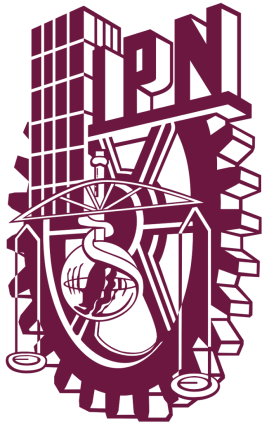
****

**Instituto Politecnico Nacional**

**ESCOM “Escuela Superior de Cómputo”**

INGENIERÍA SISTEMAS COMPUTACIONALES

*Bases de datos*

*Cuestionario libro de* Elmasri, Ramez & Navathe, Shamkant. *Fundamentals of Database Systems*

PROFE: Sergio Salinas Lugo

ALUMMNO: Rojas Alvarado Luis Enrique

GRUPO: 2CM2

1-¿Cuáles son las responsabilidades de un DBA y de un diseñador de bases de datos?

DBA

En un entorno de base de datos, el recurso principal es la base de datos en sí misma, y ​​el recurso secundario es el DBMS y software relacionado

La administración de estos recursos es responsabilidad del administrador de la base de datos (DBA). El DBA es responsable de autorizar el acceso a la base de datos, coordinando y monitoreando su uso, y adquiriendo recursos de software y hardware según sea necesario.

El DBA es responsable de problemas tales como violaciones de seguridad y un tiempo de respuesta del sistema pobre. En organizaciones grandes, el DBA es asistido por un personal que lleva a cabo estas funciones.

DISEÑADOR

Los diseñadores de bases de datos son responsables de identificar los datos que se almacenarán en la base de datos y de elegir las estructuras adecuadas para representar y almacenar estos datos.

Es responsabilidad de los diseñadores de bases de datos comunicarse con todos los posibles usuarios de la base de datos para comprender sus requisitos y crear un diseño que cumple con estos requisitos. (Elmasri & Navathe, 2011, p.15).

Opino que hay que tomar muy en cuenta estas responsabilidades ya que la información que ingresemos a la base de datos se vuelve nuestra responsabilidad ya que somos responsables de que esa información sea segura y que no cualquiera tenga acceso a esta información.

2-¿Cuáles son las diferencias básicas entre modelo relacional, modelo orientado a objetos y modelo XML?

El modelo básico de datos relacionales representa una base de datos como una colección de tablas, donde cada tabla se puede almacenar como un archivo separado. La mayoría de las bases de datos relacionales usan el lenguaje de consulta de alto nivel llamado SQL y admiten una forma limitada de vistas de usuario.

El modelo de datos de objeto define una base de datos en términos de objetos, sus propiedades y sus operaciones. Los objetos con la misma estructura y comportamiento pertenecen a una clase, y las clases están organizadas en jerarquías (o gráficos acíclicos). Las operaciones de cada clase se especifican en términos de procedimientos predefinidos llamados métodos.

El modelo XML se ha convertido en un estándar para el intercambio de datos a través de la Web y se ha utilizado como base para implementar varios prototipos de sistemas XML nativos. XML usa estructuras jerárquicas de árbol. Combina conceptos de bases de datos con conceptos de modelos de representación de documentos. Los datos se representan como elementos; con el uso de etiquetas, los datos se pueden anidar para crear estructuras jerárquicas complejas. Este modelo se parece conceptualmente al modelo de objetos, pero utiliza una terminología diferente. Las capacidades XML se han agregado a muchos productos DBMS comerciales. (Elmasri&Navathe, 2011, pp.50-51).

La diferencia entre las 3 Es su jerarquía ya que en uno se mira cómo se han estado utilizando los datos, en otro como se implementan y en el último como se almacenan.

3-¿Cuál es la diferencia entre lenguaje de manipulación de datos procedural y no procedural?

