apuntes-tema-1

DE LOS DATOS AL CONOCIMIENTO

El Big Data se utiliza para gestionar **grandes volúmenes de datos** y **extraer valor** de ellos. Proporciona soluciones a problemas de almacenamiento, procesamiento y análisis de datos masivos. *Objetivos del bid data*

- **Gestionar soluciones a problemas de Big Data:** Resolver problemas de gestión de datos heterogéneos y desesetructurados.
- **Gestionar eficientemente sistemas de almacenamiento:** Almacenar y organizar datos en infraestructuras (sistemas distribuidos o nube)
- **Garantizar integridad de los datos:** Desarrollar mecanismos para garantizar que los datos permanezcan exactos y coherentes toda su vida.
- **Aplicar técnicas de seguimiento y monitorización de sistemas Big Data:** Crear mecanismos para asegurar que los sistemas operen de manera óptima, detectando fallos o problemas en tiempo real.
- **Aprender a validar técnicas de Big Data:** Validar métodos y tecnologías para asegurar que cumplan con los objetivos esperados y funcionen correctamente en diversos entornos.

Datos, información y conocimiento

Los datos son representaciones sintácticas **sin significado** que una vez **procesados** se convierten en **información**. La información se utiliza para **construir conocimiento** cuando se generan reglas, patrones o asociaciones que ayudan en la toma de decisiones. *Diferencias entre datos, información y conocimiento*

- **Datos:** Los datos son hechos crudos, sin procesar, sin contexto y sin significado.
- **Información:** Surge cuando los datos son procesados, organidzados y con contexto.
- **Conocimiento:** Comprensión profunda y estructurada que se desarrolla a partir de la información acumulada y que permite prever, evaluar y tomar decisiones fundamentadas.

IMPORTANCIA DE LOS DATOS EN LA TOMA DE DECISIONES

Evolución en la disponibilidad de los datos Antiguamente obtener datos era complicado esta situación hizo que se utilizara la técnica de **Simulación de Datos** que consiste en recurrir a modelos matemáticos que generaban **datos irreales.** *Rol de la tecnología en la gestión y procesamiento de datos* Con el aumento de datos han surgido desafios:

- **Almacenamiento:** Los sistemas de información distribuida y la computación en la nube permiten almacenar grandes volúmenes de datos.
- **Procesamiento:** Distinguimos dos modalidades:
  - El **procesamiento online** donde los datos se procesan al generarse

- El **procesamiento offline** los datos se analizan posteriormente. **Modelos distribuidos:** Computación distribuida y la nube permiten adquirir recursos de procesamiento bajo demanda **Programación paralela y multiprocesador:** Permiten dividir y procesar datos en múltiples hilos optimizando tiempo y recursos. *Desafío actual: crecimiento exponencial de datos* La mejora de los sensores permite registrar datos en tiempo real. Las mejoras futuras en el **hardware** consisten en conseguir mejorar las capacidades de almacenamiento y en **software** consiste en crear modelos que optimicen el análisis de datos.

Evolución y diversidad de los datos en Big Data

La tecnología a evolucionado para gestionar la cantidad de datos y estos han experimentado una transformación en su cantidad, tipo y formato. *Evolución de los tipos y formatos de datos*

- **Datos tradicionales:** Antiguamente los datos provenían de archivos en formato tabular con instancias y atributos. Los formatos comunes eran las **hojas de cálculo** que facilitaban el análisis de datos organizados pero limitaban la variedad de información disponible.
- **Nuevos tipos de datos:** Los datos han aumentado en volúmen y en formatos más diversos. El texto proviene de fuentes variadas (Pag. web, PDF, RRSS) y ahora se analizan gracias al **procesamiento del lenguaje natural (PLN)**. Las imágenes, audio y vídeo se procesan usando **técnidas de AI.**

Tecnologías y motodologías actuales La disponibilidad de tecnologías avanzadas permiten trabajar con datos en múltiples formatos y escalas, esto permite extraer información más compleja y en tiempo real.

