|  |
| --- |
| Bases de Datos Relacionales (RDBMS) |

|  |
| --- |
|  |

,

|  |  |
| --- | --- |
| Conceptos a buscar: | |
| * SQL * NoSQL * Data Warehouse. * La Primera Forma Normal (1NF) * La Segunda Forma Normal (2NF) * La Tercera Forma Normal (3NF) * BCNF (Boyce-Codd Normal Form). * Normalización/Desnormalización * OLAP * OLTP | * ETL (Extract, Transform, Load) * Data Mart * teorema CAP * ACID * BASE (Basically Available, Soft state, Eventual consistency). * Clustered * índice bitmap * B-Tree * índice hash.cardinalidad |

,

|  |
| --- |
| Pregunta 1: ¿Cuál de las siguientes afirmaciones sobre normalización en bases de datos es correcta? |
| 1. La Tercera Forma Normal (3NF) elimina todas las dependencias parciales y transitivas en una tabla. 2. La Primera Forma Normal (1NF) requiere que todos los atributos de una tabla sean atómicos y no repetitivos. 3. La Segunda Forma Normal (2NF) elimina todas las dependencias transitivas entre atributos no clave. 4. Una base de datos en 3NF siempre cumple con los requisitos de BCNF (Boyce-Codd Normal Form). 5. La desnormalización aumenta la redundancia en una base de datos para mejorar la eficiencia de consultas. 6. Las bases de datos altamente normalizadas son ideales para sistemas de análisis OLAP. 7. La normalización es un proceso utilizado exclusivamente en bases de datos relacionales y no aplica a NoSQL. 8. El objetivo principal de la normalización es reducir la consistencia de los datos. 9. Ninguna de las anteriores. |
| Respuestas y explicaciones:   1. Verdadera: La 3NF elimina dependencias parciales y transitivas para mejorar la integridad de los datos. 2. Verdadera: La 1NF asegura que los datos estén atomizados y sin listas o conjuntos en una sola columna. 3. Falsa: La 2NF elimina dependencias parciales, no transitivas. 4. Falsa: No todas las tablas en 3NF cumplen con BCNF, ya que BCNF es una forma más estricta. 5. Verdadera: La desnormalización se utiliza para mejorar el rendimiento en sistemas donde la lectura es prioritaria. 6. Falsa: Las bases de datos OLAP se benefician más de la desnormalización por su enfoque en consultas rápidas. 7. Falsa: Algunos sistemas NoSQL también pueden beneficiarse de la normalización, aunque no sea tan común. 8. Falsa: El objetivo principal de la normalización es aumentar la consistencia y reducir la redundancia. |

﻿.

,

|  |
| --- |
| Pregunta 2: En el contexto de análisis de datos, ¿cuál de las siguientes afirmaciones es correcta respecto a las bases de datos transaccionales (OLTP) frente a las bases de datos analíticas (OLAP)? |
| 1. Los sistemas OLTP están optimizados para la lectura masiva de datos históricos. 2. Los sistemas OLAP son ideales para realizar análisis complejos y multidimensionales. 3. Las bases de datos OLTP generalmente utilizan esquemas en estrella para optimizar consultas analíticas. 4. Las operaciones de ETL (Extract, Transform, Load) son típicas de sistemas OLTP. 5. En un sistema OLAP, los datos suelen estar altamente normalizados para mejorar la velocidad de las transacciones. 6. Las bases de datos OLAP están diseñadas para soportar miles de transacciones por segundo. 7. Los índices en sistemas OLTP están diseñados principalmente para acelerar informes agregados. 8. Los sistemas OLTP y OLAP no pueden coexistir en un mismo entorno debido a sus diferencias en arquitectura. i) Ninguna de las anteriores. |
| Respuestas y explicaciones:   1. Falsa: Los sistemas OLTP están optimizados para transacciones rápidas y no para lectura masiva. 2. Verdadera: Los sistemas OLAP se utilizan para análisis complejos y consultas multidimensionales. 3. Falsa: El esquema en estrella es más típico en sistemas OLAP, no OLTP. 4. Falsa: Las operaciones ETL se usan en sistemas OLAP para consolidar datos para análisis. 5. Falsa: Los sistemas OLAP suelen estar desnormalizados para optimizar la lectura. 6. alsa: Las bases de datos OLTP son las que están diseñadas para manejar altas tasas de transacciones. 7. Falsa: Los índices en OLTP optimizan la velocidad de las transacciones, no informes agregados. 8. Falsa: Aunque tienen propósitos diferentes, OLTP y OLAP pueden coexistir mediante soluciones híbridas. |

