SQL

1) Es posible tener 2 triggers en una tabla, para un mismo evento y momento del mismo.

   Verdadero. Es posible tener 2 o más triggers en una misma tabla, para un mismo evento, y para un mismo momento. El problema es que nada asegura que el orden de ejecución de los mismos sea predecible.

   También depende del motor de base de datos que se esté utilizando. Algunas versiones antiguas de MySQL solo permiten tener 1 trigger para el mismo evento…

   (Ver: https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch15.htm

   Note: Be aware that triggers of different types are fired in a specific order. However, triggers of the same type for the same statement are not guaranteed to fire in any specific order. For example, all BEFORE ROW triggers for a single UPDATE statement may not always fire in the same order. Design your applications not to rely on the firing order of multiple triggers of the same type.)

2) No es posible realizar un ordenamiento de filas por el número de columna de la relación.

   Falso. Sí es posible realizarlo. Pero no es aconsejable, ya que, en el largo plazo, si le son añadidas nuevas columnas a la relación, se puede llegar a perder el orden original. Es mejor utilizar el nombre de la columna.

3) La sentencia DROP TABLE elimina la definición de la tabla y todos sus datos, índices, triggers, constraints, y permisos sobre esa tabla.

   Verdadero…

   Pero en el caso de que la tabla esté siendo referida por una constraint del tipo FOREING KEY, primero debe eliminarse la constraint FOREING KEY o la tabla que hace uso de dicha constraint para hacer referencia a la tabla que queremos eliminar.

   También cada vista o procedimientos almacenados que hagan referencia a esa tabla (pero que no sean sobre esa tabla) deben ser eliminados manualmente.

4) No es posible usar el mismo trigger para un evento de inserción o de actualización de una fila de una tabla.

   Falso. En Transact-SQL, el mismo trigger puede ser definido para más de una acción (por ejemplo, INSERT y UPDATE) en la misma sentencia CREATE TRIGGER.

   (Ver: Trigger Limitations,

https://docs.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-ver15)

5) Toda condición utilizada en un HAVING debe incluir solamente atributos incluidos en el GROUP BY.

   Falso. HAVING se ejecuta luego del GROUP BY, y antes del SELECT. Se pueden establecer condiciones con cualquiera de los atributos presentes en la tabla luego del agrupamiento.

   Por ejemplo, yo puedo agrupar Empleados por su Legajo, y luego utilizar una condición HAVING que me filtre aquellos que tengan un sueldo menor a $20.000. Sueldo es un atributo que no se encuentra incluido en el GROUP BY.

6) No es posible crear una vista cuya definición involucre una consulta sobre otras vistas.

   Falso. Sí es posible, esta práctica se llama nested views, o vistas anidadas. Su uso no es recomendado ya que puede llevar a un peor rendimiento de las consultas, pero más importante, a una pérdida de legibilidad y entendimiento para el DBA que crea las consultas.

7) Usuario(**idUsuario**, nombre, login, clave, idDepdencia, sector)

   ```
   SELECT idDependencia, count(*)+1 cant
   FROM Usuario
   GROUP BY idDependencia
   HAVING cant > 10 AND sector='Ingresos'
   ORDER BY cant
   ```

   A) No es correcto usar al alias en la secuencia order by. **Falso. Es posible utilizar un alias en el order by, ya que se realiza luego del SELECT, y el DBMS tiene conocimiento del alias.**
   B) No es posible crear una vista actualizable con la sentencia anterior. **Verdadero. Las vistas actualizables no pueden tener funciones de agregación, ni HAVING, ni GROUP BY.**
   C) No es posible referenciar al campo ¿¿?¿? en la condición del HAVING.
   D) Es posible utilizar un alias en el HAVING. **Falso. El HAVING se ejecuta antes del SELECT, por lo que el DBMS todavía no sabe acerca del alias… No puede utilizarlo.**
   E) Es correcto aplicar operaciones en las funciones agregadas en el SELECT. **Verdadero. No existe problema alguno. Incluso se pueden hacer operaciones aritméticas entre dos funciones agregadas, en la misma columna.**

8) Indique la utilidad de la cláusula Except y como puede obtenerse el mismo resultado con los Predicados de Comparación Cuantificados. Dar un ejemplo sobre la base de datos “Viviendas”

   Permite calcular la diferencia entre dos relaciones. Ambas relaciones deben tener el mismo grado y dominio en sus correspondientes atributos.

