NF).

Segunda forma normal (2NF).

Tercera forma normal (3NF).

Forma normal de Boyce-Codd (BCNF).

Cuarta forma normal (4NF).

Quinta forma normal (5NF).

Forma normal Dominio-Llave (DK/NF).

De acuerdo a lo anterior, a continuación, se detallan las formas más frecuentes:

- **Primera Forma Normal (1FN)**

  Una tabla está en 1FN si todos sus atributos contienen valores atómicos (indivisibles) y cada fila es única. Esto significa que no puede haber grupos de repetición o arreglos en una columna. Por ejemplo, una columna teléfonos que contenga "555-1234, 555-5678" viola la 1FN. Para solucionar esto se crea una tabla separada para la información repetida. En el ejemplo de los teléfonos, se crearía una tabla TelefonosCliente con las columnas IDCliente y Teléfono, donde cada número de teléfono ocupa una fila separada.

- **Segunda Forma Normal (2FN)**

  La tabla debe estar en 1FN. Todos los atributos que no forman parte de la clave primaria deben depender funcionalmente de la clave primaria completa. Esta regla solo es relevante para tablas con claves primarias compuestas (formadas por más de una columna). Si un atributo depende solo de una parte de la clave compuesta, debe moverse a una tabla separada. Considere una tabla DetallesPedido con la clave primaria (IDPedido, IDProducto). Si la tabla contiene la columna DescripcionProducto, este atributo viola la 2FN porque DescripcionProducto depende solo de IDProducto, no de la clave completa. Para solucionar esto se mueve DescripcionProducto a la tabla Productos, donde IDProducto es la clave primaria.

- **Tercera Forma Normal (3FN)**

  La tabla debe estar en 2FN. Ningún atributo no clave debe tener una dependencia transitiva de la clave primaria. Una dependencia transitiva ocurre cuando un atributo no clave depende de otro atributo no clave, que a su vez depende de la clave primaria. Por ejemplo, considere una tabla Empleados con las columnas IDEmpleado (clave), NombreEmpleado, IDDepartamento y NombreDepartamento. Aquí, NombreDepartamento depende de IDDepartamento, y IDDepartamento depende de IDEmpleado. Esto es una dependencia transitiva. Si el nombre de un departamento cambia, habría que actualizarlo en cada registro de empleado de ese departamento, lo que puede causar inconsistencias. Para solucionar esto se crea una tabla separada Departamentos con IDDepartamento y NombreDepartamento. La tabla Empleados solo contendrá la clave externa IDDepartamento.

Alcanzar la 3FN, suele ofrecer un buen equilibrio entre la disminución de la redundancia y la complejidad del modelo. Existen formas normales más avanzadas como (BCNF, 4NF, 5NF) para abordar anomalías más sutiles, pero son menos frecuentes en la práctica. Por otro lado, la desnormalización es el proceso opuesto, donde se introduce intencionalmente redundancia para mejorar el rendimiento de lectura al evitar JOINs costosos. Es una técnica común en en sistemas OLAP y almacenes de datos.

4. Arquitectura de bases de datos

La arquitectura de una base de datos se refiere al diseño integral del sistema que la sustenta. No se limita únicamente al Sistema de Gestión de Bases de Datos (SGBD), sino que abarca la integración del hardware, el software, las redes y las estrategias de distribución de datos para satisfacer los requisitos de rendimiento, disponibilidad, escalabilidad y seguridad de una aplicación.

4.1. Concepto

La arquitectura de una base de datos constituye el esquema que detalla la organización, interconexión y gestión de los diversos componentes de un sistema de datos. Una arquitectura bien diseñada busca equilibrar objetivos que frecuentemente están en conflicto:

- **Rendimiento frente a costo**

  Un sistema que ofrece un rendimiento extremadamente alto puede necesitar hardware de alto costo.

- **Consistencia frente a disponibilidad**

  En los sistemas distribuidos, asegurar una consistencia rigurosa en todos los nodos puede comprometer la disponibilidad si la red presenta fallos (Teorema CAP).

- **Seguridad frente a usabilidad**

  Controles de seguridad excesivamente estrictos pueden complicar el acceso legítimo a la información.

