Teoría De Base De Datos

DER / MR

1) En el DER todas las entidades tienen que tener un atributo asociado directamente que sea clave. Es decir, si hay 3 entidades se tienen mínimamente 3 atributos claves en todo el diagrama.

***Falso**. Podríamos tener un diagrama compuesto por una entidad "Medio De Transporte" con clave "Patente", y dos entidades "Automóvil" y "Ómnibus" que heredan de "Medio De Transporte". Estás últimas tendrán como clave "Patente", pero observaremos que visualmente no hay tres atributos claves distintos en todo el diagrama, sino solo uno.*

2) Dos relaciones unarias pueden convertirse en una relación binaria.

***Falso**. Las relaciones unarias solamente involucran a una única entidad. Por lo tanto, dos relaciones unarias nunca podrían convertirse en una sola relación binaria ya que se necesitaría de dos entidades diferentes y dejaría de cumplir el concepto de relación unaria.*

3) Un atributo calculado no puede ser nunca atributo clave.

***Verdadero**. Una de las características de los atributos claves es que no deben existir repetidos, y si tomamos como clave un atributo calculado estaríamos violando esa regla.*

4) Un mismo MR puede ser generado por distintos DER.

***Verdadero**. Si tuviéramos varios DER que solo variaran respecto a sus atributos calculados, los MR de ellos serían idénticos ya que dichos atributos no serían representados en el MR. También podríamos tener presente el hecho de las relaciones N:N y las entidades que son débiles de otras dos. Estos casos siempre se representan de*

la misma manera en el MR.

5) En el MR si una clave foránea (FK) se encuentra en la clave primaria de una entidad, entonces se puede afirmar que la clave primaria es compuesta.

***Falso**. Una clave primaria se considera compuesta si está formada por dos o más atributos de la entidad a la que hace referencia, es decir, por dos o más columnas de la tabla en cuestión.*

6) En el MR toda relación debe tener al menos un atributo que no sea clave primaria (PK) ni clave foránea (FK), sino no tiene sentido de ser la relación.

***Falso**. Lo importante es que cada entidad pueda ser reconocida y diferenciada del resto. Podríamos tener un MR con entidades que únicamente tengan atributos de clave primaria y ello no dejaría sin sentido de ser a las relaciones.*

7) No existen claves compuestas en las jerarquías.

***Falso**. Es posible tener una entidad "Persona" cuyas subentidades sean "Empleado" y "Estudiante", en la cual la clave sea TIPO + DNI. En este caso estaríamos teniendo una jerarquía con clave compuesta (Más de un atributo clave).*

8) Los atributos compuestos son equivalentes a una entidad débil.

***Falso**. Los atributos que pueden ser convertidos a entidades débiles son los multivaluados.*

9) Un atributo compuesto sirve solo para clarificar.

Verdadero. Esto se debe a que cuando se realiza el MR, el atributo compuesto desaparece, quedando todos los atributos contenidos en este asociados directamente a la entidad en cuestión.

10) El grado de una relación define cuántos elementos de la primer relación se combinan con cuantos de la segunda.

Falso. El grado de una relación define cuantas entidades participan en la misma. La definición expresada en el enunciado corresponde a la cardinalidad de una relación, y es posible que falte indicar que también debería tenerse en cuenta cuantos elementos de la segunda se relacionan con la primera.

11) Colocar un atributo en una relación N1 es equivalente a colocarlo en la entidad donde está la N.

Verdadero. Ya que la entidad que contiene la N será la que obtendrá los atributos de la relación N1, asumiendo que la cardinalidad de la otra entidad es uno.

12) Al realizar el pasaje de DER a MR no hay manera de diferenciar una relación con cardinalidad N:N de una entidad débil.

***Falso**. En el MR una entidad débil tendrá una clave primaria propia y una clave primaria y foránea proveniente de otra entidad, mientras que una relación N:N tendrá una clave propia (O puede no tenerla) más dos o más claves provenientes de otras dos entidades.*

13) En el MR es una buena práctica evitar atributos compuestos, pero en casos excepcionales se puede permitir para simplificar el pasaje.

***Falso**. En el MR nunca se representan los atributos compuestos. Los mismos desaparecen, quedando los atributos que definían al compuesto ligados directamente a la entidad en cuestión.*

NORMALIZACIÓN

1) ¿De qué forma se pueden transformar las “dependencias funcionales” de una relación de 1 FN para que se convierta en una relación de 2 FN?

