0

Javier Plaza Sisqués

Universidad europea  Grandes volúmenes de datos

Apuntes grandes volumenes de datos

# 

# Contenido

[0](#_Toc157094620)

[Contenido 1](#_Toc157094621)

[Tema 2 - Administración de SSOO Linux 4](#_Toc157094622)

[Características de Linux 4](#_Toc157094623)

[Distribuciones 4](#_Toc157094624)

[Libres 4](#_Toc157094625)

[Privativas 4](#_Toc157094626)

[Sistema de ficheros 4](#_Toc157094627)

[Jerarquía de ficheros 4](#_Toc157094628)

[Tipos de archivos 5](#_Toc157094629)

[Trabajando con directorios y ficheros 5](#_Toc157094630)

[Usuarios en Linux 5](#_Toc157094631)

[Control de acceso 5](#_Toc157094632)

[Tipos de usuarios 5](#_Toc157094633)

[Usuarios y grupos lógicos 5](#_Toc157094634)

[Usuarios y grupos físicos 6](#_Toc157094635)

[Creación de usuarios en CentOS 6](#_Toc157094636)

[Permisos y privilegios de usuarios 6](#_Toc157094637)

[Gestión de usuarios, grupos y permisos 6](#_Toc157094638)

[Procesos 6](#_Toc157094639)

[Tema 3 – Linux Bash Shell 7](#_Toc157094640)

[Introducción 7](#_Toc157094641)

[Fundamentos de Bash Shell 7](#_Toc157094642)

[Tipos de Shell 7](#_Toc157094643)

[Tipos de streams 7](#_Toc157094644)

[Pipes 7](#_Toc157094645)

[Alias 7](#_Toc157094646)

[Variables 7](#_Toc157094647)

[Alcance de variables 7](#_Toc157094648)

[Variables de entorno predefinidas 7](#_Toc157094649)

[Variables de entorno definidas por el usuario 8](#_Toc157094650)

[Bash Shell scripting 8](#_Toc157094651)

[Ejecución se scripts Bash Shell 8](#_Toc157094652)

[Variables especiales 8](#_Toc157094653)

[Comandos útiles en Shell scripting 8](#_Toc157094654)

[Test 8](#_Toc157094655)

[Read 9](#_Toc157094656)

[Script 9](#_Toc157094657)

[Estructuras de control 9](#_Toc157094658)

[Sentencia if 9](#_Toc157094659)

[Sentencia for 9](#_Toc157094660)

[Sentencia while 9](#_Toc157094661)

[Tema 4 - Introducción a las bases de datos de Oracle 10](#_Toc157094662)

[1. Presentación 10](#_Toc157094663)

[2. Noción de instancia y de base de datos 10](#_Toc157094664)

[3. Tipos de BBDD Oracle 10](#_Toc157094665)

[4. Ficheros de una BBDD Oracle 10](#_Toc157094666)

[5. Componentes BBDD Oracle 10](#_Toc157094667)

[6. Componentes Instancia BBDD Oracle 10](#_Toc157094668)

[7. Diccionario de datos 10](#_Toc157094669)

[8. Administrador de la base de datos 10](#_Toc157094670)

[Tema 5 – Oracle NET 12](#_Toc157094671)

[1. Introducción 12](#_Toc157094672)

[2. Funciones de Oracle NET 12](#_Toc157094673)

[3. Resolución de nombres de servicio de Oracle 12](#_Toc157094674)

[3.1. Local Naming 12](#_Toc157094675)

[3.2. Easy Connect 12](#_Toc157094676)

[3.3. Directory Naming 12](#_Toc157094677)

[3.4. External Naming 12](#_Toc157094678)

[4. Configuración del servidor 12](#_Toc157094679)

[4.1. Database Configuration Assistant 12](#_Toc157094680)

[4.2. Network Configuration Assistant 12](#_Toc157094681)

[4.3. Oracle Net Manager 13](#_Toc157094682)

[5. Configuración de la conexión de clientes 13](#_Toc157094683)

[5.1. Configuración de resolución de nombres de servicio 13](#_Toc157094684)

[5.2. Pruebas de conexión de cliente 13](#_Toc157094685)

[Tema 6 – Administración de BBDD Oracle 14](#_Toc157094686)

[1. Introducción 14](#_Toc157094687)

[2. Historia del lenguaje SQL 14](#_Toc157094688)

[3. Oracle SQL 14](#_Toc157094689)

[4. PL/SQL 14](#_Toc157094690)