   Ejemplo: Personas que no son propietarias

Select nif
From Persona Except Select persona From propiedad

Usando Predicados de Comparación:

Select * from Persona where
nif <> all (select persona from propiedad)

9) ¿El resultado de una consulta en SQL es estrictamente siempre una relación?

El resultado de una consulta que incluye en el SELECT a atributos no claves no es estrictamente una relación ya que aparecen tuplas duplicadas. Sin embargo, esto no es un problema ya que podemos eliminar los duplicados usando la cláusula DISTINCT.

10) Indique las distintas formas de ordenamiento admitidas por la cláusula “ORDER BY”

Por defecto se ordena en forma ascendente por la o las columnas que se le indique.

Ordena en forma descendente si se le indica ORDER BY col desc.

Se puede ordenar por una columna que no esta incluida en el select.

Se puede ordenar por varias columnas ORDER BY col1, col2.

Se puede ordenar por una columna derivada (proveniente de un cálculo).

Se puede ordenar por filas resultantes de un agrupamiento.

11) Indique qué características debe tener una vista para poder eliminar registros físicos a través de la misma.

SQL Server no permite la eliminación física de registros, ni siquiera sin vistas de por medio. SQL Server “marca” los registros como borrados, y los ignora.

12) Es posible tener un TRIGGER que se ejecute antes de un evento y otro distinto que se ejecute después del mismo evento sobre la misma tabla.

Verdadero. No hay restricciones en este caso. Primero se ejecutará el trigger BEFORE, y luego el trigger AFTER.

13) El SQL es un lenguaje de programación orientado a eventos.

Falso. Si bien se pueden especificar respuestas ante eventos (triggers), SQL no es principalmente un lenguaje orientado a eventos.

14) Si la clave primaria de una tabla está formada por más de 1 atributo alguno de ellos puede ser NULL.

    Falso. Las claves primarias no pueden tener atributos NULL, ya que necesitan identificar unívocamente a las diferentes tuplas.

15) Es más eficiente ejecutar la sentencia DELETE FROM Tabla1 que la sentencia TRUNCATE TABLE 1.

    Falso. DELETE es más lento que TRUNCATE, ya que DELETE guarda registro de las filas que fueron eliminadas hasta que la transacción sea committeada.

Costo / Optimización de Consultas

1) La optimización algebraica la debe realizar el desarrollador o el DBA, según corresponda, antes de ejecutar una consulta SQL.

    Falso. Los RBDMS tienen optimizadores de consultas, que transforman las consultas en árboles de consultas algebraicos, y realizan una optimización heurística sobre los mismos para transformar el árbol de consulta original (canónico) a un árbol de consultas eficiente.

    La mayoría de los DBMS comerciales tienen un optimizador de consultas que se encarga de realizar la optimización de las mismas, en vez del usuario.

2) ¿Qué son los Hints y cuáles tipos hay? Dé un ejemplo.

    Una sentencia SQL, especialmente una compleja, puede ser ejecutada por el motor de BD en muchísimas formas diferentes (cuál tabla en la junta leer primero, qué índice utilizar basado en los diferentes parámetros, etc).

    Un DBA experimentado puede utilizar los Hints para alentar al motor de BD a elegir un método en particular cuando genera un plan de ejecución.

    Existen varios tipos de Hints: Join Hints, Query Hints, y Table Hints. Las mismas pueden ser aplicadas en las instrucciones SELECT, INSERT, UPDATE, o DELETE.

    Ejemplo, para forzar a que SQL Server utilice MERGE JOIN en una consulta SQL:

```
SELECT *
FROM Sales.Customer AS c
INNER JOIN Sales.CustomerAddress AS ca ON c.CustomerID = ca.CustomerID
WHERE TerritoryID = 5
OPTION (MERGE JOIN);
```

3) El método de junta Hash necesita que las tablas tengan índices en los atributos de junta.