Eliminando todas las dependencias de claves parciales.

2) ¿Cómo afecta la normalización la posibilidad de tener gran cantidad de valores “nulos” en algunos atributos de la relación? Ejemplifique una solución.

Por ejemplo, si tenemos una tabla de clientes en donde tenemos varias columnas de teléfonos (Teléfono 1; Teléfono 2; etc.) por seguro tendremos columnas con muchos valores nulos. Debemos tener en cuenta que no todos los clientes tendrán tantos teléfonos como para completar todas las columnas. Una solución posible para esto es quitar las columnas que representan teléfonos en la tabla de clientes, y agregar una nueva tabla de teléfonos que se relacione con los clientes. Así si un cliente posee varios teléfonos, entonces tendrá varias tuplas en la nueva tabla, evitando tener valores nulos en los atributos de la relación.

3) ¿De qué forma se pueden transformar las “dependencias transitivas” de una relación en 2 FN para que se convierta en una relación de 3 FN?

Logrando que el determinante sea una super clave o clave candidata, o que el determinado sea primo.

4) Indique cuando son equivalentes dos conjuntos de dependencias funcionales. ¿En qué casos el Fmin es equivalente a F?

Dos conjuntos de dependencias funcionales son equivalentes entre sí cuando las clausuras de todos sus elementos son iguales en ambos conjuntos; es decir; todas las dependencias funcionales del primer conjunto se pueden inferir en el segundo así como todas las del segundo se pueden inferir en el primero. El Fmin es equivalente al F cuando no es posible inferir ninguna dependencia funcional del conjunto mediante otras del mismo aplicando propiedades.

5) Al realizar una descomposición de una relación R en R1 y R2 puede darse que no tengan atributos en común y aún así no perder información.

***Falso**. No es posible ya que por ejemplo, si realizáramos una verificación por Tableu, veríamos que nunca encontraríamos columnas de determinantes con dos o más valores para igualar en los determinados. Por ende, nunca obtendríamos una fila completa de "a", lo cual representa que existe pérdida de información.*

6) Solo puede existir pérdida de información si se pierden dependencias funcionales. Ejemplifique.

***Falso**. Si descomponemos el conjunto de dependencias funcionales en FNBC, es posible que perdiéramos dependencias funcionales pero no existiría pérdida de información.*

7) ¿Qué se obtiene al calcular la clausura de un conjunto atributos X sobre un conjunto dependencias funcionales F?

Se obtienen todos los atributos que son determinados por X pertenecientes al conjunto X.

8) Definir formalmente el concepto de dependencia funcional.

Una dependencia funcional es una relación muchos a uno desde un conjunto de atributos a otro que tiene una relación. Es decir, sea R una relación y sean X e Y subconjuntos arbitrarios del conjunto de atributos de R, se dice que Y es funcionalmente dependiente de X sí y sólo sí cada valor de X en R está asociado con, precisamente, un valor de Y en R.

ALGEBRA RELACIONAL

1) La cardinalidad de RxS es: Cardinalidad(R) + Cardinalidad(S)

***Falso**. La cardinalidad de RxS es: Cardinalidad(R) * Cardinalidad(S).*

2) La relación resultante de una proyección va a tener siempre la misma cantidad de tuplas que la relación original a la que se aplica la operación.

***Falso**. Si en la relación original existen tuplas repetidas respecto al atributo que se está proyectando, la proyección resultante no tendrá en cuenta aquellas tuplas donde se repite el atributo. Por ende, la cantidad de tuplas entre la relación original y la relación resultante será distinta.*

SQL

1) 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**. Los triggers son eventos que se ejecutan sobre una tabla. Existen tres tipos de triggers: Antes (Before), Después (After) y En Reemplazo (Instead Of). Los eventos son independientes entre si, con lo cual es posible tener tres triggers (Distintos) para cada evento de una misma tabla.*

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

***Falso**. SQL es un lenguaje de consultas estructurado. Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.*

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

***Falso**. Una clave nunca puede contener valores nulos sin importar la cantidad de atributos*

que la conformen.