- Las **metodologías de big data** y las **arquitecturas de hardware actuales** definen los procesos específicos que permiten la manipulación y análisis de estos datos en entornos concretos.
- **Soluciones avanzadas en infraestructura** facilitan el manejo eficiente de datos no estructurados, haciendo posible extraer conocimiento aplicable a la toma de decisiones.

SOLUCIONES DE ALMACENAMIENTO BIG DATA

En la era tecnologica actual se generan **petabytes (milloes de gigabytes) de datos al día**. Se estima que el 90% de los datos globales son de los últimos años. Esta era se perfila como la generación del Big Data. El concepto de Big Data engloba tanto los datos disponibles como las herramientas para analizarlos en tiempo real. *Las Vs del Big Data*

- **Volumen**, **variedad** y **velocidad** son las principales pero muchos añaden volatilidad, valor, validez, veracidad, variabilidad, visualización o vulnerabilidad.

Almacenes de datos

Motivación

Los almacenes de datos o Data Warehouses (DW) son **herramientas clave en el Business Intelligence.** Se desarrollaron para ayudar a las empresas a **organizar y analizar** los grandes volúmenes de datos,

transformando los datos en información valiosa para mejorar las decisiones en las áreas de la empresa. Para ello se crearon los **Decision Support Systems (DSS)** herramientas que ayudan a las empresas a tomar decisiones combinando las capacidades de los ordenadores y de las personas. Los almacenes de datos se han convertido en una opción de soporte en el diseño de los sistemas de ayuda a la decision gracias a su rendimiento. Los almacenes de datos se utilizan en ciencias naturales, demografía, salud o educación.

Definición

Los almacenes de datos se crearon para **transformar** los **datos** transaccionales en **información** para obtener conocmiento y tomar decisiones. La *aparición* de los almacenes de datos se debe a:

- **Accesibilidad:** Se debe acceder al DW desde **cualquier dispositivo**. Esto incluye la capacidad de **escalabilidad** del sistema.
- **Integración:** La información proviene de distintas fuentes y formatos. El DW debe integrar todos los datos de forma coherente y sin errores, para ello hay que limpiar los datos para asegurarse de que sean correctos y estén completos.
- **Consultas mejoradas:** Podemos realizar consultas avanzadas, esto permite tomar decisiones estratégicas y operativas.
- **Representación multidimensional:** Los DW proporcionan una forma de visualizar y analizar los datos desde múltiples perspectivas o dimensiones, esto sirve para observar patrones y tendencias en sus datos.

Según *W. Inmon* (padre de los almacenes de datos) lo define como una colección de datos organizada por temas, integrada, variante en el tiempo y no volátil. Esto significa que los almacenes de datos:

- **Están orientados a temas:** Los datos se organizan por temas de interés.
- **Están integrados:** toda la información está almacenada de forma consistente y uniforme.
- **Varían en el tiempo:** los datos reflejan el estado de la empresa en distintos momentos.
- **Son no volátiles:** una vez que los datos se ingresan al DW, permanecen sin modificaciones para garantizar la consistencia de la información.

Los DW se basan en **sistemas de procesamiento analítico en línea (OLAP)**, que están diseñados para analizar, interpretar y tomar decisiones estratégicas basadas en los datos. Y **OLTP (Procesamiento de Transacciones en Línea)** gestiona operaciones y transacciones del día a día.

Diseño e implementación de almacenes de datos

El proceso para construir un almacén de datos se fundamenta en el **proceso ETL** que consiste en extraer datos de diversas fuentes, limpiarlos y transformarlos antes de almacenarlos en el almacén de datos. Las fases del ETL son:

- **Extracción:** Se **recogen** los datos de las distintas fuentes. Al inicializar el almacén, se usa una *"extracción estática"* que toma una instantánea de los datos actuales. Posteriormente, se utiliza *"extracción incremental"* para actualizar los datos, obteniendo solo los cambios desde la última actualización

