Preguntas de Comprensión y Análisis sobre la Introducción a las Bases de Datos
La siguiente lista de preguntas está diseñada para evaluar la comprensión y el análisis de los conceptos clave relacionados con la introducción a las bases de datos, tal como se desarrollan en el contenido de la unidad formativa. A través de estas preguntas, se pretende explorar tanto el conocimiento teórico como la capacidad de aplicar dicho conocimiento en situaciones prácticas, abarcando desde los fundamentos de los sistemas de bases de datos hasta su arquitectura, administración y uso en organizaciones modernas.
Preguntas:

1. ¿Qué es una base de datos?
Es una colección organizada de datos que están estructurados de manera que faciliten su almacenamiento, recuperación y manipulación eficientes.

2. Menciona una ventaja de utilizar un sistema de bases de datos.
Reducción de la Redundancia de Datos, Consistencia de Datos, Acceso Rápido y Eficiente a los Datos, Seguridad de los Datos, Control de la Integridad de los Datos, Mantenimiento Centralizado, Soporte para Acceso Concurrente y Recuperación y Restauración de Datos.

3. ¿Cuál es la función principal de un Sistema de Gestión de Bases de Datos (DBMS)?
Administrar, almacenar, organizar, y recuperar los datos de manera eficiente y segura.

4. ¿Qué diferencia hay entre un sistema de almacenamiento tradicional y un sistema de bases de datos?
La posibilidad de gestionar de una manera rápida, flexible y concurrente, grandes volúmenes de datos.

5. ¿Qué tipos de datos se pueden almacenar en una base de datos?
Bases de Datos Relacionales, Bases de Datos NoSQL, Bases de Datos Orientadas a Objetos, Bases de Datos Distribuidas.

6. ¿Qué es la redundancia de datos y por qué es un problema?
Es uno de los problemas del los sistemas tradicionales de gestión de datos, la duplicación innecesaria de la misma información en diferentes lugares o archivos. La redundancia no solo ocupa espacio de almacenamiento adicional, sino que también puede conducir a la inconsistencia de datos.

7. ¿Qué es la inconsistencia de datos?
La inconsistencia es cuándo la misma información presenta valores diferentes en diferentes lugares.

8. ¿Qué es el acceso concurrente y qué problema puede generar en una base de datos?
El acceso concurrente ocurre cuando múltiples usuarios o aplicaciones intentan acceder y manipular los mismos datos al mismo tiempo. Si no existe un control adecuado, este acceso simultáneo puede dar lugar a anomalías, como la perdida de actualizaciones, lecturas inconsistentes o incluso la corrupción de los datos.

9. ¿Qué se entiende por integridad de los datos en un sistema de bases de datos?
La integridad de los datos se refiere a la precisión y consistencia de la información almacenada en una base de datos.

10. ¿Cuál es el papel del Administrador de Bases de Datos (DBA)?
Es el profesional encargado de gestionar, mantener y garantizar el correcto funcionamiento de una base de datos en una organización. El rol del DBA es crucial para asegurar que la base de datos opere de manera eficiente, segura y sin interrupciones, además de cumplir con los requisitos de los usuarios y las aplicaciones.

11. ¿Qué es un esquema en una base de datos?
Es la estructura lógica de la base de datos y define cómo se organizan los datos dentro de ella. El esquema puede dividirse en 3 niveles, físico, lógico y externo.

12. ¿Cuáles son los tres niveles de la arquitectura de una base de datos?
Interno, Conceptual y Externo, esta composición es conocida como arquitectura de tres niveles o modelo de tres esquemas.

13. ¿Qué función tiene el nivel interno en la arquitectura de una base de datos?
Se refiere a como se almacenan físicamente los datos dentro de la base de datos, en el hardware del sistema.

14. ¿Qué describe el nivel conceptual en la arquitectura de bases de datos?
Es la representación abstracta de toda la base de datos, proporciona una visión lógica y unificada de la información almacenada independientemente de como está organizada físicamente.

15. ¿Qué se muestra en el nivel externo de la arquitectura de una base de datos?
Es el nivel más cercano a los usuarios finales y las aplicaciones. También conocido como vista usuario, este nivel define como los usuarios individuales ven los datos.

16. Define el término "índice" en una base de datos.
Los índices son estructuras auxiliares utilizadas para mejorar la velocidad de las consultas en la base de datos.

17. ¿Para qué sirven las vistas en una base de datos?
Las vistas son consultas almacenadas que presentan una representación lógica o virtual de los datos de una o más tablas.