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

***Falso**. Es mas eficiente ejecutar la sentencia TRUNCATE TABLE Tabla1 porque la misma elimina los registros de la tabla sin copiarlos al log de la base de datos (Como lo hace la sentencia DELETE FROM Tabla1), y por lo tanto toma menos tiempo.*

5) Asumiendo que todos los campos utilizados existen en las tablas usadas, indique cuáles considera que son los errores en la siguiente consulta SQL. Justificando cada error encontrado.

```
SELECT Provincia, Localidad, Barrio,
SUM(m2Construidos) AS 'M2Construidos',
MAX(COUNT(idPersona)) AS 'Q habitantes'
FROM parcela p
INNER JOIN localidad l ON l.id = p.id AND l.categoría <> 2
INNER JOIN provincia pr ON pr.id = l.idProv
WHERE Provincia EXISTS('Buenos Aires', 'Córdoba', 'Santa Fe')
GROUP BY Provincia, Localidad
```

***MAX(COUNT**(idPersona)) AS 'Q habitantes' (No se puede utilizar funciones de agregación anidadas)*

***INNER JOIN** localidad l **ON** l.id = p.id **AND** l.categoría <> 2 (Por cuestiones de rendimiento, las condiciones con constantes deberían ir en la cláusula **WHERE**)*

***SELECT** Provincia, Localidad, Barrio, (El atributo “Barrio” debería estar incluido en la cláusula **GROUP BY** para el correcto funcionamiento de la consulta, o en su defecto debería ser eliminado del **SELECT**)*

6) Asumiendo que todos los campos utilizados existen en las tablas usadas, indique cuáles considera que son los errores en la siguiente consulta SQL. Justificando cada error encontrado.

SELECT nomCarrera, **COUNT**(**DISTINCT** c.id), **COUNT**(**DISTINCT** a.id) **AS** 'Qalumnos'
FROM Carrera c
INNER JOIN Materia m **ON** m.idCarrera = c.id
INNER JOIN Alumno A **ON** a.idMateria = m.id
GROUP BY nomCarrera
HAVING DescEstadoAlumno <> 'Inactivo', **COUNT**(**DISTINCT** IDAlumno) > 150
ORDER BY idCarrera

HAVING *DescEstadoAlumno <> 'Inactivo',* ***COUNT(DISTINCT*** *IDAlumno) > 150 (Faltaria un AND, OR, o el operador lógico que corresponda entre ambas condiciones)*

7) La operación junta natural no elimina aquellas tuplas que tengan NULL en los atributos involucrados.

Falso*. La operación junta natural elimina aquellas tuplas que tengan NULL en los atributos involucrados. Para conservar tuplas con atributos NULL se debe hacer uso de juntas LEFT; RIGHT; y FULL.*

8) ¿Cómo se representan en lenguaje SQL las diferentes restricciones del modelo relacional? Escriba las restricciones y un ejemplo de cada una de ellas en lenguaje SQL.

- ***NOT NULL*** *especifica que la columna no acepta valores NULL*

create table paquetes(
codigo CHAR(5) NOT NULL);

- ***CHECK*** *exige la integridad del dominio mediante la limitación de los valores que se*

pueden asignar a una columna.

CREATE TABLE cust_sample(
cust_id int PRIMARY KEY,
CONSTRAINT chk_id CHECK (cust_id BETWEEN 0 and 10000));

- ***UNIQUE** exige la unicidad de los valores de un conjunto de columnas. En una restricción UNIQUE, dos filas de la tabla no pueden tener el mismo valor en las columnas.*

CREATE TABLE usuarios(
usuario VARCHAR(25),
email VARCHAR(50),
UNIQUE usuarios_email_uk (email));

- ***PRIMARY KEY** identifica la columna o el conjunto de columnas cuyos valores identifican de forma exclusiva cada una de las filas de una tabla.*

CREATE TABLE part_sample(
part_nmbr int PRIMARY KEY,
part_name char(30));

- ***FOREIGN KEY** identifica y exige las relaciones entre las tablas.*

CREATE TABLE order_part(
order_nmbr int,
part_nmbr int,
FOREIGN KEY REFERENCES part_sample(part_nmbr) ON DELETE NO ACTION);

COSTOS

1) El factor de bloqueo indica la cantidad de bloques que ocupa una tabla en memoria secundaria (Disco).