   Falso. Este método se usa generalmente cuando las tablas no están ordenadas ni tienen índices. Es decir, que este método se suele aplicar cuando no es posible utilizar ninguno de los dos métodos anteriores. Cuando SQL Server elije el método Hash join pueden ser una mala señal porque indica que probablemente algo se podría mejorar (por ejemplo, crear un índice).

4) La utilización de índices solo puede impactar en el costo de lectura y no afecta en absoluto el costo de escritura.

   Falso. Los índices pueden afectar el costo de escritura. Tener varios índices en una tabla puede afectar el costo de las operaciones INSERT, UPDATE, y DELETE.

   La operación que más se ve afectada por el uso de índices es la operación INSERT, ya que no obtiene beneficio alguno de ellos. Para insertar una nueva fila, la base de datos debe asegurarse que la fila pueda ser encontrada mediante los índices. Por esta razón, debe añadir una nueva entrada para cada uno de los índices de la tabla. Pero además, debe mantener y asegurar que el orden del resto de los índices...

   En definitiva, los índices afectan al costo de escritura.

5) El optimizador de consultas intenta minimizar los tiempos de respuesta de las sentencias SQL que se ejecutan y para ello toma ciertas decisiones, como por ejemplo, crear índices en las columnas que considere apropiado.

   Falso. El optimizador de consultas nunca crea los índices por su cuenta. Los índices son un trade-off entre costo y beneficio, por lo que el DBMS no puede saber o decidir si es adecuado implementar un índice para una tabla en específico, ya que esta decisión podría ser un detrimento para el rendimiento de futuras consultas.

   “The index design that you put in place is something more of an art than a science. The RDBMS isn't smart enough to take common workloads and design a smart indexing strategy. It is up to human intervention (read: DBA) to analyze workload and determine what is the best approach.

   If there was no penalty of having indexes then it would be a shotgun approach to just add an infinite number of indexes. But because data modification (INSERTS, UPDATES, and DELETES) have impact on the enabled indexes on a table then there is going to be that variable overhead of these indexes.

   It takes human design and strategy to smartly create indexes that'll maximize read performance, while having the least amount of data modification overhead.”

   El DBMS sí puede sugerir la creación de ciertos índices, como hace SQL Server con sus Missing

Index DMV.

6) ¿Cuáles son los métodos de junta que utiliza SQL Server? Explique brevemente cómo funciona el método Sort-Merge.

   SQL Server utiliza principalmente 3 métodos de junta: nested loops, merge join, y hash join.

   Sort-Merge es utilizado cuando ambas tablas están ordenadas físicamente por el atributo de junta, o cuando una se encuentra ordenada y la otra es pequeña y puede ser ordenada relativamente rápido. Luego, el método lee simultáneamente una fila de cada tabla y las compara por el atributo de junta. Si el valor coincide, las filas son incluidas al resultado. Si no coincide, el valor que sea menor es descartado (ya que las tablas están ordenadas, dicho valor menor no será encontrado nunca en la otra tabla).

7) ¿Sobre qué tipo de columnas es conveniente crear índices?

   En aquellas columnas que sea utilizadas para una junta o aquellas que sean utilizadas para una clausula WHERE.

   Es conveniente crear los índices en columnas que estén relacionadas con nuestros patrones de consulta. Podemos tener una tabla con un atributo que no se repita nunca y sea buen candidato para un índice, pero si dicho atributo nunca se utiliza en nuestras consultas para filtrar filas de la tabla, o para realizar una junta con otra tabla, crear dicho índice es innecesario.

8) Una de las reglas de la optimización algebraica indica que se deben aplicas las operaciones de selección lo antes posible. ¿Por qué? ¿Cuál es la ventaja de hacer esto?

   Si bien la selección puede ser un proceso costoso en sí, la misma arroja un número menor de tuplas que aquellas encontradas en la relación original, haciendo que el resto de las operaciones sean menos costosas.