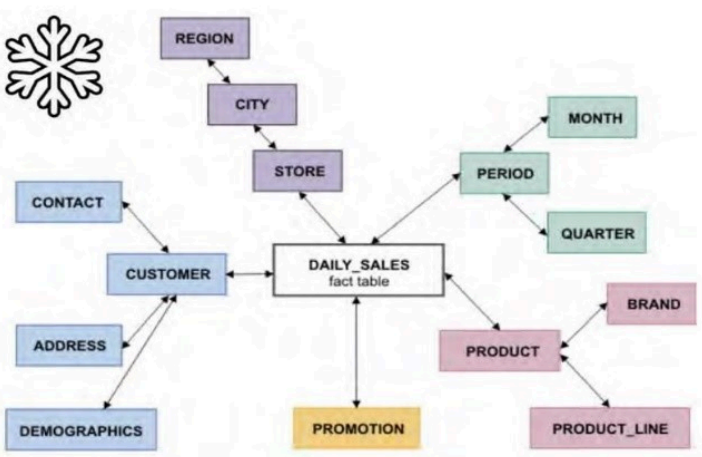
- **Transformación:** Los datos extraídos son **limpiados** y **adaptados** para un formato específico del almacén de datos. Esto implica mejorar su coherencia y completar la información faltante. Esto también incluye: 1.*Limpieza de datos:* detecta y soluciona problemas como datos incompletos, erróneos, o duplicados. También maneja valores perdidos mediante eliminación, asignación de valores fijos o de referencia, e imputación de datos. 2. *Estandarización y formateo:* Incluye la conversión de formatos, unificación de códigos y transformación de representaciones. 3. *Corrección e integración:* arregla errores y asegura la consistencia entre fuentes diversas. 4. *Reducción de dispersión:* escala y centra los datos para facilitar su análisis. 5. *Discretización:* convierte variables continuas en valores discretos cuando el análisis lo requiere.
- **Carga:** Es la última fase y consiste en **volcar los datos transformados al almacén**. Las cargas pueden ser:
    i. *Refresco completo:* sustituye todos los datos existentes, útil para cargas iniciales.
    ii. *Actualización incremental:* añade solo los datos nuevos, ideal para actualizaciones regulares.

Diseño en estrella Es una estructura sencilla y muy utilizada. Tiene una **tabla central de hechos** rodeada de varias **tablas de dimensiones**.

- **Tabla de hechos:** es el núcleo del diseño en estrella y almacena los datos cuantitativos que se desean analizar. Esta tabla suele tener claves ajenas (foreign keys) que la conectan con las tablas de dimensiones.
- **Tablas de dimensiones:** rodean a la tabla de hechos y contienen la **información descriptiva** que permite analizar los hechos. Estas tablas son generalmente más pequeñas y están desnormalizadas. *Ventajas*
- Facilita la comprensión del modelo y permite realizar consultas rápidamente.
- Reduce la complejidad en las consultas y mejora el rendimiento. *Desventajas*
- Puede haber redundancia de datos en las tablas de dimensiones, ya que se mantiene información duplicada.

Diseño en copo de nieve El diseño en copo de nieve es una variación del diseño en estrella que añade un nivel de normalización en tablas de dimensiones, que se dividen en varias tablas adicionales para evitar redundancia.

- **Tabla de hechos:** la tabla de hechos contiene los datos numéricos y cuantitativos principales del análisis.
- **Tablas de dimensiones normalizadas:** las dimensiones están organizadas en múltiples tablas normalizadas. *Ventajas*
- Menor redundancia en las tablas de dimensiones, lo que ahorra espacio de almacenamiento.
- Mayor consistencia de datos, ya que la normalización evita duplicación de información. *Desventajas*
- Las consultas pueden ser más complejas, ya que requieren combinar varias tablas de dimensiones.
- El diseño en copo de nieve es más difícil de entender y de construir en comparación con el diseño en estrella.