18. ¿Qué es una transacción en un sistema de bases de datos?
Una transacción es una secuencia de operaciones que se ejecutan como una unidad lógica de trabajo. Las transacciones aseguran que las operaciones sobre la base de datos se realicen de manera coherente y segura.

19. ¿Qué asegura el modelo ACID en una base de datos?
El ACID son las propiedades que se deben cumplir en las operaciones de transacción, Atomicidad, Consistencia, Aislamiento y Durabilidad.

20. Menciona un ejemplo de comando del Lenguaje de Definición de Datos (DDL).
Son un conjunto de comandos utilizados para definir, modificar y eliminar la estructura de los objetos en una base de datos. Los objetos más comunes que se gestiona con DDL son tablas, índices y esquemas. Uno de los comandos ejemplos sería el comando CREATE, y se utiliza para crear una nueva tabla.

21. ¿Cuál es la diferencia entre DDL y DML?
Data Dfinition Lenguaje se enfoca en la estructura de la base de datos, mientras que el Data *Manipulation Lenguage* se enfoca en los datos mismos. DDL crea, modifica o elimina estructuras de datos, mientras que DML inserta, actualiza o elimina registros. La DDL afectan a la estructura física de la base de datos y la DML afectan a los datos almacenados.

22. ¿Qué es el control de concurrencia en un sistema de bases de datos?
Es una función esencial en cualquier sistema de bases de datos que maneja múltiples usuarios o aplicaciones que acceden a los datos de manera simultánea.

23. ¿Qué técnicas se utilizan en un DBMS para garantizar la seguridad de los datos?
Esto se logra a través de implementación de controles de accesos no autorizados, autenticación de usuarios y encriptación de datos, por medio de roles y permisos que determinan que usuarios pueden acceder a ciertos datos y que operaciones pueden realizar (lectura, escritura, modificación, eliminación).

24. ¿Qué es una consulta en una base de datos?
El procesador de consultas es un componente del DBMS que interpreta y ejecuta las consultas de los usuarios. Cuando un usuario o una aplicación envía una consulta SQL, el procesador de consultas analiza, optimiza y genera el plan de ejecución más eficiente para recuperar los datos solicitados.

25. ¿Qué hace un analista de datos con la información en una base de datos?
Los analistas de datos son usuarios que trabajan con grandes volúmenes de datos para extraer información útil y generar conocimientos que apoyen la toma de decisiones en la organización.

26. ¿Cuál es la importancia de las copias de seguridad en la gestión de bases de datos?
La recuperación ante Fallos.

27. Explica brevemente la arquitectura cliente-servidor en bases de datos.
Es un enfoque común de arquitectura en los entornos empresariales. En este modelo, la base de datos reside en un servidor central(servidor de base de datos), mientras que los usuarios interactúan con la base de datos a través de aplicaciones cliente que se ejecutan en sus computadores.

28. ¿Qué ventajas ofrece una arquitectura de bases de datos en la nube?
Con el auge de la computación en la nube, michas organizaciones están adoptando arquitecturas de bases de daros en la nube, donde las bases de datos se alojan en plataformas en la nube en lugar de en servidores locales. Ofrecen mayor Elasticidad, Alta Disponibilidad, Modelo de pago por uso.

29. ¿Qué es la fragmentación en una base de datos distribuida?
Los datos pueden estar fragmentados, es decir, divididos en partes que se almacenan en diferentes servidores.

30. ¿Por qué es importante la escalabilidad en la planificación de la capacidad de una base de datos?
La escalabilidad de la base de datos sirve para asegurarse que el sistema pueda manejar el crecimiento futuro den términos de volumen de datos y número de usuarios.

Preguntas de Ampliación:
1. Explica cómo el modelo de tres niveles en la arquitectura de bases de datos garantiza la independencia de datos física y lógica, y analiza las consecuencias de no mantener esta independencia en un sistema de bases de datos.
El modelo de tres niveles en la arquitectura de bases de datos, también conocido como la arquitectura ANSI/SPARC, garantiza la independencia de datos física y lógica, al separar la estructura de la base de datos en los tres niveles distintos, externo, conceptual e interno. Esta separación permite que los cambios en un nivel no afecten a los demás, lo que facilita la adaptabilidad y el mantenimiento del sistema. Si esto no se implementa, podrían aparecer los siguientes problemas, complejidad de mantenimiento, ya que cualquier cambio en la estructura física o lógica afectará a todos los demás niveles y requeriría de su ajuste también. Costos elevados, estos cambios podrían generar que las aplicaciones tuviesen que ser reescritas para realizar ajustes en las vistas usuarios, y así también repercutirá en el impacto de rendimiento y en una flexibilidad reducida,