   Las operaciones binarias (como las juntas) aumentan el tamaño del fichero (o relación) con el que estamos trabajando. Por lo tanto, lo aconsejable es primero hacer una selección, y luego hacer una junta u otra operación binaria.

9) Toda tabla debe tener al menos un índice.

   Falso. Los índices no son obligatorios. Podemos tener tablas sin ningún tipo de índice… Pero esto afectará el rendimiento de nuestras consultas.

10) El operador de consultas analiza cada consulta SQL que se intenta ejecutar y decide, entre otras cosas, si es conveniente usar o no índices, y por más que quiera, el usuario no tiene forma de modificar estas decisiones.

Falso. El usuario puede modificar estas decisiones a través del uso de Hints. Los Hints alientan al optimizador de consultas a utilizar métodos en específico. Existen Hints para forzar al optimizador de consultas a que utilice los índices que el usuario desee.

11) ¿Cómo influye la aplicación temprana de las operaciones de proyección en la fase de optimización algebraica?

    La proyección de atributos nos permite trabajar con tablas más chicas, obteniendo un costo para su lectura/escritura menor que la original. Además, nos beneficia en el caso de haber juntas, ya que el resultado va a ser menor (en cuanto a costo y a cantidad de atributos) que si lo hubiese hecho con la relación sin las proyecciones.

12) ¿Cuáles son los factores que componen el costo de producción de una respuesta a una consulta? ¿Cuál de ellos se tiene en cuenta en la fase de optimización de consultas?

    Costo de Comunicaciones: El tiempo que tarda en recuperar los datos desde la fuente hasta desplegar el resultado en la terminal desde donde se realizó la consulta. Este costo es preponderante en el caso de las bases de datos distribuidas. En nuestro curso solo se ha considerado a las bases de datos centralizadas. Donde la base está en un solo repositorio y en el mismo edificio que las terminales de consulta. Por eso este costo es insignificante ya que el tráfico de redes es ínfimo.

    Costo de acceso a memoria secundaria: La cantidad de accesos a disco para levantar los bloques de datos y llevarlos a memoria. Como así también la cantidad de accesos para grabar los bloques con las tuplas resultantes. Este es importante a la hora de evaluar el costo de procesar una consulta.

    Costo de procesador: Tiempo de CPU usado para procesar la consulta. Este costo será despreciable frente a de acceso a disco. Por lo tanto, es preferible consumir más tiempo en la ejecución de un Complejo Algoritmo que minimice el costo de Acceso a disco.

    De estos tres costos solamente será considerado el de acceso a memoria secundaria durante la fase de optimización de consulta.

13) Dada la consulta “select * from R where a <> “x” ¿Cómo podría optimizarla tal que se obtenga el menor costo?

    La operación “distinto a” es una de las más costosas, ya que obliga a leer toda la tabla para encontrar los valores que sean distintos… El optimizador de consultas decide no utilizar índices en estos casos, ya que no otorgan ningún beneficio, y es más barato hacer un escaneo total de la tabla (Full Table Scan).

    En vez de utilizar el operador distinto, intentaría modificar la consulta para que no utilice dicho

operador. Intentaría realizar una nueva consulta que utilice el operador EQUALS or IN, o en su defecto, intentaría utilizar los operadores < y >.

14) Dado el siguiente esquema y la siguiente instrucción SQL:

Pelicula(id, titulo, año, director, cantOscars)

CREATE INDEX Idx_Pelicula ON Pelicula(director, año)

A) En qué caso/casos de WHERE el índice creado anteriormente sería utilizado:
1.a) WHERE director = 'Steven Spilberg'
1.b) WHERE año > 1990
1.c) WHERE director = 'Steven Spillberg' and año > 1990
1.d) UPPER(director) LIKE 'St%'

B) Escriba una consulta en lenguaje SQL en donde sea más óptimo crear el índice idx_Pelicula en forma clúster que en forma no clúster. Justifique.

```
SELECT *
FROM Pelicula
ORDER BY director, año
```

Siendo el índice clúster, la tabla ya estará ordenada por director y año, por lo que se satisfacerá más rápido el ORDER BY, ya que lo único que tendrá que hacer el motor de base de datos es traer los registros tal cual están ordenados por el índice.