Un lenguaje de manipulación de datos (DML) se utiliza para especificar recuperaciones y actualizaciones de bases de datos. Los LMD pueden ser de alto nivel (orientado a conjuntos, no procedimental) o de bajo nivel (orientado a registros, de procedimiento). Un DML de alto nivel puede integrarse en un lenguaje de programación de host, o puede usarse como un lenguaje independiente; en este último caso, a menudo se lo denomina lenguaje de consulta. (Elmasri&Navathe, 2011, p.53).

Básicamente un no procedural especifica las operaciones en una base de datos complejas y un procedural está ligado al idioma de la programación para recuperar registros y trabajarlos por separado.

4-¿Cuál es la diferencia entre independencia lógica de datos e independencia física de datos, indique cuál es más difícil de lograr y por qué?

La independencia de los datos lógicos es la capacidad de cambiar el esquema conceptual sin tener que cambiar esquemas externos o programas de aplicación. Podemos cambiar el esquema conceptual para expandir la base de datos (agregando un tipo de registro o elemento de datos), cambiar las restricciones o reducir la base de datos (eliminando un tipo de registro o elemento de datos). En el último caso, los esquemas externos que se refieren solo a los datos restantes no deberían verse afectados.

La independencia de los datos físicos es la capacidad de cambiar el esquema interno sin tener que cambiar el esquema conceptual. Por lo tanto, los esquemas externos no necesitan ser cambiados también. Los cambios en el esquema interno pueden ser necesarios porque algunos archivos físicos se reorganizaron, por ejemplo, al crear estructuras de acceso adicionales, para mejorar el rendimiento de la recuperación o actualización. Si los mismos datos que antes permanecen en la base de datos, no deberíamos tener que cambiar el esquema conceptual

La independencia lógica de datos es más difícil de lograr porque permite cambios estructurales y de restricciones sin afectar los programas de aplicación, un requisito mucho más estricto. (Elmasri & Navathe, 2011, pp.35-36).

Nosotros podemos cambiar el esquema conceptual para expandir la base de datos (agregando un tipo de registro o elemento de datos), para cambiar las restricciones o para reducir la base de datos (eliminando un tipo de registro o elemento de datos).

En general, la independencia de los datos físicos existe en la mayoría de las bases de datos y entornos de archivos donde los detalles físicos como la ubicación exacta de los datos en el disco y el hardware detalles de codificación de almacenamiento, ubicación, compresión, división, fusión de registros, etc., están ocultos para el usuario. Las aplicaciones no tienen conocimiento de estos detalles.

5-Discuta las restricciones de la integridad de la entidad y la integridad referencial y describa porqué son importantes.

“La restricción de integridad de la entidad establece que ningún valor de clave principal puede ser NULL. Esto se debe a que el valor de la clave primaria se usa para identificar tuplas individuales en una relación.

Tener valores NULL para la clave primaria implica que no podemos identificar algunas tuplas. Por ejemplo, si dos o más tuplas tienen NULL para sus claves principales, es posible que no podamos distinguirlas si tratamos de hacer referencia a ellas desde otras relaciones

La restricción de integridad referencial se especifica entre dos relaciones y se usa para mantener la coherencia entre tuplas en las dos relaciones. Informalmente, la restricción de integridad referencial establece que una tupla en una relación que se refiere a otra relación debe referirse a una tupla existente en esa relación.“ (Elmasri&Navathe, 2011, p.73).

Al poner una llave primaria directo a NULL lo que estamos haciendo es decir que esa llave no tiene un valor o que toma valores nulos pero no podremos distinguirlas al hacer el llamado desde una llave foránea y no tendría coherencia.

6- ¿Que es una transacción y cómo difiere de una operación de actualización?

“Una transacción es un programa en ejecución que incluye algunas operaciones de base de datos, como leer desde la base de datos o aplicar inserciones, eliminaciones o actualizaciones a la base de datos

Una sola transacción puede implicar cualquier número de operaciones de recuperación,y cualquier cantidad de operaciones de actualización. Estas recuperaciones y actualizaciones juntas formarán una unidad de trabajo atómica contra la base de datos.” (Elmasri&Navathe, 2011, p.79).