La elección de una arquitectura depende críticamente de la naturaleza de la aplicación. Un blog personal con pocos visitantes, tiene requisitos arquitectónicos radicalmente diferentes a los de un sistema de comercio electrónico global o una plataforma de redes sociales.

4.2. Tipos

Las arquitecturas de bases de datos han evolucionado desde sistemas monolíticos y centralizados hasta ecosistemas distribuidos y elásticos en la nube. De acuerdo con Özsu & Valduriez (2011), se clasifican en:

- **Arquitectura centralizada**

  Tanto los datos como el SGBD se encuentran en un solo servidor o clúster en una ubicación específica. Los usuarios y las aplicaciones acceden a este punto central. Es sencillo de administrar, ofrece alta seguridad (ya que solo hay un punto que proteger) y garantiza la consistencia de los datos. Sin embargo, presenta la desventaja de ser un único punto de fallo. La capacidad de escalar está limitada por el servidor (escalado vertical). La latencia puede ser elevada para usuarios que se encuentran a gran distancia geográfica. Se suele usar en pequeñas y medianas empresas, aplicaciones departamentales.

- **Arquitectura descentralizada**

  Se caracteriza por la presencia de múltiples bases de datos autónomas, cada una con su propio Sistema de Gestión de Bases de Datos (SGBD) y datos, que pueden estar ubicadas en diferentes localizaciones. No comparten un control centralizado y tienen la capacidad de operar de

manera independiente. Un ejemplo de este tipo de sistemas es aquellos basados en la tecnología blockchain. Como desventaja se tiene que la consistencia global de los datos es muy difícil de lograr.

- **Arquitectura distribuida**

  Se refiere a un sistema de base de datos dividido físicamente en múltiples servidores (nodos), ubicados en diferentes localizaciones. Los nodos se comunican mediante una red y colaboran para ofrecer una vista unificada de los datos. Este paradigma predomina en aplicaciones a gran escala. Sus ventajas incluyen alta escalabilidad (permite añadir nodos para gestionar mayor carga), alta disponibilidad (si un nodo falla, otros asumen su función) y baja latencia (los datos pueden ubicarse cerca de usuarios). Como desventaja, presenta mayor complejidad de gestión, y garantizar la consistencia entre nodos es un desafío técnico (consistencia eventual vs. fuerte). Se suele aplicar en aplicaciones web globales, redes sociales, IoT, Big Data.

- **Arquitectura en la Nube (DBaaS - Database as a Service)**

  Es una forma de arquitectura distribuida donde un proveedor de la nube (como Amazon Web Services, Google Cloud o Microsoft Azure) gestiona la infraestructura de la base de datos. El cliente simplemente consume la base de datos como un servicio a través de una API o un panel de control. Permite una gran elasticidad (se puede escalar hacia arriba o hacia abajo bajo demanda), pago por uso, gestión simplificada (el proveedor se encarga de los parches, respaldos, etc.), alta disponibilidad y durabilidad incorporadas, sin embargo, dentro de sus desventajas se tiene la

dependencia de un proveedor, preocupaciones sobre la seguridad y soberanía de los datos para ciertas industrias. Su aplicación se ve en startups hasta grandes empresas que buscan agilidad y reducir la carga operativa.

4.3. Métodos de funcionamiento

En las arquitecturas distribuidas, hay dos métodos esenciales para dividir y manejar los datos entre los nodos (Kleppmann, 2017):

A. Replicación

Implica el mantenimiento de copias idénticas de los datos en múltiples nodos, principalmente para asegurar una alta disponibilidad y mejorar el rendimiento de lectura. En caso de que el nodo principal (maestro) falle, una de las réplicas (esclavo) puede ser promovida para asumir su función, minimizando así el tiempo de inactividad. Además, las solicitudes de lectura pueden distribuirse entre las réplicas para evitar la sobrecarga del maestro. Dentro de los modelos en esta arquitectura se tiene a los modelos:

- **Maestro-Esclavo**

  Todas las operaciones de escritura se realizan en el nodo maestro, el cual posteriormente propaga los cambios a los nodos esclavos. Las operaciones de lectura pueden llevarse a cabo desde cualquier nodo. Este modelo es el más comúnmente utilizado.