15) Si tenemos una tabla que posee índices y sobre ella realizamos dos operaciones (una selección y luego una junta con otra tabla) sus índices solo podrán ser utilizados en la primera operación.

Verdadero, ya que luego de haber aplicado una de las dos operaciones, el resultado que poseemos no es una tabla que tenga índices. El resultado es una nueva "relación/tabla" que no tiene índices asignados.

Seguridad

1) ¿Cuál de las siguientes sentencias permite quietar el permiso de Lectura al usuario User2 para la tabla Tabla2?

A. REVOKE SELECT TO User1 FOR Tabla1
B. REVOKE SELECT TO User1 ON Tabla1
C. REVOKE SELECT TO User1 FROM Tabla1
D. REVOKE SELECT FROM User1 ON Tabla1
E. REVOKE SELECT ON Tabla1 FROM User1
F. REVOKE SELECT ON Tabla1 TO User1

G. REVOKE SELECT ON Tabla1 FOR User1

2) Escribir las sentencias SQL necesarias para realizer lo siguiente:

   Crear un Rol llamado Tester y darle permiso de borrado sobra la tabla Articulo y permiso de consulta sobra la tabla Proveedor, con la posibilidad de otorgar a su vez este último permiso a otros usuarios/roles.
   Finalmente, asignar dicho Rol al usuario RDIAZ.

   CREATE ROLE Tester
   GRANT DELETE ON Articulo TO Tester
   GRANT SELECT ON Proveedor TO Tester WITH GRANT OPTION

   ALTER ROLE Tester ADD MEMBER RDIAZ

3) Crear una copia de seguridad de la base de datos Auditoria y almacenarla en el archivo C:\seguridad\auditoria_2.bak. Luego restaurar en esa misma base la copia de seguridad existente en el archivo C:\seguridad\auditoria_1.bak

   BACKUP DATABASE Auditoria
   TO DISK = 'C:\seguridad\auditoria.bak'

   RESTORE DATABASE Auditoria
   FROM DISK = 'C:\seguridad\auditoria.bak'
   WITH FILE = 2

4) ¿Cuáles de los siguientes métodos se utilizan en el control de acceso discrecional?

   A. Generación de niveles de acceso
   B. Creación de roles
   C. Creación de vistas
   D. Mantener actualizado la nómina de los roles de los usuarios
   E. Todas las anteriores
   F. Ninguna de las anteriores

5) Describa brevemente a qué se denomina "SQL Injection". Dé un ejemplo.

   Las inyecciones SQL son un método de infiltración de código malicioso, el cual se aprovecha de la falta de una correcta validación de las entradas (inputs) que posee una aplicación. Debido a esto, el código inyectado puede realizar acciones que escapan al funcionamiento normal del programa.

   Un ejemplo de inyección SQL sería ingresar en el cuadro de texto Username lo siguiente: Usuario

‘ OR 1=1 DROP TABLE USERS

6) Dadas las siguientes sentencias SQL:

CREATE TABLE #TEST (campo 1 int, campo2 varchar(10))
INSERT INTO #TEST SELECT * FROM Tabla1 WHERE campoT1 = 1
INSERT INTO Tabla2 SELECT * FROM #TEST

A. Estas sentencias deben ser ejecutadas con el usuario “usarapp”. Indicar los permisos necesarios que deben asignarse al usuario para que pueda ejecutar dichas sentencias a través del lenguaje SQL.

Permiso de CREATE TABLE en la base de datos.
Permiso de SELECT para la Tabla1.
Permiso de INSERT para la Tabla2.

Las usuarios tienen automáticamente permisos para crear y usar tablas temporales en SQL Server. Si un usuario tiene acceso a una instancia, tendrá la posibilidad usar tablas temporales.

B. ¿Qué debería hacer si quisiera que cualquier usuario pudiera realizar estas operaciones?

Se podrían añadir a los usuarios a los roles DataReader y DataWriter. Si estos roles son muy permisivos, se podría crear un rol específico que tenga los permisos requeridos en los objetos anteriormente listados, y luego añadir los usuarios a ese rol.