**﻿**,

|  |
| --- |
| Pregunta 3: ¿Cuáles de las siguientes operaciones son típicas del proceso **ETL** en el contexto de análisis de datos? |
| 1. Actualizar registros en la base de datos en tiempo real. 2. Extraer datos de múltiples fuentes heterogéneas. 3. Transformar datos para cumplir con los requisitos del análisis. 4. Cargar datos transformados en un almacén de datos (Data Warehouse). 5. Realizar análisis predictivo en tiempo real con modelos de machine learning. 6. Normalizar datos para reducir la redundancia en el Data Warehouse. 7. Realizar consultas en una base de datos relacional para informes rutinarios. 8. Procesar datos directamente desde fuentes de datos NoSQL a un Data Mart sin transformación. 9. Ninguna de las anteriores. |
| Respuestas y explicaciones:   1. Falsa: Actualizar registros en tiempo real no es parte del proceso ETL tradicional. 2. Verdadera: El paso de extracción implica obtener datos de diversas fuentes. 3. Verdadera: La transformación de datos es esencial para un análisis correcto. 4. Verdadera: El cargado finaliza el proceso ETL al almacenar los datos en un Data Warehouse. 5. Falsa: El análisis predictivo no forma parte del proceso ETL, sino que utiliza los datos ya transformados. 6. Falsa: La desnormalización es más común en almacenes de datos para mejorar la velocidad de consulta. 7. Falsa: Las consultas en bases de datos no son operaciones típicas de ETL. 8. Falsa: El ETL usualmente incluye una etapa de transformación para adecuar los datos. |

**﻿.**,

|  |
| --- |
| Pregunta 4:¿Cuál de las siguientes afirmaciones sobre índices en bases de datos es correcta? |
| 1. Un índice único permite múltiples entradas duplicadas para una clave. 2. Los índices clustered determinan el orden físico de los datos en la tabla. 3. Crear un índice mejora la velocidad de inserciones en la tabla. 4. Los índices full-text se utilizan para mejorar consultas numéricas. 5. Los índices ocupan espacio adicional en disco, lo que puede ralentizar el rendimiento de la base de datos. 6. Los índices non-clustered no afectan el rendimiento de las consultas SELECT. 7. Un índice bitmap es ideal para tablas con una gran cantidad de valores distintos. 8. Crear múltiples índices en una tabla siempre mejora el rendimiento de todas las consultas. 9. Ninguna de las anteriores. |
| Respuestas y explicaciones:   1. Falsa: Un índice único no permite duplicados. 2. Verdadera: Los índices clustered afectan el orden físico de los datos. 3. Falsa: Los índices pueden ralentizar las inserciones debido al tiempo de actualización. 4. Falsa: Los índices full-text mejoran consultas de texto, no numéricas. 5. Verdadera: Los índices ocupan espacio adicional y pueden afectar el rendimiento. 6. Falsa: Los índices non-clustered pueden acelerar SELECT pero no siempre. 7. Falsa: Los índices bitmap son más eficientes para datos con pocos valores distintos. 8. Falsa: Demasiados índices pueden ralentizar las inserciones y actualizaciones. |

﻿.