***Falso**. El factor de bloqueo es el número de registros lógicos que puede contener como máximo un registro físico o bloque. El factor de bloqueo será > 1 si el tamaño del registro lógico es menor que el del físico, será < 1 si el tamaño del registro lógico es mayor que el del físico, y será = 1 si los dos tienen el mismo tamaño.*

2) 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 primer operación.

***Falso**. Los índices también se utilizan en las juntas. Esto es por la sencilla razón de que cuando se realiza una junta lo que se está haciendo es un producto + una selección + una proyección. Por lo tanto entre estas operaciones se estaría utilizando la selección, en la cual se utilizan los índices. Esto nos da la pauta de que para realizar la junta se estarían utilizando los índices.*

3) El método de junta de iteración por segmentos siempre tendrá un costo menor o igual que el método de iteración por bloques.

***Verdadero**. Porque el método de iteración menos óptimo es el de bloques, ya que en este último leeríamos la segunda tabla de la junta tantas veces como bloques tenga la primera. En el método de iteración por segmentos, teniendo en cuenta que estos son formados por cantidades de bloques variables, leeríamos la segunda tabla de la junta tantas veces como segmentos tenga la primera. Por lo tanto la iteración por segmentos realizaría una menor cantidad de lecturas de la segunda tabla de la junta.*

4) Indique al menos un caso donde un índice de clave primaria no es útil.

Cuando una estructura de datos este ordenada por un campo que no es la clave primaria y pudieran existir repetidos.

5) Indique por qué se optimiza una consulta al aplicar proyecciones luego de las selecciones.

Si aplicamos proyecciones luego de las selecciones, reducimos la cantidad de bloques ocupados por la tabla en memoria, por lo que a menor cantidad de bloques, menor costo y mayor optimización.

6) En las optimizaciones basadas en costos, se debe realizar las selecciones tan pronto como sea posible.

***Verdadero**. Solo que hay que tener en cuenta que no podemos proyectar directamente sobre una tabla sino luego de una operación de selección o de una junta.*

7) Un hash join es más óptimo cuando tenemos presencia de una tabla pequeña y una tabla grande y por lo general es más óptimo que un nested loop o un merge join.

***Falso**. Un hash join es más óptimo ante grandes volúmenes de filas entre las tablas. En el caso de que una de las tablas sea pequeña es recomendable utilizar nested loops, sobre todo si la tabla pequeña tiene un índice.*

8) La utilización de índices en una consulta SQL:

- **Siempre permite reducir los costos de la misma.**
- **No suele ser conveniente.**
- **Puede afectar tanto el costo de lectura como el costo de escritura.**
- **No afecta el costo de lectura ni tampoco el costo de escritura.**
- **Ninguna de las opciones es correcta.**

Puede afectar tanto el costo de lectura como el costo de escritura.

TRANSACCIONES

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

***Falso**. No se puede cambiar la estrategia de bloqueo por cada transacción.*

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

***Verdadero**. Ya que el aislamiento es una propiedad que define cómo y cuándo los cambios producidos por una operación se hacen visibles para las demás operaciones concurrentes. Por lo que un nivel de aislamiento bajo permite más operaciones concurrentes, mientras que uno alto no lo permitiría.*

3) 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**. En una transacción ideal, los bloqueos deberían liberarse una vez que la misma ha sido confirmada. Si el bloqueo fuera liberado antes de que la transacción fuera confirmada, y ante alguna circunstancia esta fallara, otra transacción podría utilizar el recurso modificado por la que falló y dejar la base de datos en un estado inconsistente.*

4) Se llama “Lock Escalation” cuando el desarrollador escala el nivel de lockeo por ejemplo, de una página a una tabla.

***Falso**. El “Lock Escalation” permite promocionar bloqueos de granularidad fina a bloqueos de granularidad más gruesa para reducir la sobrecarga del sistema. En otras palabras convierte un bloqueo específico en otro más general. Por ejemplo, convierte un bloqueo de fila a un bloqueo de objeto completo. El desarrollador puede configurar algunas*

opciones de lock escalation, pero quien lo lleva a cabo es el sistema gestor de la base de datos según la cantidad de bloqueos de granularidad fina que se estén produciendo.