- **Multi-Maestro**

  Varios nodos tienen la capacidad de aceptar escrituras. Este enfoque es beneficioso para aplicaciones distribuidas geográficamente, ya que reduce

la latencia de escritura; sin embargo, introduce una considerable complejidad en la resolución de conflictos de escritura, especialmente cuando dos usuarios modifican simultáneamente el mismo dato en diferentes nodos maestros.

Es importante tener presente que en este tipo de método de funcionamiento puede haber un pequeño retraso entre el momento en que se escribe en el maestro y el momento en que la escritura se refleja en los esclavos. Leer desde un esclavo podría devolver datos obsoletos, un concepto conocido como consistencia eventual.

B. Particionamiento o fragmentación

La base de datos se divide en particiones más pequeñas, llamadas shards, y cada shard es almacena en un nodo diferente. Cada nodo es responsable de un subconjunto de los datos. Cuando una base de datos se vuelve demasiado grande para un solo servidor o el volumen de escrituras es demasiado alto, el particionamiento permite distribuir la carga entre muchos servidores. Para crear las particiones se suelen utilizar estrategias como:

- **Basado en rango**

  Se asignan rangos de la clave de particionamiento a diferentes shards (p.ej., usuarios con ID de 1 a 1,000,000 en el shard 1; 1,000,001 a 2,000,000 en el shard 2).

- **Basado en hash**

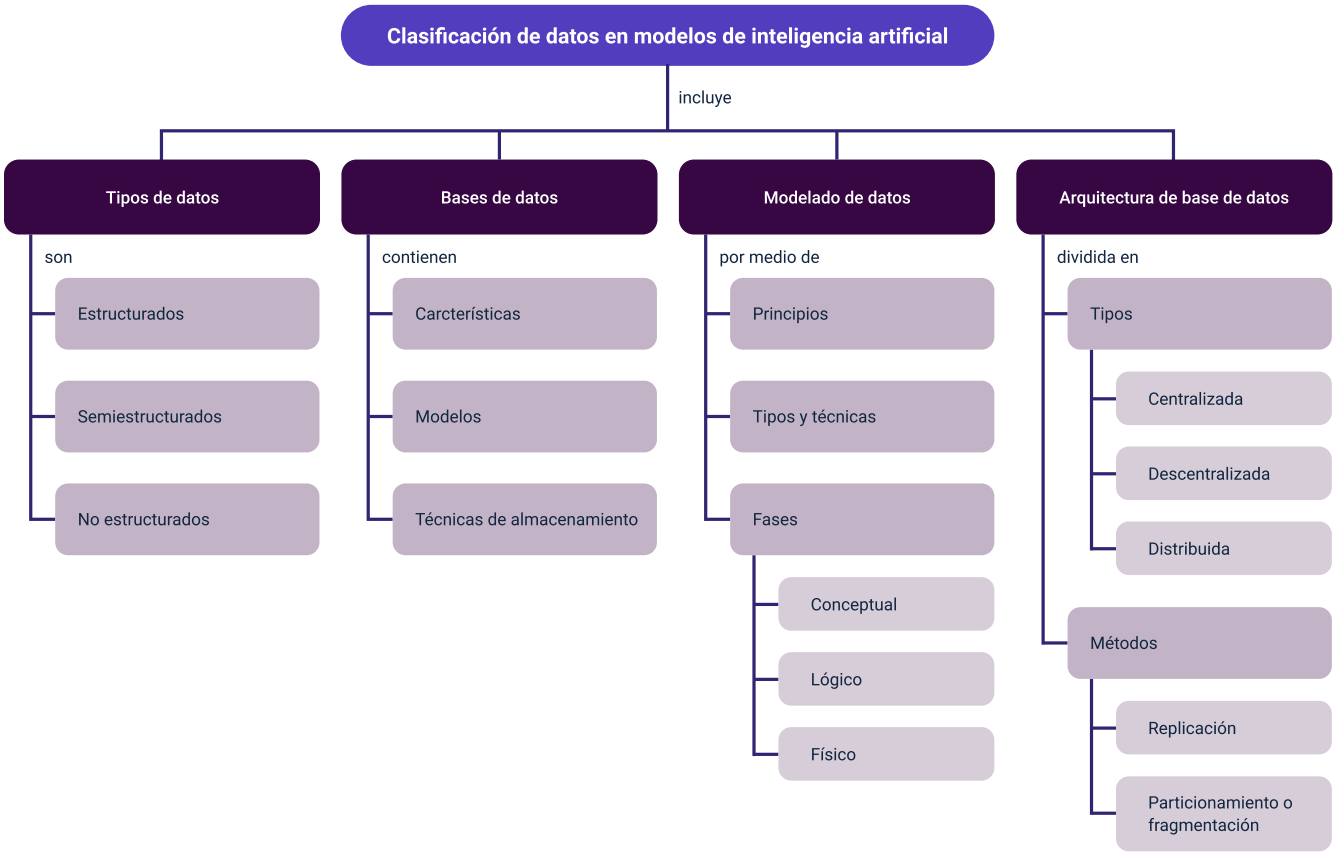
  Se aplica una función hash a la clave de particionamiento y el resultado determina en qué shard se almacenan los datos. Esto distribuye los datos de manera más uniforme, pero dificulta las consultas por rango.