[4.1. Hello World en PL/SQL 14](#_Toc157094691)

[4.2. Funcionalidades de PL/SQL 14](#_Toc157094692)

[5. Herramientas de administración de BBDD Oracle 14](#_Toc157094693)

[5.1. Oracle Enterprise Manager 14](#_Toc157094694)

[5.2. SQL\*PLUS 15](#_Toc157094695)

[5.3. Oracle Live SQL 15](#_Toc157094696)

[5.4. Oracle SQL Developer 15](#_Toc157094697)

[Tema 7 – Cloud computing 16](#_Toc157094698)

[1. Introducción 16](#_Toc157094699)

[2. Fundamentos de Cloud Computing: 16](#_Toc157094700)

[2.1. Tecnologías de Cloud Computing 16](#_Toc157094701)

[2.2. Arquitectura Cloud Computing 16](#_Toc157094702)

[2.3. Características Cloud Computing 16](#_Toc157094703)

[3. Tipos de Cloud Computing 17](#_Toc157094704)

[3.1. Modelos de Cloud Computing 17](#_Toc157094705)

[3.2. Servicios de Cloud Computing 17](#_Toc157094706)

[4. Proveedores de Cloud Computing 17](#_Toc157094707)

[4.1. OpenStack 17](#_Toc157094708)

[4.2. Amazon Web Services (AWS) 17](#_Toc157094709)

[4.3. Google Cloud Platform 17](#_Toc157094710)

[4.4. Microsoft Azure 18](#_Toc157094711)

# Tema 1 – Sistemas de computación distribuida

## 1. Introducción

Los sistemas distribuidos son comunes pero desafiantes. Este tema describe estos sistemas, sus objetivos de diseño y tipos comunes. Son fundamentales en la computación a media y gran escala, requiriendo conocimiento de componentes para un despliegue eficiente.

## 2. Definiciones

Tanenbaum y Coulouris definen sistemas distribuidos. Se introducen conceptos clave: programa, proceso, datos, red de ordenadores y protocolo.

## 3. Inicios

Desde 1945 hasta mediados de los 80, las computadoras eran independientes. Avances en microprocesadores y redes de alta velocidad cambiaron esto.

## 4. Microprocesadores

El desarrollo de microprocesadores, desde 8 bits hasta multinúcleo, permitió la adaptación y desarrollo de programas para explotar el paralelismo.

## 5. Redes informáticas de alta velocidad

Las redes LAN y WAN posibilitaron la conexión de miles de máquinas. La miniaturización llevó a dispositivos como teléfonos inteligentes y nano computadoras.

## 6. Diseño

Aunque es posible construir sistemas distribuidos, deben cumplir cuatro metas: accesibilidad, ocultamiento de la distribución, apertura y escalabilidad. Se deben considerar aspectos como:

### 6.1. Aspectos de Diseño:

* **Escalabilidad:** Medida por usuarios, objetos, servicios, distancia geográfica y administrativa.
* **Extensibilidad:** Apertura a nuevos estándares, aplicaciones y tecnologías.
* **Heterogeneidad:** Considerar diferentes plataformas de hardware, software, lenguajes y velocidad.
* **Tolerancia a fallos:** Identificar y gestionar fallos para asegurar disponibilidad y consistencia.
* **Seguridad:** Incluir confidencialidad, integridad y autenticación.
* **Control de concurrencia:** Coordinar acciones entre diferentes sistemas para evitar conflictos.

### 6.2. Técnicas para el Aumento de Escalabilidad:

* **Replicación de datos:** Crear múltiples copias del mismo dato.
* **Replicación de procesos:** Ejecutar múltiples instancias de la misma computación.
* **Caching y problema de consistencia de datos:** Almacenar en caché datos para mejorar el rendimiento, pero gestionar la consistencia.
* **Distribución de carga:** Equilibrar la carga entre nodos del sistema distribuido.
* **Calidad de servicio:** Ofrecer garantías de servicio en términos de velocidad, seguridad y disponibilidad.

## 7. Consistencia, Disponibilidad y Tolerancia

El teorema CAP establece que es imposible que un sistema distribuido proporcione simultáneamente consistencia, disponibilidad y tolerancia a partición.

## 8. Arquitecturas Distribuidas

Sistemas Clusters y Supercomputadores: Agrupación de máquinas para resolver problemas distribuidamente.

* **Grid Computing:** Agregación y compartición de recursos distribuidos.
* **Cloud Computing:** Recursos virtualizados, escalabilidad y pago por uso.