Transacciones

1) ¿Qué propiedad garantiza que el estado de la transacción será público al usuario? Explique.

La propiedad de la durabilidad, ya que especifica que las transacciones no tienen que “perderse” en el tiempo. Es por eso que las transacciones son registradas en un log. Al estar registradas, el estado de la transacción será público al usuario.

2) ¿Cómo hace el motor de bases de datos para garantizar que la concurrencia de las transacciones generará un estado consistente? Explique.

A través del uso de bloqueos, más específicamente, del bloqueo de 2 fases estricto (2PL). Este protocolo mantiene los bloqueos hasta que la transacción termine (ya sea por commit, o por abort). Los bloqueos impiden que otras transacciones realicen cambios en objetos que están bloqueados, garantizando hace consistencia.

3) Indique en el siguiente script cómo afectan las operaciones Commit y Rollback a las transacciones.

BEGIN TRAN

UPDATE EMPLEADOS
SET NyA = 'Juan Perez'
WHERE ID = 101

BEGIN TRAN
UPDATE EMPLEADOS
SET NyA = 'Juan Manuel Perez'
WHERE ID = 101
COMMIT

ROLLBACK

Es posible tener transacciones anidadas. Ahora, si bien es posible, el hecho de tener un solo rollback en cualquier nivel de anidamiento de las transacciones, hace que TODAS las transacciones sean revertidas. En este caso, el ROLLBACK se encuentra en la transacción exterior. Lo que sucederá es que, a pesar de que la transacción interior sea commiteada, cuando se llegue al rollback de la transacción exterior, todos los cambios serán revertidos y las tablas quedarán sin modificaciones.

(Véase: https://sqlstudies.com/2013/12/17/transactions-rolling-back-a-transaction/)

4) La granularidad del lockeo (bloque, fila, tabla) se puede especificar en cada transacción, o bien tomará el valor que posea el motor por defecto.

Falso. La granularidad del lockeo no puede ser especificada, ni tiene valores por defecto. Es el motor de base de datos que decide al momento de ejecutar una transacción, qué granularidad les asignará a los bloqueos producidos por las sentencias SQL, según él crea apropiado.

5) La propiedad de aislamiento de una transacción permite aumentar o disminuir la performance de una consulta.

Falso. Las consultas SQL tendrán el mismo rendimiento independientemente del nivel de aislamiento. Lo que puede llegar a suceder es que, teniendo un nivel de aislamiento menor, podamos ejecutar consultas sin que estas tengan que esperar (lock-wait) por un bloqueo. Pero en sí el rendimiento de la consulta es el mismo siempre: no tendrá menor costo por tener menor nivel de aislamiento la transacción.

6) En una transacción ideal, los lockeos se liberan una vez que no se usen más dentro de la transacción para que otra transacción pueda tomar el recurso y así aumentar la concurrencia.

Falso. Una de las propiedades de las transacciones ideales es el aislamiento, lo cual significa que las transacciones son independientes unas con otras. Al ser independientes, no comparten

recursos, por lo que no podría liberarse el recurso antes de que la transacción termine.

7) Se llama “lock-escalation” cuando el desarrollador escala el nivel de lockeo, por ejemplo, de una página a una tabla.

   Falso. No es el desarrollador el que realiza el lock-escalation. Es el motor de la base de datos. El desarollador lo único que puede es utilizar hints para forzar cierto nivel de granularidad, pero nada más.

8) Cuando se produce un “System Crash” (memoria, cpu, discos) todos los bloques que se encuentren modificados en memoria se perderán en el reinicio del motor.

   Verdadero. La memoria RAM es volátil. Si los bloques modificados no fueron commiteados, ni fueron “checkpointeados”, entonces dichas modificaciones se perderán.

9) El proceso de checkpoint permite actualizar físicamente todas las transacciones confirmadas en el motor.

   Verdadero. Los checkpoints del log de transacciones escriben a disco los cambios que las mismas transacciones realizaron.