Arquitectura almacenes de datos

La arquitectura de un almacén de datos se describe en términos de capas y componentes que facilitan la organización, almacenamiento, y acceso eficiente a grandes volúmenes de datos. Esta arquitectura asegura que los datos se almacenen y estructuren para facilitar su análisis y soportar la toma de decisiones. Un esquema común de la arquitectura de un almacén de datos sería:

- **Capa de origen:** Recoge los datos de diferentes fuentes que se integrarán en el almacén de datos.
- **Capa de Extracción, Transformación y Carga (ETL):** Aqui los datos son limpiados, transformados, y estandarizados antes de cargarse en el almacén de datos. La ETL garantiza que los datos sean consistentes y de calidad.
- **Capa de almacenamiento de datos:** los datos se organizan y almacenan en una estructura de datos optimizada.
- **Capa de presentación y acceso:** ofrece herramientas para que los usuarios puedan acceder y analizar los datos. Este tipo de arquitectura permite una gestión eficiente y escalable de los datos, facilitando el acceso a información histórica y consolidada para el análisis y la toma de decisiones.

BASES DE DATOS DOCUMENTALES

Motivación Las BBDD noSQL surgen para dar respuesta a las limitaciones de las BBDD relacionales. En los sistemas de **bases de datos relacionales**, las transacciones garantizan la consistencia y escalabilidad de las operaciones mediante las características del modelo ACID:

- **Atomicidad (A):** Asegura que cada transacción se realize completamente.
- **Consistencia (C):** Garantiza que cualquier transacción lleva la base de datos de un estado válido a otro.
- **Aislamiento (I):** Permite que las transacciones no interfieran entre sí.
- **Durabilidad (D):** Asegura que los cambios de una transacción completada se conservan en todos los casos.

Las **bases de datos NoSQL** suelen emplear un modelo más flexible conocido como *BASE*. Este modelo se adapta mejor a los sistemas distribuidos y no relacionales al priorizar la disponibilidad y escalabilidad de los datos:

- **Disponibilidad básica (Basic Availability):** El sistema responde a cada solicitud, incluso si los datos aún no están completamente consistentes o si se ha producido algún fallo.
- **Estado suave (Soft-state):** La consistencia en NoSQL es eventual, lo que significa que el estado del sistema puede cambiar con el tiempo, aun sin nuevas entradas de datos.
- **Consistencia eventual (Eventual Consistency):** Los datos eventualmente se vuelven consistentes una vez que dejan de recibir nuevas actualizaciones, permitiendo que el sistema siga operando sin verificar la consistencia en cada transacción. El enfoque BASE permite a las BBDD NoSQL ofrecer disponibilidad y rendimiento en contextos de datos masivos y aplicaciones de gran escala.

Definición Una BBDD documental es un tipo de base de datos **NoSQL** diseñada para almacenar, recuperar y gestionar datos en forma de **documentos**, (JSON, BSON o XML). Cada documento contiene **datos semi-estructurados** y suele incluir **información compleja y anidada**, como listas y objetos. Este enfoque permite una gran **flexibilidad**.

MongoDB: ejemplo representativo de BBDD documental Es una base de datos **NoSQL** de código abierto, orientada a documentos y desarrollada en C++, compatible con diferentes plataformas y sistemas operativos. Permite manejar grandes volúmenes de datos y su esquema flexible que permite un desarrollo ágil en aplicaciones modernas. Las características básicas son:

- **Esquema flexible:** Permite almacenar datos en documentos JSON o BSON sin un esquema fijo
- **Alto rendimiento:** MongoDB está optimizado para lecturas y escrituras rápidas.
- **Escalabilidad horizontal:** Divide datos entre varios servidores, esto facilita el crecimietno de manera escalable y distribuida
- **Alta disponibilidad:** Soporta la replicación de datos lo que permite la recuperación rápida y disponibilidad continua incluso en fallos.
- **Consultas flexibles y Ad-Hoc:** Potente capacidad de consultas permitiando búsquedas por campos específicos, rangos y expresiones regulares.
- **Indexación:** Soporta la creación de índices primarios y secundarios para acelerar el acceso a los datos
- **Almacenamiento de Archivos con GridFS:** Permite almacenar archivos de gran tamaño (Fotos, Vídeos) mediante *GridFS* (distribuye los datos de archivo entre múltiples documentos.
- **Compatibilidad con JavaScript en el lado del servidor:** Permite ejecutar consultas y operaciones en JavaScript

Estas características hacen que MongoDB sea una opción sólida para aplicaciones web, móviles, análisis en tiempo real y sistemas que requieren escalabilidad, flexibilidad y alto rendimiento.