## 9. Almacenamiento Distribuido

Tendencias incluyen almacenamiento en la nube, almacenamiento secundario, almacenamiento en capas y Storage-as-a-Service. La elección de tecnologías y topologías de interconexión es crucial.

# Tema 2 – Bases de datos NoSQL

## 1. Introducción

En el desarrollo tecnológico, las bases de datos (BD) son fundamentales para almacenar y gestionar datos. Las BD relacionales han dominado hasta ahora, pero surgen limitaciones con datos no estructurados. Las bases de datos NoSQL emergen como alternativas, permitiendo manejar datos de manera más flexible.

## 2. Historia de las bases de datos NoSQL

A medida que la web 2.0 avanzaba, grandes empresas notaron limitaciones en las BD relacionales. Surge la necesidad de sistemas que escalen horizontalmente y manejen grandes volúmenes de datos en tiempo real. Las primeras BD, previas a las relacionales, presentaban problemas de duplicidad y consistencia.

## 3. Características de las bases de datos NoSQL

* **Escalabilidad:** Las BD NoSQL se adaptan eficientemente a diferentes niveles de carga de trabajo, escalando horizontalmente.
* **Coste:** Muchas BD NoSQL son gratuitas, lo que reduce los costes de implementación.
* **Flexibilidad:** Permiten manejar datos semiestructurados o sin estructura, adaptándose a cambios en la estructura de los datos.
* **Disponibilidad:** Diseñadas para operar distribuidamente, manteniendo la disponibilidad incluso si un nodo falla.

### 3.1. Bases de datos distribuidas

No todas las BD NoSQL son distribuidas, pero muchas lo son para cumplir con las demandas de escalabilidad y disponibilidad. Deben garantizar la persistencia, consistencia y disponibilidad de la información.

### 3.2. Teorema del CAP

Afirma que en un sistema distribuido no se puede garantizar simultáneamente consistencia, disponibilidad y tolerancia al particionado de los datos.

### 3.3. ACID vs. BASE

* **ACID:** Atomicidad, Consistencia, Aislamiento y Durabilidad.
* **BASE:** Básicamente disponible, Estado suave, Consistencia eventual.

## 4. Tipos de bases de datos NoSQL

* **Clave/Valor:** Almacenan datos como pares de clave y valor.
* **Documentos:** Almacenan documentos semiestructurados.
* **Familia de columnas:** Organizan datos en filas y columnas, similar a las BD relacionales.
* **Grafos:** Almacenan datos como nodos y relaciones entre ellos, ideal para representar redes complejas.

# Tema 3 - Bases de datos clave-valor

## 1. Bases de datos NoSQL del tipo clave-valor

### 1.1. Definición

Una base de datos clave-valor es un tipo de base de datos NoSQL que utiliza el método clave-valor para almacenar los datos. Los datos se almacenan de forma simple como una colección de pares clave-valor.

### 1.2. La clave

La clave debe ser única para los pares clave-valor, permitiendo el acceso al valor asociado. En Redis, la longitud máxima de la clave es 512MB y puede ser cualquier secuencia binaria, desde texto hasta imágenes.

### 1.3. El valor

El valor puede ser de cualquier tipo, como texto, imagen, código de programación, lista, etc.

### 1.4. Características

Las características principales de las bases de datos de clave-valor son:

* Gran escalabilidad
* Alto rendimiento
* Fácil manejo
* Alta disponibilidad
* Redundancia

### 1.5. Usos de la base de datos

Algunos usos de la base de datos clave-valor incluyen almacenar sesiones de usuario, carritos de compra, caché de datos de acceso y servicio de anuncios en tiempo real.

## 2. Ejemplo de BD clave-valor: Redis

### 2.1. Historia

Redis, cuyo nombre significa Remote Dictionary Server, comenzó como un proyecto para mejorar la escalabilidad del startup de Salvatore Sanfilippo. Fue lanzado con licencia open source en mayo de 2009 y desde junio de 2015, su desarrollo está a cargo de la empresa Redis Labs.

### 2.2. Descripción

Redis es un sistema de almacenamiento de estructuras de datos en memoria utilizado como base de datos, caché y distribuidor de mensajes. Soporta diversos tipos de datos y ofrece características como replicación, scripting de LUA, reemplazo LRU, transacciones y persistencia en disco.

## 3. Modelo de datos

Redis ofrece diversos tipos de datos como strings, listas, conjuntos, conjuntos ordenados, hashes y HyperLogLogs. También cuenta con soporte para transacciones y programación de módulos.