2. ¿Cómo implementa un DBMS la gestión de transacciones distribuidas en una base de datos distribuida geográficamente, y qué desafíos se presentan en términos de consistencia de datos y latencia?
Esta implementación sería un desafío complejo que implica coordinar operaciones entre múltiples nodos o sistemas, asegurando que todas las partes de una transacción se completen juntas o fallen juntas.
Un DBMS implementa transacciones distribuidas mediante varios mecanismos:

1. Protocolos de dos fases (2PC) este enfoque común garantiza la consistencia en transacciones distribuidas. Este proceso implica dos fases, fase de preparación en donde cada recurso involucrado en la transacción prepara los cambio y confirma su capacidad para realizarlos. Y fase de compromiso, si todos los recursos confirman su preparación, la transacción se compromete; de lo contrario, se revierte.
2. Algoritmo de Consenso: algoritmos como RAFT o Paxos ayudan a lograr un consenso entre los nodos sobre el estado de la transacción, asegurando que todos los nodos estén de acuerdo antes de comprometer una transacción.
3. Gestión de Redo Logs: los logs de Redo se utilizan para mantener un registro de los cambios realizados durante una transacción. En sistemas distribuidos, estos logs deben ser replicados entre nodos para garantizar la recuperación en caso de fallas.

Desafíos.
La consistencia de datos es crucial en un sistema distribuido. Los desafíos incluyen:
Modelos de consistencia: se deben elegir modelos de consistencia adecuados para el tipo de aplicación. La consistencia fuerte es ideal para transacciones financieras, mientras

que la consistencia eventual es más adecuada para aplicaciones que permiten cierto retraso en la actualización de datos,

Conflictos de Actualización Concurrente: cuando múltiples nodos intentan actualizar el mismo dato simultáneamente, se puede producir conflictos. Mecanismos como el control de concurrencia optimista ayudan a detectar y resolver estos conflictos.

Latencia.

También comporta un desafío significativo en relación con la comunicación entre Nodos y esto puede afectar el rendimiento de las transacciones, especialmente en sistemas que requieren confirmación sincrónica

Estrategias para mitigar la latencia, técnicas como la replicación de datos, el uso de caché, y la optimización de rutas de red pueden ayudar a reducir la latencia. Sistemas como GaussDB.Global y CockroachDB implementan transacciones distribuidas utilizando enfoques descentralizados y algoritmos de consenso para manejar la latencia y garantizar la consistencia.

En resumen, la gestión de transacciones distribuidas en bases de datos geográficamente distribuidas implica desafíos significativos en términos de consistencia y latencia. Los sistemas modernos utilizan algoritmos de consenso y modelos de consistencia adecuados para equilibrar estos aspectos y asegurar un rendimiento óptimo.

3. Analiza el impacto de los mecanismos de replicación de datos en bases de datos distribuidas sobre la consistencia eventual. ¿Cómo se equilibran la consistencia, disponibilidad y partición en el teorema CAP?

Los mecanismos de replicación de datos en bases de datos distribuidas tienen un impacto significativo sobre la consistencia eventual, ya que permiten que los datos se repliquen en múltiples nodos para mejorar la disponibilidad y tolerancia a fallos, pero pueden introducir retrasos en la actualización de todos los nodos.

Impacto de la Replicación en la Consistencia Eventual: la replicación de datos es crucial en sistemas distribuidos para asegurar que los datos estén disponibles en diferentes nodos. Sin embargo, esto puede afectar la consistencia eventual de varias maneras:

1. Retrasos en la actualización: Cuando se realiza una actualización en un nodo, puede tardar mucho tiempo en propagarse a todos los demás nodos. Esto significa que, durante un periodo, los nodos pueden tener versiones diferentes de los datos, lo que afecta la consistencia eventual.
2. Conflictos de Actualización: si dos o más nodos intentan actualizar el mismo dato simultáneamente, pueden surgir conflictos. Los mecanismos de resolución de conflictos son necesarios para garantizar que los datos converjan a un estado consistente eventualmente.

Equilibrio en el teorema CAP

El teorema CAP establece que un sistema distribuido no puede garantizar simultáneamente más de dos de las siguientes propiedades: Consistencia (C), Disponibilidad (A) y Tolerancia a Particiones (P).