BASES DE DATOS ORIENTADOS A GRAFOS

Que es un grafo

Es una **estructura de datos** que consta de un conjunto de **nodos** (vértices) y un conjunto de **aristas** que conectan pares de nodos. Se utilizan en ciencias de la computación y matemáticas. Los nodos representan **entidades individuales**, mientras que las aristas representan las relaciones entre los nodos. Los grafos pueden ser dirijidos o no dirigidos, ponderados (las aristas tienen valor), distancia o relación. Los grafos son útilies en la **representación de redes** de comunicación, modelado de entidades en BBDDs, la representación de dependencias entre tareas en programación y la resolución de problemas de rutas más cortas y problemas de flujo en algoritmos.

Redes sociales: lugares donde los grafos son más que utilizados Los grafos ayudan a comprender la estructura de la red social, identificar comunidades de usuarios, analizar la difusión de información y contenido, y recomendar conexiones y contenido relevante. Algunas maneras específicas en las que las redes sociales pueden utilizar grafos:

- **Análisis de redes sociales:** Mediante el análisis de la estructura del grafo, esto puede ayudar a comprender mejor cómo se conectan los usuarios, cómo se propagan las ideas y la información dentro de la red.
- **Recomendaciones personalizadas:** los grafos pueden utilizarse para recomendar conexiones y contenido relevante a los usuarios en función de sus interacciones y las conexiones de sus amigos.
- **Detección de anomalías y spam:** al analizar la estructura del grafo y los patrones de comportamiento de los usuarios, las redes sociales pueden detectar actividades sospechosas, como cuentas falsas, bots o comportamientos inusuales. Esto ayuda a mejorar la seguridad y la integridad de la red social.
- **Análisis de tendencias y opiniones:** al analizar los patrones de interacción y las discusiones en la red social, se pueden identificar tendencias y opiniones predominantes.
- **Optimización de publicidad dirigida:** al comprender las conexiones entre los usuarios y sus intereses, las redes sociales pueden optimizar la segmentación de la publicidad y mejorar la efectividad de las campañas publicitarias al dirigirse a audiencias específicas de manera más precisa.

Usando Neo4J para dar forma a mis grafos

Neo4j es una base de **datos de grafo** altamente escalable y de alto rendimiento que se utiliza para almacenar, recuperar y administrar datos altamente conectados. *Características*

- **Modelo de datos de grafo nativo:** Está diseñado específicamente para trabajar con datos altamente relacionales, lo que lo hace ideal para situaciones en las que las relaciones entre los datos son tan importantes como los propios datos.
- **Lenguaje de consulta Cypher:** Neo4j utiliza el lenguaje de consulta *Cypher*, que permite a los usuarios trabajar con patrones de grafo de manera intuitiva y poderosa.
- **Alta escalabilidad y rendimiento:** Neo4j es conocido por su capacidad de escalar y gestionar grandes conjuntos de datos
- **Flexibilidad en el modelado de datos:** Debido a su naturaleza de grafo flexible, Neo4j permite a los usuarios modelar datos de manera más intuitiva y natural, lo que facilita la representación de relaciones complejas entre entidades.
- **Aplicaciones de uso común:** Neo4j se utiliza en una variedad de aplicaciones, que van desde redes sociales y análisis de redes hasta recomendaciones de productos y detección de fraudes.

