AUTOEVALUACIÓN UT4.-Tratamiento de Datos.

1. Indicar cuál de las siguientes sentencias ES CORRECTA:
2. INSERT INTO PRODUCTOS (Codigo, Nombre, Existencias) VALUES (3, Leche, 100);
3. INSERT INTO PRODUCTOS (3, 'Leche', 100);
4. **INSERT INTO PRODUCTOS (Codigo, Nombre, Existencias) VALUES (3, 'Leche', 100);**
5. INSERT INTO PRODUCTOS (Codigo, Nombre, Existencias) VALUES ('Leche', 3, 100);
6. En la sentencia DELETE se borra todo el contenido de la tabla especificada si no se especifica ninguna condición.
7. **Verdadero**.
8. Falso.
9. ¿Cuál de las siguientes sentencias INSERT es la correcta para insertar en la tabla CLIENTES los nombres de los registros de la tabla NUEVOS\_CLIENTES, suponiendo que los campos que contienen los nombres se llaman Nombre\_CLI en la tabla CLIENTES y Nombre\_NCLI en la tabla NUEVOS\_CLIENTES?
10. INSERT INTO CLIENTES (Nombre\_CLI) VALUES Nombre\_NCLI FROM NUEVOS\_CLIENTES;
11. **INSERT INTO CLIENTES (Nombre\_CLI) VALUES (SELECT Nombre\_NCLI FROM NUEVOS\_CLIENTES);**
12. INSERT INTO CLIENTES VALUES (SELECT Nombre\_CLI, Nombre\_NCLI FROM NUEVOS\_CLIENTES);
13. INSERT INTO NUEVOS\_CLIENTES (Nombre\_CLI) VALUES (SELECT Nombre\_NCLI FROM CLIENTES);
14. ¿Cuál de las siguientes sentencias UPDATE es la correcta para actualizar en la tabla USUARIOS el crédito del usuario con código 3 para asignarle el mismo crédito que el del usuario con código 5?
    1. UPDATE USUARIOS SET Credito = Credito WHERE Codigo = 3 AND WHERE Codigo = 5;
    2. UPDATE USUARIOS SET Credito = (SELECT Credito FROM USUARIOS WHERE Codigo = 3 AND WHERE Codigo = 5);
    3. UPDATE USUARIOS SET Codigo = 5 WHERE (SELECT Credito FROM USUARIOS WHERE Codigo = 3);
    4. **UPDATE USUARIOS SET Credito = (SELECT Credito FROM USUARIOS WHERE Codigo = 3) WHERE Codigo = 5;**
15. ¿Cuál de las siguientes sentencias DELETE es la correcta para eliminar de la tabla USUARIOS todos aquellos cuyo código se encuentra en una tabla llamada ANTIGUOS?
    1. **DELETE FROM USUARIOS WHERE Codigo IN (SELECT Codigo FROM ANTIGUOS);**
    2. DELETE FROM USUARIOS WHERE Codigo IN ANTIGUOS;
    3. DELETE FROM (SELECT Codigo FROM ANTIGUOS) WHERE Codigo IN (SELECT Codigo FROM USUARIOS);
    4. DELETE FROM Codigo WHERE USUARIOS IN (SELECT Codigo FROM ANTIGUOS);
16. ¿Se pueden deshacer los cambios con la sentencia ROLLBACK después de que se haya ejecutado COMMIT?
    1. Verdadero.
    2. **Falso.**
17. ¿Qué significan las propiedades ACID de las transacciones en un SGBD?
    1. **Atomicidad, Consistencia, Aislamiento, Durabilidad.**
    2. Atomicidad, Correlación, Aislamiento, Durabilidad.
    3. Atomicidad, Consistencia, Acoplamiento, Durabilidad.
    4. Ninguna de las anteriores.
18. Relaciona los siguientes conceptos sobre los datos con su definición:

|  |  |  |
| --- | --- | --- |
| Concepto |  | Definición |
| Concurrencia |  | Los datos que están siendo consultados o modificados por un usuario no pueden ser cambiados por otros hasta que el usuario haya finalizado la operación completa. |
| Consistencia |  | Los datos y sus estructuras deben reflejar todos los cambios efectuados sobre ellos en el orden correcto. |
| Integridad |  | Asegura que los usuarios pueden acceder a los datos al mismo tiempo. |

1. Si se bloquea una tabla para lectura (con LOCK TABLES), indica cuál de las siguientes afirmaciones es correcta:
   1. **Sólo se permite realizar lecturas sobre la tabla.**
   2. Se pueden realizar lecturas y escrituras sobre la tabla bloqueada
   3. Se pueden realizar escrituras sobre la tabla, pero no lecturas.
   4. Ninguna de las anteriores es correcta.
2. Relaciona las diferentes formas de configurar una transacción (**SET TRANSACTION**) con su definición:

|  |  |  |
| --- | --- | --- |
| Configuración |  | Definición |
| READ UNCOMMITTED |  | Permite acceder a los datos de manera consistente dentro de las transacciones, de manera que todas las lecturas de los datos, dentro de una transacción de tipo REPEATABLE READ, permitirán obtener los datos como al inicio de la transacción, aunque ya hubieran cambiado. |
| READ COMMITTED |  | Se permite acceder a los datos de las tablas, aunque no se haya hecho un commit. Por lo tanto, es posible acceder a datos no consistentes ( dirty read ). |
| REPEATABLE READ |  | Permite acceder a los datos de manera consistente en cualquier lectura de los datos, aunque no nos encontramos dentro de una transacción. |
| SERIALIZABLE |  | Sólo se permite acceder a datos que se hayan aceptado ( commit ) |

1. Relaciona los diferentes tipos de problemas ocasionados por la concurrencia en el acceso a los datos con su descripción:

|  |  |  |
| --- | --- | --- |
| Problema |  | Descripción |
| Dirty Read o Lectura Sucia |  | Una transacción lee unos datos que no existían cuando se inició la transacción. |
| Lectura no repetible |  | Una transacción lee datos escritos por una transacción que no ha hecho COMMIT. |
| Lectura Fantasma. |  | Una transacción vuelve a leer datos que leyó previamente y encuentra que han sido modificados por otra transacción. |

1. Indicar cómo se inicia una transacción en MySQL:
   1. BEGIN**;**
   2. BEGIN WORK;
   3. START TRANSACTION;
   4. **Todas las anteriores son correctas.**
2. Para deshacer los cambios de todas las operaciones que se han realizado bajo una transacción, se utiliza:
   1. COMMIT;
   2. **ROLLBACK;**
   3. Las dos anteriores.
   4. Ninguna respuesta es correcta.
3. Para poder activar las transacciones de múltiples sentencias, se realiza:
   1. **Set autocommit=0;**
   2. Set autocommit=1;
   3. Set autocommit=2;
   4. Ninguna de las anteriores.
4. Si se desea tener la posibilidad de deshacer parte de los cambios que se realizan bajo una transacción , se utilizará:
   1. **SAVEPOINT**.
   2. ROLLBACK**.**
   3. COMMIT.
   4. Ninguna de las anteriores.
5. ¿En una base de datos MySQL con motor de almacenamiento InnoDB pueden producirse deadlocks?
   1. **Sí**.
   2. No.