2 EXAMEN

· 4 PROBLEMAS IMP

1) PRODUCTOR Y CONSUMIDOR:

El modelo de productor y consumidor se utiliza en programación concurrente para resolver el problema de espera cuando el productor y el consumidor tienen diferentes velocidades de procesamiento. Los beneficios de este modelo incluyen el *equilibrio* entre productores y consumidores y la eliminación de la necesidad de que estén directamente conectados. Se puede utilizar una cola para comunicarse entre el productor y el consumidor o pipe. El uso de la cola evita la necesidad de cerraduras y se asegura de que los datos se procesen en orden.

El productor aporta unos recursos que el consumidor usa.

- PROBLEMAS ----> Si el prooductor produce + lento de lo que consume.
  ----> El consumidor consume + rapido de lo q produce.

- SOLUCIÓN: Estos problemas los arreglamos con una *gestión de recursos* para evitar el interbloqueo, exclusión mutua y evitar q la latencia no sea muy grande. Por lo tanto, usaremos sincronía para que estén al mismo nivel (en equilibrio).

2) CENA DE LOS FILÓSOFOS

Es una metáfora de como se acceden a los recursos. Son 5 filósofos que se sientan en una mesa redonda y tienen que compartir 5 tenendores para poder comer. El problema surge al coordinar varios procesos que *compiten por recursos* compartidos ---> cuando intentan tomar el tenedor de la izq y el de la derecha simultáneamente.

3) BARBERO DURMIENTE

Problema en el cual solo se puede dar solución a 1 . El barbero solo puede afeitar a una persona, x lo tanto, si no hay nadie en la barberia: duerme, si hay alguien en la silla lo afeita (se despierta) ,si hay alguien en la sala d espera lo afeita y si esta llena la sala d espera vuelve otro dia.
El barbero solo puede afeitar a 1.
Cola, orden, gente esperando.

4) LOS FUMADORES

Hay un grupito de personas dnd cada uno tiene determinados recursos pero no son suficientes ---> Necesitan recursos entre ellos.
Cada uno tenga los recursos necesarios ----> OBJ: Sincronizarse para compartir los recursos y obtener el resultado.
ALL-IN: todos los recursos para 1 procesoo.

· MANIPULACIÓN DE BASES DE DATOS RELACIONALES Y NO

INTRODUCCIÓN:

Una base de datos es un software que permite almacenar, manipular y buscar información de forma optimizada e inteligente. El almacenamiento de información en un archivo XML puede ser suficiente para una pequeña cantidad de datos.
Una base de datos es un espacio de almacenamiento de datos que debe ser confiable y coherente.

El paradigma más utilizado de las bases de datos es el "relacional". La información se agrupa en "tablas" (filas y columnas) unidas por enlaces, donde cada columna (campos) es un atributo y cada fila (registros) es una instancia.

Lenguaje de consultas sobre la base de datos: ----> **; EN LENGUAJE SQL**
- Select * from tabla**;** (devuelve toda la info)
- Insert into tabla (, ,) values (, ,)**;**
- Delete from tabla where ID = 1**;**
- Update tabla set nombre =""**;**

Modelos de Base de Datos
1) Jerárquicas ---> Modelo de árbol invertido. Los nodos padre tienen nodos hijos con info.
2) De red ---> Los nodos hijos pueden tener varios padres.
3) Transaccional ---> Reciben y envían datos a gran velocidad. Y envían información completa, no se envía x trozos. Para grandes volúmenes de datos. Actualmnte no hay muchos (poco comuns).
4) Relacional ---> Modelo actual y + USADO. Tablas de información específica que establecen correspodencia con otras tablas.
5) Documental ---> Almacenan objetos tipo texto entero. Las que + se usan junto con relacional en desarrollo web. Almacenan gran volumen de datos. (Datos históricos: historial clínico)
6) Orientadas a objetos ---> Nuevo/recientes. Propio de sistemas informátc orientados a objetos.
7) Deductivas ---> Aplicar reglas/ hechos para obtener INFORMACIÓN COMPLEJA.

SQL (Structured query language) es un lenguaje estandarizado utilizado para comunicarse con una base de datos RELACIONAL y realizar operaciones de mantenimiento y consultas sobre los datos.

SQLite es una base de datos minimalista. Permite desplegar bases de datos (pequeñas y medianas) de forma rapida y sencilla. Se usa un archivo simple para almacenar datos. Conectar con python (Intercambio de dato ágil).

Persistencia e intercambio de datos
La persistencia de datos implica la conversión de datos entre la memoria y el formato de almacenamiento, y trabajar con los datos convertidos almacenados. Si tenemos una app o una plataforma q usa una base de datos almacenada (o ficheros ---> mal para la memoria).---> Existe un intercambio fluido y persistencia de la información.
La información: Disponible/ Veraz/ Íntegra

PERSISTENCIA: JSON , DBM , SHELVE
La biblioteca estándar de Python ofrece una variedad de módulos [dbm (interfaz con claves), shelve(similar a un diccionario)] que manejan ambos aspectos en diferentes situaciones. Para la persistencia de datos, es común utilizar el módulo pickle para serializar objetos, mientras que json se utiliza con más frecuencia para aplicaciones web ---> para proyectos en producción (no dataframe).