1. Consistencia, garantiza que todos los nodos tengan los mismos datos, esto requiere bloquear actualizaciones hasta que todos los nodos estén sincronizados, lo que afecta la disponibilidad,
2. Disponibilidad, asegura que el sistema responda a toda las solicitudes de lectura y escritura. Para lograr alta disponibilidad, los sistemas pueden sacrificar la consistencia fuerte y optar por la consistencia eventual, permitiendo que los nodos respondan incluso si no tienen los datos más recientes.

3. Tolerancia a particiones, permite que el sistema siga funcionando a pesar de fallos en la comunicación entre nodos. Para mantener esta tolerancia, los sistemas pueden sacrificar la consistencia fuerte y permitir que los nodos operen de forma independiente hasta que se resuelva la partición.

Estrategias para Equilibrar CAP

CP Consistencia y Tolerancia a Particiones, sacrificas la disponibilidad. Los sistemas que eligen CP garantizan que los datos sean consistentes en todos los nodos, pero pueden no estar disponibles durante una partición.

AP Disponibilidad y Tolerancia a Particiones, sacrificas la consistencia fuerte. Estos sistemas aseguran que siempre haya respuesta, pero pueden devolver datos inconsistentes durante una partición.

CA Consistencia y Disponibilidad, sacrifica la tolerancia a particiones. Los sistemas CA garantizan consistencia y disponibilidad, pero no pueden tolerar fallos en la comunicación entre nodos.

4. Explica cómo la técnica de particionamiento horizontal y vertical en bases de datos distribuidas afecta el rendimiento de las consultas y la escalabilidad del sistema. Proporciona ejemplos específicos donde cada tipo de particionamiento sería beneficioso.

La técnica de particionamiento en bases de datos distribuidas es crucial para mejorar el rendimiento de las consultas y la escalabilidad del sistema. Existen dos tipos principales de particionamiento: horizontal y vertical.

Particionamiento Horizontal

El particionamiento horizontal, también conocido como *sharding*, implica dividir una tabla en múltiples particiones basadas en filas. Cada partición contiene el mismo conjunto de columnas pero menos fila.

Beneficios y Ejemplos:

Mejora del Rendimiento de Consultas: Al distribuir las filas en múltiples nodos, las consultas pueden ejecutarse en paralelo, reduciendo el tiempo de respuesta. Es ideal para aplicaciones con grandes volúmenes de datos y consultas que requieren acceso rápido a filas específicas.

Ejemplo: en un sistema de gestión de inventario, si los productos se dividen por regiones geográficas, cada región podría tener su propia partición. Esto facilita las consultas locales y reduce la carga en cada nodo.

Escalabilidad: Facilita la escalabilidad horizontal al permitir agregar más nodos para manejar el crecimiento del conjunto de datos sin afectar el rendimiento.

Particionamiento Vertical.

El particionamiento vertical implica dividir una tabla en múltiples particiones basadas en columnas. Cada partición contiene un subconjunto de columnas de la tabla original.

Beneficios y Ejemplos:

Optimización del Acceso a Datos: Al almacenar columnas frecuentemente accedidas en una partición separada, se reduce el tiempo de acceso a los datos necesarios para una consulta. Esto mejora el rendimiento al minimizar la cantidad de datos que se deben leer.

Ejemplo: En una base de datos de productos, las columnas de nombre, descripción y precio podrían estar en una partición, mientras que las columnas de inventario y fecha de última orden podrían estar en otros. Esto es útil cuando diferentes columnas tienen patrones de acceso distintos.

Reducción del Costo de Almacenamiento: Almacenar columnas menos frecuentemente accedidas en un almacenamiento más económico pueden reducir los costos totales.
Comparación y Selección

Característica	Particionamiento Horizontal	Particionamiento Vertical
Enfoque	Divide filas en particiones	Divide columnas en particiones
Rendimiento	Mejora consultas que requieren filas específicas	Optimiza consultas que necesitan columnas específicas
Escalabilidad	Escalabilidad horizontal fácil	Reduce el tamaño de cada partición, mejorando la eficiencia
Uso Ideal	Aplicaciones con grandes volúmenes de datos y consultas rápidas	Aplicaciones con columnas de acceso frecuente y diverso

Ejemplo Específicos
Particionamiento Horizontal: Un sistema de gestión de clientes podrías dividir sis datos por regiones geográficas, mejorando el acceso local y la escalabilidad.
Particionamiento Vertical: Un sitio web de comercio electrónico podría dividir sus datos de productos en columnas de descripción del producto y columnas de inventario, optimizando el acceso a datos frecuentemente consultados.
En resumen, el particionamiento horizontal es ideal para aplicaciones que requieren acceso rápido a filas específicas y necesitan escalabilidad horizontal, mientras que el particonamiento vertical es beneficioso para optimizar el acceso a columnas específicas y reducir el costo de almacenamiento.