|  |
| --- |
| Pregunta 5: En el contexto de bases de datos distribuidas, ¿cuál de las siguientes afirmaciones sobre el teorema CAP (Consistencia, Disponibilidad, Tolerancia a particiones) es correcta? |
| 1. En un sistema distribuido, siempre es posible garantizar Consistencia, Disponibilidad y Tolerancia a particiones simultáneamente. 2. La consistencia eventual es una propiedad típica de los sistemas que priorizan la Disponibilidad y la Tolerancia a particiones. 3. Si un sistema distribuido prioriza la consistencia, inevitablemente renunciará a la disponibilidad en caso de una partición en la red. 4. Un sistema que prioriza la tolerancia a particiones siempre será más rápido que uno que prioriza la consistencia. 5. Los sistemas que cumplen con el teorema CAP suelen usar el modelo ACID para gestionar transacciones. 6. El teorema CAP aplica exclusivamente a bases de datos NoSQL y no tiene relevancia en bases de datos relacionales. 7. En el caso de una partición, los sistemas que priorizan la disponibilidad pueden devolver respuestas desactualizadas. 8. Un sistema con alta disponibilidad y consistencia perfecta es siempre tolerante a fallos de red. 9. Ninguna de las anteriores. |
| Respuestas y explicaciones:   1. Falsa: El teorema CAP establece que solo se pueden garantizar dos de las tres propiedades al mismo tiempo. 2. Verdadera: La consistencia eventual es común en sistemas que priorizan disponibilidad y tolerancia a particiones. 3. Verdadera: Si un sistema prioriza la consistencia, puede sacrificar la disponibilidad durante una partición. 4. Falsa: No hay una relación directa entre la tolerancia a particiones y la velocidad del sistema. 5. Falsa: Los sistemas que priorizan el teorema CAP usualmente no usan el modelo ACID sino el BASE (Basically Available, Soft state, Eventual consistency). 6. Falsa: El teorema CAP aplica a cualquier sistema distribuido, incluyendo bases de datos relacionales. 7. Verdadera: Los sistemas que priorizan la disponibilidad pueden devolver respuestas que no sean totalmente consistentes. 8. Falsa: La consistencia y la disponibilidad no garantizan automáticamente la tolerancia a fallos de red. |

﻿.

|  |
| --- |
| Pregunta 6:¿Cuáles de las siguientes afirmaciones son correctas en relación con el uso de índices en bases de datos relacionales? |
| 1. Un índice B-Tree es más eficiente para búsquedas de rangos que un índice hash. 2. Los índices bitmap son adecuados para tablas con alta cardinalidad en los campos indexados. 3. El uso de índices siempre mejora el rendimiento de las consultas SELECT. 4. Los índices hash son ideales para búsquedas exactas en columnas de baja cardinalidad. 5. Los índices no tienen impacto en las operaciones de actualización y eliminación. 6. El uso excesivo de índices puede degradar el rendimiento de las inserciones y actualizaciones en la tabla. 7. Un índice compuesto puede mejorar el rendimiento solo si las consultas utilizan los primeros campos en el orden del índice. 8. Un índice parcial solo indexa filas que cumplen con una condición específica, mejorando el uso de espacio. 9. Ninguna de las anteriores. |
| Respuestas y explicaciones:   1. Verdadera: Los índices B-Tree son más eficientes para búsquedas por rango. 2. Falsa: Los índices bitmap son más adecuados para columnas con baja cardinalidad. 3. Falsa: En algunos casos, los índices pueden ralentizar las consultas si no están diseñados adecuadamente. 4. Verdadera: Los índices hash son eficientes para búsquedas exactas, especialmente en columnas de baja cardinalidad. 5. Falsa: Los índices impactan negativamente las operaciones de actualización y eliminación debido al mantenimiento adicional. 6. Verdadera: Un alto número de índices puede ralentizar las inserciones y actualizaciones. 7. Verdadera: Un índice compuesto solo se aprovecha si las consultas incluyen los campos en el orden en que fueron definidos. 8. Verdadera: Un índice parcial optimiza el uso de espacio al indexar solo filas que cumplen con un criterio. |

﻿.