La complejidad de las consultas abarca múltiples shards. El enrutamiento de consultas se vuelve más complicado y las operaciones de JOIN entre shards pueden ser muy ineficientes. El re-particionamiento (re-sharding) cuando se agregan nuevos nodos, también es una operación delicada.

En la práctica, las arquitecturas a gran escala, suelen integrar tanto la replicación como el particionamiento. Cada shard o partición de datos, se replica en múltiples nodos para asegurar tanto la escalabilidad como la alta disponibilidad. Este enfoque es comúnmente adoptado por la mayoría de las grandes bases de datos NoSQL y los servicios de bases de datos en la nube.

Síntesis

Este componente analiza los principios universales que sustentan las aplicaciones encargadas de gestionar grandes volúmenes de datos o que poseen lógica compleja, enfocándose en tres objetivos fundamentales que todo sistema eficiente debe alcanzar: confiabilidad, escalabilidad y mantenibilidad. Para alcanzar estos objetivos, se examina el funcionamiento interno de las bases de datos y las diferentes arquitecturas para el procesamiento de datos.


Clasificación de datos en modelos de inteligencia artificial
incluye
Tipos de datos
son
Estructurados
Semiestructurados
No estructurados
Bases de datos
contienen
Carcterísticas
Modelos
Técnicas de almacenamiento
Modelado de datos
por medio de
Principios
Tipos y técnicas
Fases
Conceptual
Lógico
Físico
Arquitectura de base de datos
dividida en
Tipos
Centralizada
Descentralizada
Distribuida
Métodos
Replicación
Particionamiento o fragmentación

Material complementario

Tema	Referencia	Tipo de material	Enlace del recurso
1. Tipos de datos.	Ecosistema de Recursos Educativos SENA. 2022, (7 de junio). Recursos y herramientas para el análisis efectivo de datos: introducción [Video]. YouTube.	Video.	https://www.youtube.com/watch?v=BP8OeszBSCc
1. Tipos de datos.	Ecosistema de Recursos Educativos SENA. (2023, 25 de marzo). Etapas del procesamiento de datos y métodos estadísticos Introducción [Video]. YouTube.	Video.	https://www.youtube.com/watch?v=ndzj15PQEVw
3. Modelado de datos	Ecosistema de Recursos Educativos SENA. (2022, 23 de diciembre). Modelo de análisis de datos [Video]. YouTube.	Video.	https://www.youtube.com/watch?v=KMRGyi1ZB9k

Glosario

Almacén de datos: sistema optimizado para consultas analíticas y reporte que integra datos históricos de diferentes orígenes, organizados mediante esquemas dimensionales o relacionales para facilitar el análisis y la generación de informes.

Almacén de Modelo (de IA/ML): modelo computacional inspirado en la estructura y funcionamiento de las redes neuronales biológicas del cerebro. Consiste en nodos interconectados ("neuronas") organizados en capas, que procesan información y aprenden a reconocer patrones complejos.