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

***Verdadero**. Esto se debe a que si el motor dejo de responder y se fuerza el cierre del mismo, el sistema operativo liberará los bloques de memoria RAM utilizados por el software para dar lugar a otros programas, de manera que se perderán todos los bloques modificados que no hayan sido guardados antes del system crash. También puede darse el caso de que el system crash afecte a todo el sistema operativo, por lo que si este deja de funcionar, al reiniciar se pierde todo el contenido de la memoria RAM, debido a su condición de volatibilidad.*

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

***Verdadero**. El chekpoint es un registro del log que indica que todos los buffers modificados de la base fueron actualizados al disco. Ninguna transacción T tal que [commit, T] aparece en el Log antes que [checkpoint] necesita Redo. El checkpoint consiste de los siguientes pasos: Suspender la ejecución de todas las transacciones; grabar todos los buffers modificados en el disco; registrar el [checkpoint] en el log y grabar el Log en disco; permitir la continuación de las transacciones.*

7) ¿Qué problemas pueden ocurrir si no se controla la concurrencia en las transacciones?

Actualización perdida; actualización temporal; suma o resumen incorrecto; y lectura irrepetible.

8) Toda transacción que ejecuta un motor de base de datos debe cumplir con las

propiedades ACID, de lo contrario el motor rechaza la transacción.

***Falso**. Cumplir las propiedades ACID garantiza que no habrá conflictos con lecturas; que los datos de la base serán consistentes; y evitar muchos otros problemas que podrían suceder si no se las cumple. Pero esto no significa que el motor rechace las transacciones que no cumplan con dichas propiedades. De hecho; en la práctica; es difícil cumplir todas estas propiedades.*

9) En el protocolo de marcas temporales:

- **Cada transacción cuenta con una marca temporal.**
- **Hay un ordenamiento secuencial de transacciones.**
- **Hay un ordenamiento paralelo de transacciones.**
- **Las transacciones pueden caer en ciclos interminables.**
- **Se producen interbloqueos.**
- **No se producen interbloqueos.**

Cada transacción cuenta con una marca temporal; Hay un ordenamiento secuencial de transacciones; No se producen interbloqueos.

10) Indique la propiedad con la que el Sistema Gestor de la Base De Datos garantiza que una transacción siempre finalice.

***Atomicidad**: Es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.*

11) Indique y explique la propiedad con la que el Sistema Gestor de Base de Datos

garantiza que las transacciones no se interfieran entre sí.

***Aislamiento**: Es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean*

independientes y no generen ningún tipo de error. Esta propiedad define cómo y cuándo los cambios producidos por una operación se hacen visibles para las demás operaciones concurrentes.

12) Enuncie y explique cómo el motor de base de datos garantiza el servicio de bloqueo a una fila de una tabla para que dos procesos no puedan actualizar esa misma fila en el mismo instante.

El motor de base de datos realizará un lock_x(elemento). De esta manera el elemento estará bloqueado para su escritura mientras la primera transacción se esté ejecutando sobre ese elemento. Cuando finaliza desbloquea el elemento y el mismo puede ser utilizado para la ejecución de la otra transacción.

13) ¿En qué consiste la propiedad de Durabilidad y en qué situación se lleva a cabo?

La durabilidad es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema y que de esta forma los datos sobrevivan de alguna manera.

XML

1) Explique como se define la gramática de los lenguajes XML, es decir, la estructura y cuales son los elementos permitidos en los documentos XML.

La gramática de los lenguajes XML está basada en etiquetas:

Tag de inicio: <etiqueta_comienzo>
Tag de fin: </etiqueta_cierre>
Línea vacía: <etiqueta_sin_contenido />

Los contenidos deben estar entre las etiquetas, como por ejemplo:

<nombre> Jorge </nombre>

Las etiquetas pueden contener otras etiquetas:

```
<empleado>
    <nombre> Jorge </nombre>
    <apellido> Sala </apellido>
</empleado>
```

Las etiquetas también pueden contener atributos:

<nombre largo_max = “20” largo_min = ”3”> Jorge </nombre>

Para definir un XML de manera adecuada debemos:

Poner un elemento único del tipo raíz (O root); corresponder etiquetas de apertura y cierre, respetando el anidamiento para los elementos creados; y si cada elemento tiene atributos, estos deben estos ser únicos.

```
<raiz>
    <nodo> texto </nodo>
    <nodo/>
    <nodo>
        <subnodo>
            <subsubnodo>
                <texto> texto </texto>
            </subsubnodo>
        <subnodo>
    </nodo>
</raiz>
```

2) Indique que definimos cuando definimos un DTD.