XML: Formato fichero q guarda info a modo de etiqueta , usarlo para poca información. < >, <>

SQLite3 es un módulo de Python q permite conectar una base de datos al código. Implementar una base de datos relacional (integrada) es rápida, rigurosamente probada y flexible. Para crear prototipos y despliegue de producción para algunas aplicaciones.

· BASES DE DATOS RELACIONALES

INTRODUCCIÓN
Una base de datos relacional es un tipo de base de datos organizada en tablas de tal manera que no necesita guardar toda la informacion en una unica tabla (ideas divide info).
El contenido son los propios datos.

4 gestores de bases de datos relacionales (SQL): SQLITE, MySQL, PostgreSQL, Oracle

MySQL: componentes esenciales del álgebra relacional con buenos rendimientos.

SQLite no funciona según un modelo cliente-servido. Es útil cuando se desea distribuir una aplicación que deba manipular datos sobrepuestos que no disponen, necesariamente, de un servidor de bases de datos relacionales.

ORM

Es una herramienta que permite manipular datos de una base de datos relacional a través de objetos, en lugar de manipular directamente tablas de datos ----> Es un mapeo de entidades relacionales y de objeto. Su objetivo es simplificar el proceso y se puede aprovechar la sintaxis POO. Se guardan los atributos. MAPEA RELACIONES CON OBJETOS. Función principal es aislar al usuario de las diferencias entre las distintas bases de datos.

Existen diversas soluciones de ORM para Python: QLAlchemy, Django, SQLObject ,Storm.

·OTRAS BASES DE DATOS

No relacionales (No son como una tabla): Objeto, grafo, documento

Proyectos con docker [(contenedor), guarda info virtualizada(imágenes) q no requiere la instancia completa]:
---> bases de datos no relacionales (NoSQL)
Se suelen usar bases de:

- dato objeto: ZODB ---> varios servidores y clientes simultaneamemente
- dato grafo: Neo4J ---> para almacenar estructuras complejas de datos
- dato documento: Couch DB y Mongo DB

NoSQL es un movimiento que defiende que las bases de datos relacionales no son la única forma de almacenar datos. Incluye varias tecnologías diferentes que tienen objetivos y formas de almacenar datos distintas.

Redis es una base de datos clave-valor.

MongoDB permite manipular objetos estructurados en forma binaria mediante BSON. MongoDB permite la replicación maestro-esclavo.

Cassandra es una base de datos NoSQL. Diseñada para un volumen importante de datos distribuidos en varios servidores y se utiliza para obtener tiempos de respuesta mínimos y una alta tolerancia a fallos. Cassandra es una base de datos orientada a columnas y presenta funcionalidades clave-valor.

·LDAP

Lightweight Directory Access Protocol

Es decir, PROTOCOLO DE ACCESO a la info contenida en un directorio jerarquico. No se accede a la información aleatoriamente sino siguiendo una pauta (podría ser mala).

Los servidores LDAP pueden trabajar en modo síncrono o asíncrono. El modo síncrono se utiliza cuando se espera una respuesta rápida y el modo asíncrono se utiliza cuando se pueden obtener muchos resultados o cuando no se quiere tener una aplicación aparentemente parada esperando una respuesta del servidor.

· BIG DATA

- ¿Que inició la revolución de los datos masivos?

Al principio tenemos el proyecto del genoma humano y despues el IOT: Intenet Of Things
Tenemos :

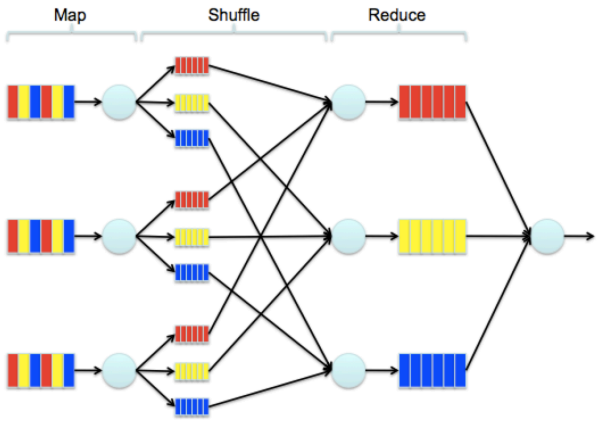
EDGE COMPUTING: mayor capacidad computacional y de procesamiento alejados de la nube ---> en servidores alejados d dnd ocurre.
CLOUD COMPUTING: La computación en nube---> subconjunto de cloud + cercano

Definición: Conjunto de técnicas y tecnologías para el tratamiento y almacenamiento de datos, en entornos de: [VOLUMEN/ VARIEDAD/ VELOCIDAD] / VERACIDAD(añadida x IBM)/ VALOR.