## 4. Instalación y Uso

La instalación de Redis implica descargar los archivos fuente, compilarlos y copiar los archivos de instalación en el directorio /usr/local/bin. Redis puede ejecutarse con la configuración por defecto o con un archivo de configuración específico. Se pueden utilizar comandos como SET, GET, DEL, INCR, LPUSH, RPUSH, LRANGE, SADD, SREM, SINTER, SCARD, entre otros, para realizar operaciones básicas y avanzadas en Redis.

* **SET clave valor:** Este comando se utiliza para almacenar un valor en una clave en Redis. Por ejemplo, SET nombre "Juan" almacenaría el valor "Juan" en la clave "nombre".
* **GET clave:** Se utiliza para obtener el valor almacenado en una clave específica. Por ejemplo, GET nombre devolvería "Juan" si la clave "nombre" tiene asignado ese valor.
* **DEL clave:** Elimina la clave y su valor asociado de la base de datos Redis. Por ejemplo, DEL nombre eliminaría la clave "nombre" y su valor.
* **INCR clave:** Incrementa el valor numérico almacenado en la clave en 1. La clave debe contener un valor numérico o estar vacía. Por ejemplo, si la clave "contador" tiene el valor 5, al ejecutar INCR contador, su valor se incrementaría a 6.
* **LPUSH clave valor [valor ...]:** Agrega uno o más valores al principio de una lista. Por ejemplo, LPUSH lista1 "a" "b" "c" agregaría los valores "a", "b" y "c" al principio de la lista "lista1".
* **RPUSH clave valor [valor ...]:** Similar a LPUSH, pero agrega los valores al final de la lista.
* **LRANGE clave inicio fin:** Devuelve una lista de elementos de la lista especificada, en el rango indicado por los índices de inicio y fin. Por ejemplo, LRANGE lista1 0 -1 devolvería todos los elementos de la lista "lista1".
* **SADD clave miembro [miembro ...]:** Agrega uno o más miembros a un conjunto. Si el conjunto no existe, se crea. Por ejemplo, SADD conjunto1 "a" "b" "c" agregaría los elementos "a", "b" y "c" al conjunto "conjunto1".
* **SREM clave miembro [miembro ...]:** Elimina uno o más miembros de un conjunto. Por ejemplo, SREM conjunto1 "a" eliminaría el elemento "a" del conjunto "conjunto1".
* **SINTER clave [clave ...]:** Devuelve la intersección de conjuntos especificados por las claves proporcionadas. Es decir, los elementos que aparecen en todos los conjuntos. Por ejemplo, SINTER conjunto1 conjunto2 devolvería los elementos comunes entre "conjunto1" y "conjunto2".
* **SCARD clave:** Devuelve el número de elementos en un conjunto. Por ejemplo, SCARD conjunto1 devolvería la cantidad de elementos en el conjunto "conjunto1".
* **SMEMBERS clave:** Este comando devuelve todos los miembros de un conjunto especificado por la clave. Por ejemplo, si tienes un conjunto llamado "conjunto1" con los elementos "a", "b" y "c", al ejecutar SMEMBERS conjunto1, obtendrás como resultado los elementos "a", "b" y "c". Este comando es útil para recuperar todos los elementos presentes en un conjunto en Redis.

# Tema 4 - Base de datos documentales

## 1. Introducción

Las bases de datos documentales son un tipo de BD no relacional diseñada para datos semiestructurados como documentos. Estos documentos suelen estar en formato JSON o XML, proporcionando flexibilidad en la organización y almacenamiento de datos.

## 2. Desarrollo

Ofrecen un desarrollo ágil y adaptativo, evitando retrasos en la gestión de la BD. La facilidad de uso del formato JSON es una ventaja sobre las bases de datos relacionales.

## 3. Utilidad de las Bases de Datos Documentales

Son ideales para comercios con productos que tienen características y atributos únicos. La administración eficiente de atributos por producto es una ventaja clave.

## 4. Ejemplo de Base de Datos Documental: MongoDB

MongoDB almacena datos en documentos BSON, permitiendo agruparlos en colecciones. Ofrece funciones Map y Reduce y es escalable. Características incluyen consultas por campos, índices, replicación primaria y secundaria, escalabilidad y sistema de archivos.

## 5. Características de MongoDB

* Consultas por campos
* Índices
* Replicación primaria y secundaria
* Escalabilidad
* Sistema de archivos
* Agrupación y MapReduce