|  |
| --- |
| **Pregunta 7:**Cuál de las siguientes afirmaciones sobre **modelos de datos NoSQL** es correcta? |
| 1. Las bases de datos **documentales** organizan los datos en filas y columnas como las bases de datos relacionales 2. Las bases de datos **clave-valor** son eficientes para consultas complejas que involucran múltiples uniones. 3. Las bases de datos **orientadas a grafos** son ideales para modelar redes sociales y sistemas de recomendaciones. 4. Las bases de datos **columnar** almacenan datos en filas para mejorar la velocidad de lectura en consultas OLTP. 5. El uso de bases de datos NoSQL garantiza la consistencia fuerte en todas las operaciones de escritura. 6. Las bases de datos **documentales** no permiten el uso de esquemas flexibles. 7. Las bases de datos **clave-valor** son ideales para aplicaciones que requieren un acceso rápido a valores mediante claves únicas. 8. Las bases de datos NoSQL no soportan operaciones de agregación complejas como las bases de datos relacionales. 9. Ninguna de las anteriores. |
| **Respuestas y explicaciones:**   1. **Falsa**: Las bases de datos documentales no utilizan filas y columnas como las relacionales. 2. **Falsa**: Las bases de datos clave-valor no son adecuadas para consultas con uniones complejas. 3. **Verdadera**: Las bases de datos de grafos están diseñadas para modelar relaciones complejas, como en redes sociales. 4. **Falsa**: Las bases de datos columnar están optimizadas para lectura en sistemas OLAP, no OLTP. 5. **Falsa**: No todas las bases de datos NoSQL garantizan consistencia fuerte; algunas solo garantizan consistencia eventual. 6. **Falsa**: Las bases de datos documentales permiten esquemas flexibles. 7. **Verdadera**: Las bases de datos clave-valor son óptimas para accesos rápidos mediante claves únicas. 8. **Falsa**: Algunas bases de datos NoSQL, como MongoDB, soportan operaciones de agregación. |

﻿.

|  |
| --- |
| **Pregunta 8:**¿Cuál de las siguientes afirmaciones sobre **transacciones ACID** en bases de datos es **correcta?** |
| 1. La propiedad de **Atomicidad** asegura que una transacción parcial pueda ser revertida en cualquier momento. 2. La **Consistencia** garantiza que una transacción lleve la base de datos de un estado válido a otro estado válido. 3. La propiedad de **Aislamiento** permite que múltiples transacciones se ejecuten simultáneamente sin interferir entre sí. 4. La propiedad de **Durabilidad** significa que una transacción confirmada puede ser revertida en caso de un fallo del sistema. 5. El uso de **ACID** es obligatorio en sistemas NoSQL para garantizar la integridad de los datos. 6. La propiedad de **Aislamiento** asegura que una transacción sea visible para otras transacciones tan pronto como comience. 7. La **Consistencia eventual** es una propiedad de las bases de datos que cumplen con ACID. 8. Las transacciones ACID son menos comunes en aplicaciones de análisis de datos debido a su alto costo. 9. Ninguna de las anteriores. |
| **Respuestas y explicaciones:**   1. **Falsa**: La atomicidad asegura que una transacción sea todo o nada, no que sea parcialmente reversible. 2. **Verdadera**: La consistencia asegura que la base de datos esté en un estado válido después de la transacción. 3. **Verdadera**: El aislamiento evita que las transacciones interfieran entre sí. 4. **Falsa**: La durabilidad asegura que una transacción confirmada permanezca incluso tras un fallo. 5. **Falsa**: No todos los sistemas NoSQL garantizan las propiedades ACID. 6. **Falsa**: El aislamiento en realidad evita que las transacciones sean visibles hasta que se confirmen. 7. **Falsa**: La consistencia eventual es una propiedad de sistemas BASE, no ACID. 8. **Verdadera**: Las transacciones ACID son menos comunes en análisis de datos debido a su sobrecarga. |

﻿.