5. Discute las ventajas y desventajas del uso de bases de datos NoSQL frente a bases de datos relacionales en aplicaciones que requieren alta escalabilidad y disponibilidad. ¿Qué compromisos se deben hacer en términos de consistencia y transacciones?
Las bases de datos NoSQL y relacionales tienes ventajas y desventajas diferentes, especialmente en aplicaciones que requieren alta escalabilidad y disponibilidad. A continuación, se discuten los puntos de vista diferente de cada tipo de base de datos y los compromisos que se deben hacer en términos de consistencia y transacciones.
Ventajas de las bases de Datos NoSQL.

1. Escalabilidad Horizontal: Las bases de datos NoSQL están diseñadas para escalar horizontalmente, lo que significa que se pueden agregar más nodos para manejar el

crecimiento del tráfico sin necesidad de hardware más potente. Esto las hace ideales para aplicaciones con grandes volúmenes de dato y tráficos variable.

2. Flexibilidad en el Esquema: NoSQL permite esquemas dinámicos, lo que facilita la adaptación a cambios en la estructura de los datos sin necesidad de alterar la base de datos.
3. Manejo de Datos No Estructurados; Son especialmente útiles para manejar datos estructurados o semiestructurados como JSON o XML, lo que las hace adecuadas para aplicaciones que requieren flexibilidad en el formato de los datos.
4. Mejora en la Disponibilidad: Ofrecen alta disponibilidad gracia a su capacidad para tolerar particiones y mantenerse operativas, incluso si algunos nodos fallan.

Desventajas de las Based de Datos NoSQL

1. Consistencia Eventual: Muchas bases de datosNoSQL sacrifican la consistencia fuerte por disponibilidad y tolerancias a particiones, lo que puede resultar en datos inconsistentes temporalmente.
2. Limitaciones en Consultas Complejas: NoSQL no están eficiente para realizar consultas complejas que requieren *joins* o subconsultas, lo que puede ser un desafío en ciertas aplicaciones.
3. Complejidad en el Mantenimiento: La arquitectura distribuida puede requerir un mantenimiento más complejo, incluyendo la configuración de *sharding* y el equilibrio de carga.

Ventajas de las bases de Datos Relacionales

1. Consistencia Fuerte: garantizan la consistencia de los datos en todo momento, lo que es crucial para aplicaciones que requieren transacciones precisas y seguras.
2. Soporte para Consultas Complejas: Son ideales para realizar consultas complejas que involucran *joins* y subconsultas, lo que facilita el análisis de datos.
3. Seguridad y Madurez: Ofrecen características de seguridad más robustas y una documentación más completa debido a su larga historia de uso.

Desventajas de las Bases de Datos Relacionales

1. Estabilidad Limitada: Suelen escalar verticalmente, lo que significa que requieren hardware más potente para manejar el crecimiento, lo que puede ser costoso.
2. Esquema Rígido: Requieren un esquema predefinid, lo que puede ser inflexible para aplicaciones que necesitan adaptarse rápidamente a cambios en la estructura de datos.
3. Costos: Pueden ser más costosas debido a los costos de licencia y *hardware.*

Compromisos en Consistencia y Transacciones

1. Consistencia Eventual vs Consistencia Fuerte: Las bases de datos NoSQL suelen optar por la consistencia eventual, lo que significa que los datos pueden no estar actualizados en todos los nodos al mismo tiempo. Esto se hace a favor de la disponibilidad y la tolerancia a particiones. En contraste, las bases de datos relacionales garantizan la consistencia fuerte, lo que es esencial para aplicaciones que requieren transacciones precisas.
2. Transacciones ACID vs. BASE: las bases de datos relacionales suelen seguir el modelo ACID, que garantiza la atomicidad, consistencia, aislamiento y durabilidad de las transacciones. Las bases de Datos NoSQAL a menudo usan el modelo BASE, que prioriza la disponibilidad básica, el estado suave y la consistencia eventual. Esto significa que las transacciones en NoSQL pueden no ser tan robustas como en las bases de datos relaciones.

En resumen, las bases de datos NoSQL son ideales para aplicaciones que requieren alta escalabilidad, disponibilidad y flexibilidad en el manejo de datos no estructurados, pero sacrificar la consistencia fuerte. Las bases de datos relacionales, por otro lado, garantizan la consistencia y soportan consultas complejas, pero pueden ser menos escalables y más costosas. El compromiso entre consistencia y disponibilidad depende de las necesidades específicas de la aplicación.