Herramientas de Big Data:
BigData: *Hadooop, MapReduce*, Spark -----> Manipulación de información masiva.

- MapReduce:

MAP	SHUFFLE	REDUCE
Mapea	Recolectar x orden(valor)	Encontrar subgrupo menor
Convierte el fichero en clave-valor		Combinacion de valores segun el problema
Encontrar agrupaciones		



- ***APACHE HADOOP:***
El ecosistema Hadoop es una herramienta de Big Data para manejar datos poco estructurados en un clúster. Toda operación en Hadoop se realiza mediante MapReduce.

HDFS ---> Distribución de info en diferentes nodos, tratamiento de info en //, toleracia a fallos puesto que existen múltiples copias de los datos en diferentes nodos.

Alternativas al modelo MapReduce:
- Apache Storm (computación distribuida en tiempo real) o
- Graph (sistema de procesamiento distribuido en grafos).

- SPARK: (apache spark)

CARACTERÍSTICAS

- procesamiento streaming
- soluciona desventajas de map- reduce
- para proogramas iterativos mucho mejor

MODULOS

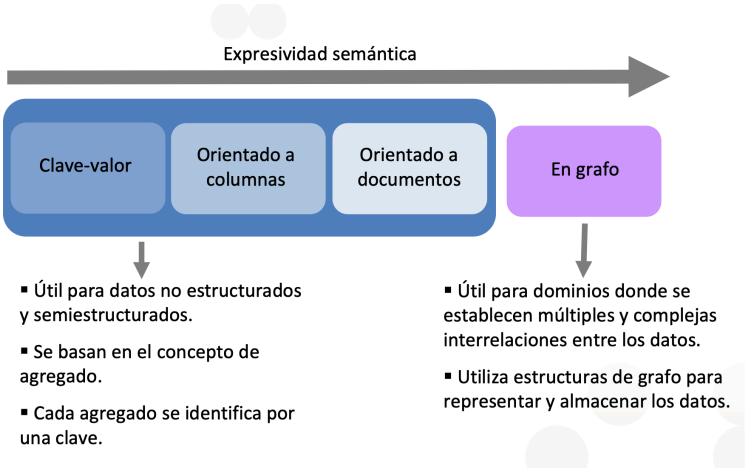
- Spare core engine: Motor de ejecución
- Spark SQL: Da soporte a consultas iterativas
- Spark Streaming: Procesar y analizar datos en tiempo real

- Mlib: Aprendizaje automático----> extracción de patrones (outlier, frecuentes) y relaciones
- GraphX : motor para el análisis de grafos
- SparkR : permite integrar el lenguaje estadístico R

- INTRODUCCIÓN NO SQL

Diferencias entre NoSQL y SGBDR
- No existes procesos para asegurar integridad de datos.
- Datos no analizados antes de almacenarse.
- Almacenan datos iindexadoos por una clave
- Usado en entornos con flexibilidad
- Existen funcionalidades no implementadas

Caracteristicas



Esquema de datos flexible ----> schemaless

- Teorema Cap:

- Consistencia (Consistency): usuarios del sistema recuperan los mismos valores para mismos datos en un mismo instante de tiempo.
- Disponibilidad (Availability): las peticiones de servicio enviadas por los usuarios a un nodo que está disponible deben obtener su debida respuesta.
- Tolerancia a particiones (tolerance to network Partitions): en caso de fallo el sistema no puede caer

----> El teorema CAP enuncia que es imposible garantizar simultáneamente las tres características. Sacrificar 1 depende d lo q sacrifiques tendrás una BD u otra.

ACID	**(no se usan a la vez)**	**BASE**
atomicidad		disponibilidad limitada (Basic availability)
consistencia		estado flexible (Soft-state)
aislamiento		consistencia final en el tiempo (eventual)
definitividad		

MODELOS DE AGREGACIÓN -NO SQL

- Funcionalidades claramente fijados y sin variantes
- Pocos solapamientos en necesidades de datos
- No existen interrelaciones complejas
- Datos sujetos a pocos cambios

Ej: Carro de la compra

1) Modelo clave - valor
2) Modelo documentos: doc en formato json o xml
3) Modelo orientado a columnas: Filas representan agregaciones, columnas ambitos y pueden agruparse en familias de columnas.

MODELOS DE DATOS -NO SQL

Conceptualizamos a traves de: modelos identidad relacion, digramas uml, modelos relacionales y prerelacionales y modelos nooo sql.
Modelos de datos:
- Estructuras de datos: para construirlas
- Operaciones de manipulacion y consultas
- Mecanismos para definir restricciones de integridad

MODELOS DE GRAFOS -NO SQL

- nodos unidos x aristas (direccion)
- no existe modelo estandar pero mejora tiempo en navegacion entre nodos
- schemaless
- suelen emplear lenguajes de consulta de otro nivel
- existenn restricciones
- nodos solo se eliminan si no tienen conexion
- noo se define arista sin origen y destino