|  |
| --- |
| Pregunta 9:¿Cuál de las siguientes afirmaciones sobre el uso de índices en bases de datos relacionales y su impacto en el rendimiento es correcta? |
| 1. Un índice único garantiza que no haya filas duplicadas en la columna indexada, mejorando la velocidad de inserción. 2. El uso de un índice clustered reorganiza físicamente los datos en el disco según el orden del índice. 3. Los índices no-clustered son siempre más rápidos que los índices clustered para búsquedas de rangos. 4. El mantenimiento de índices puede ralentizar significativamente las operaciones de escritura, especialmente en tablas con un gran volumen de inserciones. 5. El uso de índices parciales mejora el rendimiento de consultas en bases de datos que tienen un número constante de actualizaciones. 6. Un índice compuesto es más eficiente si las consultas utilizan solo el último campo del índice en su cláusula WHERE. 7. La eliminación de un índice en una tabla muy grande no afecta el rendimiento de las consultas hasta que se reinicien las conexiones a la base de datos. 8. Los índices siempre deben crearse en todas las columnas usadas en una cláusula JOIN para optimizar el rendimiento. 9. Ninguna de las anteriores. |
| Respuestas y explicaciones:   1. Falsa: Aunque un índice único asegura la unicidad, en realidad puede ralentizar la velocidad de inserción debido a la verificación adicional requerida. 2. Verdadera: Un índice clustered reorganiza físicamente los datos en el disco según el orden del índice. 3. Falsa: Los índices clustered son generalmente más rápidos para búsquedas de rangos porque los datos están almacenados en el orden del índice. 4. Verdadera: El mantenimiento de índices ralentiza las operaciones de escritura debido al tiempo adicional necesario para actualizar los índices. 5. Falsa: Los índices parciales son útiles para consultas en subconjuntos específicos, pero pueden ser menos eficientes en tablas con actualizaciones constantes. 6. Falsa: Un índice compuesto solo es eficiente si las consultas utilizan los primeros campos en el orden definido del índice. 7. Falsa: La eliminación de un índice puede afectar inmediatamente el rendimiento de las consultas, sin necesidad de reiniciar conexiones. 8. Falsa: No siempre es necesario crear índices en todas las columnas usadas en JOIN, ya que depende del uso específico de la consulta. |

﻿.

|  |
| --- |
| Pregunta 10: En el contexto de sistemas OLAP (Online Analytical Processing), ¿cuáles de las siguientes afirmaciones son correctas? |
| 1. Las bases de datos OLAP están optimizadas para transacciones rápidas en tiempo real. 2. Los cubos OLAP permiten realizar análisis multidimensionales y consultas agregadas complejas de manera eficiente. 3. Las bases de datos OLAP están diseñadas para operaciones CRUD frecuentes en grandes volúmenes de datos. 4. Los sistemas OLAP son ideales para análisis de datos históricos y generación de reportes. 5. Las consultas OLAP suelen requerir índices clustered para mejorar el rendimiento en operaciones de lectura. 6. Las bases de datos OLAP utilizan modelos normalizados para minimizar la redundancia de datos. 7. Las tablas fact en un modelo de datos OLAP contienen métricas numéricas, mientras que las tablas dimension almacenan detalles descriptivos. 8. Los sistemas OLAP suelen tener un mayor rendimiento que los sistemas OLTP para actualizaciones masivas de datos. |
| Respuestas y explicaciones:   1. Falsa: Los sistemas OLAP están optimizados para consultas analíticas, no para transacciones en tiempo real. 2. Verdadera: Los cubos OLAP permiten realizar análisis multidimensionales y consultas agregadas. 3. Falsa: Los sistemas OLAP no están diseñados para operaciones CRUD frecuentes. 4. Verdadera: Los sistemas OLAP son ideales para análisis de datos históricos y reportes. 5. Falsa: Aunque los índices pueden ayudar, no necesariamente tienen que ser clustered en OLAP. 6. Falsa: Los sistemas OLAP a menudo usan modelos desnormalizados para mejorar la velocidad de consulta. 7. Verdadera: Las tablas fact contienen datos numéricos y las dimension información descriptiva. 8. Falsa: Los sistemas OLTP son más adecuados para actualizaciones masivas. |

﻿.