En general, la independencia de los datos físicos existe en la mayoría de las bases de datos y entornos de archivos donde los detalles físicos como la ubicación exacta de los datos en el disco y el hardware detalles de codificación de almacenamiento, ubicación, compresión, división, fusión de registros, etc., están ocultos para el usuario. Las aplicaciones no tienen conocimiento de estos detalles.

7. ¿Cómo permite la implementación de SQL de la integridad de la entidad y de la integridad referencial?

Elmasri & Navathe describen las restricciones básicas que se pueden especificar en SQL como parte de la creación de tablas, con la que se permite la integridad de la entidad (llave primaria) y de la integridad referencial (llave foranea).

Debido a que las claves y las restricciones de integridad referencial son muy importantes, existen cláusulas especiales dentro de la sentencia CREATE TABLE para especificarlas.

La cláusula PRIMARY KEY especifica uno o más atributos que componen la clave primaria de una relación. Si una clave principal tiene un solo atributo, la cláusula puede seguir el atributo directamente. Por ejemplo, la clave principal de DEPARTAMENTO se puede especificar de la siguiente manera (en lugar de la forma en que se especifica en la Figura 4.1):

Dnumber INT **PRIMARY KEY**;

(Elmasri & Navathe, 2011, pp. 95)

Mencionando también:

“La integridad referencial se especifica a través de la cláusula FOREIGN KEY, como se muestra en la figura 4.1.”

**-----------------------FIGURA 4.1-----------------------**

**CREATE** **TABLE** DEPARTMENT

**(**Dnumber **INT** **NOT** **NULL,**

Mgr\_ssn **CHAR(**9**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**Dnumber**),**

**FOREIGN** **KEY** **(**Mgr\_ssn**));**

(Elmasri & Navathe, 2011, pp. 96)

Es una referencia a otra tabla que contiene atributos que pueden ser usados en una tabla aparte, es por eso que se declara como llave primaria o foránea.

8. Describa la sintaxis de las seis cláusulas de la sentencia SQL para recuperación de datos (query). Indique cuales se requieren y cuales son opcionales.

Las cláusulas se especifican en el siguiente orden, con las cláusulas entre corchetes […] siendo opcionales.

SELECT <lista de atributos y funciones>

FROM <nombre de la tabla>

[ WHERE <condición>]

[GROUP BY < atributo (s) de agrupación>]

[ HAVING <condición de grupo>]

[ ORDER BY <lista de atributos>];

(Elmasri & Navathe, 2011, pp. 129)

Estos son comandos para visualización de los datos, para que sea más cómodo a la vista y para que se vea que datos se están desplegando para consultar.

9. Describa cada una de las siguientes construcciones utilizadas en SQL, consultas o queries anidados, reunión de tablas, funciones de acumulación y agrupamiento, Triggers, vistas.

Respecto a cada una de las construcciones utilizadas en SQL, Elmasri & Navathe describen:

🡺Consultas/Consultas Anidadas:

“Algunas consultas requieren que los valores existentes en la base de datos se busquen y luego se utilicen en una condición de comparación. Dichas consultas se pueden formular de forma conveniente mediante el uso de consultas anidadas, que son bloques completos de selección de origen dentro de la cláusula WHERE de otra consulta. Esa otra consulta se llama consulta externa. Query 4 está formulado en Q4 sin una consulta anidada, pero puede reformularse para usar consultas anidadas como se muestra en Q4A. Q4A introduce el operador de comparación IN, que compara un valor v con un conjunto (o multiset) de valores V y lo evalúa como TRUE si v es uno de los elementos en V.

-------------------------------Q4A:-------------------------------

**SELECT** **DISTINCT** Pnumber

**FROM** PROJECT

**WHERE** Pnumber **IN**

**(** **SELECT** Pnumber

**FROM** PROJECT**,** DEPARTMENT**,** EMPLOYEE

**WHERE** Dnum**=**Dnumber **AND**

Mgr\_ssn**=**Ssn **AND** Lname**=**‘Smith’ **)**

**OR**

Pnumber **IN**

**(** **SELECT** Pno

**FROM** WORKS\_ON**,** EMPLOYEE

**WHERE** Essn**=**Ssn **AND** Lname**=**‘Smith’ **);**

”

(Elmasri & Navathe, 2011, pp. 117)

“En general, la consulta anidada devolverá una tabla (relación), que es un conjunto o multiset de tuplas.