Al definir un DTD estamos definiendo qué etiquetas se definen y qué atributos tiene cada una de ellas; cuál será el orden de las etiquetas, es decir, cual irá primero y cuál después; y como será el anidamiento, es decir, cómo irán definidas unas dentro de otras.

3) Indique cuales son las formas de almacenamiento de un archivo .xml dentro de una base de datos y especifique para cada caso un ejemplo que justifique ese tipo de almacenamiento.

Por deducción lógica, una base de datos nativa en XML almacena la información en formato XML, pero esto es solamente una deducción lógica, pues este tipo de bases de datos tienen repositorios con un formato "tipo XML", como puede ser DOM o Infoset. En este mismo "repositorio" (paquete de archivos) se almacenan los índices que se generan por cada documento XML almacenado. En el caso de DOM se utiliza para acceder; añadir; y cambiar dinámicamente el contenido de un documento. Infoset se utiliza cuando existen especificaciones que necesitan referirse a la información de un documento XML bien formado.

SEGURIDAD

1) Indique como se utilizan los permisos en cascada y como se definen.

Cuando a un usuario se le otorgan permisos en cascada, el mismo puede otorgar dichos permisos (O de menor privilegio) a otros si este tiene permisos para dar permisos. Si el usuario que dio el permiso decide eliminarlo con la primitiva de cascada, se eliminarán también los permisos obtenidos por todos los usuarios de la cascada, pero si no utiliza la primitiva de cascada, solo se lo quitará al usuario que especifique sin afectar al resto.

2) Cuales de los siguientes métodos se utilizan en el control de acceso

discrecional:

- **Generación de niveles de acceso.**
- **Creación de roles.**
- **Creación de vistas.**
- **Mantener actualizado la nómina de roles de usuarios.**
- **Todas las anteriores.**
- **Ninguna de las anteriores.**

Creación de vistas.

3) Un algoritmo de encriptación simétrico consta de:

- **Algoritmo de desencriptación.**
- **Texto plano.**
- **Texto cifrado.**
- **Algoritmo de encriptación.**
- **Clave pública.**
- **Todas las anteriores.**
- **Ninguna de las anteriores.**

Algoritmo de encriptación; Algoritmo de desencriptación; Texto cifrado.

4) Describa brevemente a qué se denomina SQL Injection.

SQL Injection es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar operaciones sobre una base de datos. Por ejemplo, el usuario ingresa la siguiente instrucción en un cuadro de texto que corresponde a ún código de producto que desea consultar: 1; DROP TABLE PRODUCTOS. Si no se valida la información ingresada en el cuadro de texto, se podría borrar toda la tabla de productos de la base de datos.

BASES DE DATOS DISTRIBUIDAS

1) Indique cuales son las técnicas de fragmentación y replicación en las bases de datos distribuidas.

Fragmentación:

Fragmentación Horizontal: La fragmentación horizontal de una relación R produce una serie de fragmentos en los que cada uno de ellos contiene un subconjunto de las tuplas de R que cumplen determinadas propiedades.

Fragmentación Vertical: La fragmentación vertical de una relación R produce una serie de fragmentos en los que cada uno de ellos contiene un subconjunto de los atributos de R así como la clave primaria de R.

Fragmentación Mixta: La fragmentación mixta es la combinación de las dos técnicas anteriores. Existen varias formas de aplicar la fragmentación mixta, aunque lo más común es desarrollar primero la fragmentación vertical, y posteriormente aplicar la partición horizontal sobre los fragmentos verticales; o bien aplicar primero una división horizontal para luego, sobre los fragmentos generados, desarrollar una fragmentación vertical.

Replicación:

Replicación Total: Una base de datos totalmente replicada guarda varias copias de cada fragmento de la base de datos en varios sitios.

Replicación Parcial: Una base de datos parcialmente replicada guarda múltiples copias de algunos fragmentos de la base de datos en múltiples sitios

No Replicación: Una base de datos no replicada guarda cada fragmento de base de

datos en un solo sitio.