|  |
| --- |
| **Pregunta 11:**  ¿Cuáles de las siguientes afirmaciones sobre el uso de **transacciones en bases de datos NoSQL** son correctas? |
| 1. La mayoría de las bases de datos NoSQL no soportan transacciones ACID, sino transacciones **BASE** (Basically Available, Soft state, Eventual consistency). 2. Las bases de datos NoSQL con soporte para transacciones ACID tienden a tener un menor rendimiento en aplicaciones que requieren alta escalabilidad. 3. Las bases de datos **documentales** no pueden garantizar la consistencia en escrituras distribuidas. 4. Los **locks** utilizados en transacciones en bases de datos NoSQL son más ligeros que los de bases de datos relacionales. 5. La consistencia eventual es una característica que garantiza que los datos serán consistentes en un tiempo finito después de una escritura. 6. Las bases de datos clave-valor no pueden soportar transacciones que abarquen múltiples claves. 7. En sistemas NoSQL, la propiedad de **Durabilidad** suele ser sacrificada para mejorar el rendimiento de las consultas. 8. Las transacciones distribuidas son fáciles de implementar en bases de datos NoSQL gracias a su modelo de datos flexible. |
| **Respuestas y explicaciones:**   1. **Verdadera**: La mayoría de las bases de datos NoSQL utilizan el modelo BASE en lugar de ACID. 2. **Verdadera**: El soporte de transacciones ACID en NoSQL puede afectar la escalabilidad. 3. **Falsa**: Las bases de datos documentales pueden implementar consistencia fuerte mediante configuraciones adicionales. 4. **Falsa**: Los locks en NoSQL pueden ser más ligeros o más pesados, según la implementación. 5. **Verdadera**: La consistencia eventual asegura que los datos sean consistentes eventualmente. 6. **Falsa**: Algunas bases de datos clave-valor, como Redis, soportan transacciones. 7. **Verdadera**: Sacrificar durabilidad es una práctica común para optimizar el rendimiento. 8. **Falsa**: Las transacciones distribuidas en NoSQL son notoriamente difíciles de implementar. |

﻿.

|  |
| --- |
| Pregunta 12:En el análisis de datos con Pandas, ¿cuál de las siguientes afirmaciones sobre la función groupby() es correcta? |
| 1. El método groupby() siempre devuelve un DataFrame. 2. La función groupby() no puede aplicarse a columnas que contengan valores nulos. 3. Utilizar groupby() seguido de agg() permite realizar múltiples cálculos agregados en un solo paso. 4. El uso de groupby() es equivalente a un pivot\_table() en términos de funcionalidad. 5. El parámetro as\_index=False evita que la columna usada para agrupar se convierta en el índice del resultado. 6. Una vez aplicado groupby(), los datos quedan modificados permanentemente en el DataFrame original. 7. La función groupby() es ineficiente para grandes conjuntos de datos y debe evitarse para optimizar el rendimiento. 8. La función groupby() solo acepta columnas numéricas para su agrupación. |
| Respuestas y explicaciones:   1. Falsa: groupby() devuelve un objeto DataFrameGroupBy, no un DataFrame directamente. 2. Falsa: groupby() puede manejar columnas con valores nulos, aunque los ignora en el agrupamiento. 3. Verdadera: groupby().agg() permite múltiples cálculos agregados en un solo paso. 4. Falsa: pivot\_table() y groupby() tienen algunas diferencias en funcionalidad y uso. 5. Verdadera: as\_index=False evita que la columna agrupada se convierta en el índice. 6. Falsa: groupby() no modifica el DataFrame original. 7. Falsa: groupby() puede ser eficiente si se utiliza correctamente. 8. Falsa: groupby() puede agrupar por columnas no numéricas también. |

﻿.