Primeros pasos con Neo4J

```
Creamos un nodo
  CREATE (c1:City {name: 'Ciudad A'})
  CREATE (c2:City {name: 'Ciudad B'})

Mostramos el grafo
  MATCH (n ) RETURN (n )

Creamos relaciones entre ciudades
  MATCH (x:City), (y:City) WHERE x.name = 'Ciudad A' AND y.name = 'Ciudad B' CREATE (x )-
[:CONNECTED {distance: 200}]→(y )

Guardamos el grafo
  CALL gds.graph.project('myGraph1', 'City', 'CONNECTED', { relationshipProperties:
'distance' })

Recorrer un grafo
  BFS (en anchura)
    CALL gds.bfs.stream('myGraph1', { sourceNode: 0 })
    YIELD path
    RETURN path;

  DFS (en profundidad)
    CALL gds.dfs.stream('myGraph1', { sourceNode: 0 })
    YIELD path
    RETURN path;

Para mostrarlo en forma de tabla
  BFS (en anchura)
    CALL gds.bfs.stream('myGraph1', { sourceNode: 0 })
    YIELD path
    UNWIND [ n in nodes(path) | n.name ] AS tags
    RETURN tags;

  DFS (en profundidad)
    CALL gds.dfs.stream('myGraph1', { sourceNode: 0 })
    YIELD path
    UNWIND [ n in nodes(path) | n.name ] AS tags
    RETURN tags;

Obtener el camino mínimo
  MATCH (start:City {name: 'Ciudad A'}), (end:City {name: 'Ciudad B'})
  CALL gds.shortestPath.dijkstra.stream('myGraph1', {
    sourceNode: start,
    targetNode: end,
    relationshipWeightProperty: 'distance'
  })
  YIELD path
  RETURN path

Obtener el camino mínimo poniendolo en una tabla
  MATCH (start:City {name: 'Ciudad A'}), (end:City {name: 'Ciudad B'})
  CALL gds.shortestPath.dijkstra.stream('myGraph1', {
    sourceNode: start,
```

```
    targetNode: end,
    relationshipWeightProperty: 'distance'
  })
  YIELD path
  UNWIND [ n in nodes(path) | n.name ] AS tags
  RETURN tags

Borrar un grafo
  CALL gds.graph.drop('myGraph1');

Borrar el proyecto
  MATCH (n ) DETACH DELETE n;

Calcular la centralidad de grado
  CALL gds.degree.stream('myGraph')
  YIELD nodeId, score
  RETURN gds.util.asNode(nodeId).name AS
  name, score AS followers
  ORDER BY followers DESC, name DESC

Calculamos intermediación
  CALL gds.betweenness.stream('myGraph')
  YIELD nodeId, score
  RETURN gds.util.asNode(nodeId).name AS
  name, score
  ORDER BY name ASC

Calculamos el conteo de triangulos
  CALL
  gds.triangleCount.stream('myGraph')
  YIELD nodeId, triangleCount
  RETURN gds.util.asNode(nodeId).name
  AS name, triangleCount
  ORDER BY triangleCount DESC

Calculamos el coeficiente local de clustering
  CALL
  gds.localClusteringCoefficient.stream
  ('myGraph')
  YIELD nodeId,
  localClusteringCoefficient
  RETURN gds.util.asNode(nodeId).name
  AS name, localClusteringCoefficient
  ORDER BY localClusteringCoefficient
  DESC

PREDICCIONES DE ENLACES
Método vecinos comunes
  MATCH (p1:Person {name: 'Michael'})
  MATCH (p2:Person {name: 'Karin'})
  RETURN
  gds.alpha.linkprediction.commonNeighbors(p1, p2) AS score

Método adhesión preferencial
  MATCH (p1:Person {name: 'Michael'})
  MATCH (p2:Person {name: 'Karin'})
```

```
  RETURN
  gds.alpha.linkprediction.preferentialAttachment(p1, p2) AS score

Método de la asignación de recursos
  MATCH (p1:Person {name: 'Michael'})
  MATCH (p2:Person {name: 'Karin'})
  RETURN
  gds.alpha.linkprediction.resourceAllocation(p1, p2) AS score
```