Canalización de datos: secuencia automatizada de procesos para extraer, transformar, validar y cargar datos (ETL/ELT), desde múltiples fuentes hasta el entorno de entrenamiento o producción de modelos de IA.

Entrenamiento (Training): proceso mediante el cual un algoritmo de aprendizaje automático ajusta los parámetros internos de un modelo utilizando un conjunto de datos específico (datos de entrenamiento). El objetivo es que el modelo aprenda a identificar patrones o realizar la tarea deseada con precisión.

Gobernanza de datos: conjunto de políticas, procedimientos y responsabilidades que aseguran la calidad, la seguridad, el cumplimiento normativo y la correcta gestión de los datos a lo largo de todo su ciclo de vida.

Ingeniería de características: conjunto de técnicas para transformar y seleccionar variables derivadas de datos brutos con el fin de mejorar el rendimiento de los modelos de IA. Incluye creación de nuevas características, codificación de categorías y escalado de valores.

Normalización: proceso de organizar los datos en tablas conforme a las formas normales para reducir redundancias y evitar anomalías en inserciones, actualizaciones o eliminaciones.

Referencias bibliográficas

Abiteboul, S., Buneman, P. & Suciu, D. (2000). Data on the web: from relations to semistructured data and XML. Morgan Kaufmann.

Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377-387.

Date, C. J. (2006). An introduction to database systems. Pearson Education India.

Elmasri, R. & Navathe, S. B. (2016). Fundamentals of database systems seventh edition.

Gómez, Á. P., Jalca, J. J. R., García, J. G., Sánchez, O. Q., Parrales, K. M. & Merino, J. M. (2017). Fundamentos sobre la gestión de base de datos (23). 3Ciencias.

Kleppmann, M. (2017). Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems. O'Reilly Media, Inc.

Kroenke, D. M. (2003). Procesamiento de bases de datos. Pearson educación.

Özsu, M. T. & Valduriez, P. (2011). Principles of Distributed Database Systems (3rd ed.). Springer.

Robinson, I., Webber, J. & Eifrem, E. (2015). Graph databases: new opportunities for connected data. O'Reilly Media, Inc.

Silberschatz, A., Korth, H. F. & Sudarshan, S. (2011). Database system concepts.

Silberschatz, A., Korth, H. F., Sudarshan, S., Pérez, F. S., Santiago, A. I. & Sánchez, A. V. (2002). Fundamentos de bases de datos (11). Ciudad de México, México: McGraw-Hill.

Simsion, G. & Witt, G. (2004). Data modeling essentials. Elsevier.

Créditos

Nombre	Cargo	Centro de Formación y Regional
Milady Tatiana Villamil Castellanos	Responsable Ecosistema de Recursos Educativos Digitales (RED)	Dirección General
Diana Rocio Possos Beltrán	Responsable de línea de producción	Centro de Comercio y Servicios - Regional Tolima
Deivis Eduard Ramírez Martínez	Experto temático	Centro de Comercio y Servicios - Regional Tolima
Andrés Felipe Velandia Espitia	Evaluador instruccional	Centro de Comercio y Servicios - Regional Tolima
Oscar Iván Uribe Ortiz	Diseñador web	Centro de Comercio y Servicios - Regional Tolima
Juan Daniel Polanco Muñoz	Diseñador web	Centro de Comercio y Servicios - Regional Tolima
Diego Fernando Velasco Güiza	Desarrollador full stack	Centro de Comercio y Servicios - Regional Tolima
Francisco José Vásquez Suárez	Desarrollador full stack	Centro de Comercio y Servicios - Regional Tolima
Ernesto Navarro Jaimes	Animador y productor audiovisual	Centro de Comercio y Servicios - Regional Tolima
Norma Constanza Morales Cruz	Evaluadora de contenidos inclusivos y accesibles	Centro de Comercio y Servicios - Regional Tolima

Nombre	Cargo	Centro de Formación y Regional
Javier Mauricio Oviedo	Validador y vinculador de recursos educativos digitales	Centro de Comercio y Servicios - Regional Tolima