|  |
| --- |
| Pregunta 13: ¿Cuál de las siguientes afirmaciones es correcta respecto al uso de **JOINs**, **índices** y **consultas optimizadas**? |
| 1. a) Una consulta que utilice INNER JOIN entre Clientes y Pedidos devolverá todas las filas de ambas tablas, incluso si no hay coincidencia en ClienteID. 2. Un índice en la columna ClienteID de la tabla Pedidos mejorará el rendimiento de las consultas que utilicen JOIN con la tabla Clientes. 3. Si Pedidos.Total tiene muchos valores NULL, un índice en esta columna mejorará la eficiencia de las consultas de agregación (SUM, AVG). 4. Un LEFT JOIN entre Clientes y Pedidos con la condición ON Clientes.ClienteID = Pedidos.ClienteID siempre devolverá el mismo número de filas que la tabla Pedidos. 5. En una consulta con múltiples JOINs, el optimizador de consultas siempre evaluará los JOINs en el orden en que aparecen en la consulta. 6. Las operaciones de JOIN que utilizan columnas con índices compuestos siempre son más rápidas que aquellas que utilizan índices simples. 7. Al utilizar un FULL OUTER JOIN, si no hay coincidencia en ninguna de las tablas, se devolverán filas con valores NULL en todas las columnas. 8. Para optimizar la consulta:   SELECT Nombre, SUM(Total) FROM Clientes  JOIN Pedidos ON Clientes.ClienteID = Pedidos.ClienteID  GROUP BY Nombre;  deberíamos agregar un índice en Clientes.Nombre. |
| **Respuestas y explicaciones:**   1. **Falsa**: INNER JOIN solo devuelve filas donde hay coincidencia entre ambas tablas. 2. **Verdadera**: Un índice en ClienteID de Pedidos mejorará el rendimiento al realizar un JOIN con Clientes. 3. **Falsa**: Los índices en columnas con muchos NULL no mejoran significativamente las consultas de agregación. 4. **Falsa**: LEFT JOIN devolverá todas las filas de Clientes, no de Pedidos. 5. **Falsa**: El optimizador de consultas decide el orden de los JOINs basado en estimaciones de costo, no en el orden del código. 6. **Falsa**: Un índice compuesto puede ser menos eficiente que uno simple dependiendo del patrón de consultas. 7. **Verdadera**: FULL OUTER JOIN devuelve filas con NULL cuando no hay coincidencias. 8. **Falsa**: El índice en Clientes.Nombre no optimizaría la consulta dado que la agrupación ocurre después del JOIN. |

﻿.

|  |
| --- |
| **Pregunta 13:**  Dado el siguiente esquema de bases de datos relacionales y sus restricciones:   * Tabla Clientes:   + ClienteID (PRIMARY KEY)   + Nombre   + Edad   + Pais * Tabla Pedidos:   + PedidoID (PRIMARY KEY)   + ClienteID (FOREIGN KEY que referencia Clientes.ClienteID)   + FechaPedido   + Total |
| 1. Una consulta que utilice INNER JOIN entre Clientes y Pedidos devolverá todas las filas de ambas tablas, incluso si no hay coincidencia en ClienteID. 2. Un índice en la columna ClienteID de la tabla Pedidos mejorará el rendimiento de las consultas que utilicen JOIN con la tabla Clientes. 3. Si Pedidos.Total tiene muchos valores NULL, un índice en esta columna mejorará la eficiencia de las consultas de agregación (SUM, AVG). 4. Un LEFT JOIN entre Clientes y Pedidos con la condición ON Clientes.ClienteID = Pedidos.ClienteID siempre devolverá el mismo número de filas que la tabla Pedidos. 5. En una consulta con múltiples JOINs, el optimizador de consultas siempre evaluará los JOINs en el orden en que aparecen en la consulta. 6. Las operaciones de JOIN que utilizan columnas con índices compuestos siempre son más rápidas que aquellas que utilizan índices simples. 7. Al utilizar un FULL OUTER JOIN, si no hay coincidencia en ninguna de las tablas, se devolverán filas con valores NULL en todas las columnas. 8. Para optimizar la consulta: SELECT Nombre, SUM(Total) FROM Clientes JOIN Pedidos ON Clientes.ClienteID = Pedidos.ClienteID GROUP BY Nombre; deberíamos agregar un índice en Clientes.Nombre. |
| Respuestas y explicaciones:   1. Falsa: INNER JOIN solo devuelve filas donde hay coincidencia entre ambas tablas. 2. Verdadera: Un índice en ClienteID de Pedidos mejorará el rendimiento al realizar un JOIN con Clientes. 3. Falsa: Los índices en columnas con muchos NULL no mejoran significativamente las consultas de agregación. 4. Falsa: LEFT JOIN devolverá todas las filas de Clientes, no de Pedidos. 5. Falsa: El optimizador de consultas decide el orden de los JOINs basado en estimaciones de costo, no en el orden del código. 6. Falsa: Un índice compuesto puede ser menos eficiente que uno simple dependiendo del patrón de consultas. 7. Verdadera: FULL OUTER JOIN devuelve filas con NULL cuando no hay coincidencias. 8. Falsa: El índice en Clientes.Nombre no optimizaría la consulta dado que la agrupación ocurre después del JOIN. |

﻿.

|  |
| --- |
| Pregunta 14:Teniendo en cuenta la consistencia y durabilidad en bases de datos distribuidas con replicación maestro-esclavo, ¿cuáles de las siguientes afirmaciones son correctas? |
| 1. En un sistema de replicación maestro-esclavo, solo el maestro puede realizar operaciones de escritura. 2. Los esclavos pueden manejar operaciones de escritura si el maestro falla, garantizando así la consistencia de los datos. 3. En un sistema con consistencia eventual, es posible que dos nodos devuelvan diferentes resultados para la misma consulta. 4. La propiedad de durabilidad implica que los datos se guardan permanentemente en el disco antes de que se confirme una transacción. 5. El uso de journaling en bases de datos distribuidas asegura que los datos sean siempre consistentes, incluso si hay fallos durante la replicación. 6. Si la latencia entre los nodos es alta, un sistema que utiliza consistencia estricta puede experimentar un mayor tiempo de respuesta para las consultas de lectura. 7. En un entorno con alta disponibilidad, el failover automático permite que un esclavo se convierta en el nuevo maestro sin pérdida de datos. 8. Las bases de datos con ACID garantizan una mayor escalabilidad que las que usan BASE debido a su modelo transaccional. |
| Respuestas y explicaciones:   1. Verdadera: En un sistema maestro-esclavo tradicional, solo el maestro maneja escrituras. 2. Falsa: En replicación maestro-esclavo, los esclavos solo manejan lecturas. 3. Verdadera: Consistencia eventual permite que diferentes nodos muestren datos distintos temporalmente. 4. Verdadera: La durabilidad garantiza que los datos sean permanentes antes de confirmar una transacción. 5. Falsa: El journaling asegura la durabilidad, no la consistencia. 6. Verdadera: Consistencia estricta aumenta la latencia debido a la sincronización entre nodos. 7. Verdadera: El failover automático evita la pérdida de datos si está bien configurado. 8. Falsa: Los sistemas BASE tienden a ser más escalables que los ACID. |

﻿.

|  |
| --- |
| Pregunta 15:  Al trabajar con Pandas y analizar un conjunto de datos grande, considere el siguiente código:  import pandas as pd  df = pd.read\_csv('ventas.csv')  df['Total'] = df['Cantidad'] \* df['Precio']  ¿Cuál de las siguientes afirmaciones sobre el rendimiento y el uso de memoria es correcta? |
| 1. Si ventas.csv tiene 50 millones de filas, df['Total'] se evaluará perezosamente y solo calculará los valores cuando se acceda a la columna. 2. Utilizar astype('float32') en las columnas Cantidad y Precio reducirá el uso de memoria a la mitad sin pérdida de precisión. 3. df['Total'] crea una vista sobre las columnas Cantidad y Precio en lugar de una copia independiente en memoria. 4. El método eval() puede optimizar el cálculo de df['Total'] al reducir el uso de memoria. 5. El uso de df.loc[:, 'Total'] en lugar de df['Total'] mejora significativamente el rendimiento en este caso. 6. Si la columna Total ya existe, el cálculo será automáticamente vectorizado para evitar recalculaciones. 7. Para grandes conjuntos de datos, es más eficiente utilizar Dask en lugar de Pandas para operaciones de cálculo como la anterior. 8. Utilizar @numba.jit en la función que calcula Total es más eficiente que utilizar directamente la vectorización de Pandas. |
| Respuestas y explicaciones:   1. Falsa: La evaluación de df['Total'] es inmediata y no perezosa. 2. Verdadera: astype('float32') puede reducir el uso de memoria con mínima pérdida de precisión. 3. Falsa: df['Total'] crea una nueva serie, no una vista. 4. Verdadera: eval() puede optimizar el cálculo al reducir el uso de memoria. 5. Falsa: df.loc[:, 'Total'] y df['Total'] son equivalentes en este contexto. 6. Falsa: Pandas no optimiza automáticamente para recalculaciones de columnas existentes. 7. Verdadera: Dask es más eficiente para conjuntos de datos grandes. 8. Falsa: La vectorización de Pandas suele ser más eficiente que numba.jit en este caso. |

﻿.