SQL permite el uso de tuplas de valores en las comparaciones al colocarlas entre paréntesis. Para ilustrar esto, considere la siguiente consulta:

**SELECT** **DISTINCT** Essn

**FROM** WORKS\_ON

**WHERE** **(**Pno**,** Hours**)** **IN** **(** **SELECT** Pno**,** Hours

**FROM** WORKS\_ON

**WHERE** Essn**=**‘123456789’ **);**

Esta consulta seleccionará los Essns de todos los empleados que trabajen igual (proyecto, horas) combinación en algún proyecto que el empleado 'John Smith' (cuyo Ssn = '123456789')”

(Elmasri & Navathe, 2011, pp. 118)

🡺Reunión de Tablas:

“El concepto de una tabla unida (o relación unida) se incorporó a SQL para permitir a los usuarios especificar una tabla resultante de una operación de unión en la cláusula FROM de una consulta. Esta construcción puede ser más fácil de comprender que mezclar todas las condiciones de selección y unión en la cláusula WHERE. Por ejemplo, considere la consulta Q1, que recupera el nombre y la dirección de cada empleado que trabaja para el departamento de 'Investigación'. Puede ser más fácil especificar la unión de las relaciones EMPLEADO y DEPARTAMENTO primero, y luego seleccionar las tuplas y atributos deseados. Esto se puede escribir en SQL como en Q1A:

-----------------------Q1A-----------------------

**SELECT** Fname**,** Lname**,** Address

**FROM** **(**EMPLOYEE **JOIN** DEPARTMENT **ON** Dno**=**Dnumber**)**

**WHERE** Dname**=**‘Research’**;**

La cláusula FROM en Q1A contiene una sola tabla unida. Los atributos de dicha tabla son todos los atributos de la primera tabla, EMPLEADO, seguidos de todos los atributos de la segunda tabla, DEPARTAMENTO.

El concepto de una tabla unida también permite al usuario especificar diferentes tipos de combinación, como NATURAL JOIN y varios tipos de OUTER JOIN. En una UNIÓN NATURAL en dos relaciones R y S, no se especifica ninguna condición de unión; se crea una condición EQUIJOIN implícita para cada par de atributos con el mismo nombre de R y S.”

(Elmasri & Navathe, 2011, pp. 123)

🡺Funciones de acumulación y agrupamiento:

“Las funciones agregadas se usan para resumir información de múltiples tuplas en un resumen de un solo tuplo. La agrupación se usa para crear subgrupos de tuplas antes del resumen. La agrupación y la agregación son necesarias en muchas aplicaciones de bases de datos”

(Elmasri & Navathe, 2011, pp. 124)

“Existe una cantidad de funciones agregadas integradas: COUNT, SUM, MAX, MIN y AVG. La función COUNT devuelve la cantidad de tuplas o valores especificados en una consulta. Las funciones SUM, MAX, MIN y AVG se pueden aplicar a un conjunto o multiset de valores numéricos y devolver, respectivamente, la suma, el valor máximo, el valor mínimo y el promedio (promedio) de esos valores. Estas funciones se pueden utilizar en la cláusula SELECT o en una cláusula HAVING. Las funciones MAX y MIN también se pueden usar con atributos que tienen dominios no numéricos si los valores del dominio tienen un orden total entre sí.”

(Elmasri & Navathe, 2011, pp. 125)

“En muchos casos queremos aplicar las funciones agregadas a subgrupos de tuplas en una relación, donde los subgrupos se basan en algunos valores de atributo. Por ejemplo, es posible que deseemos encontrar el salario promedio de los empleados en cada departamento o la cantidad de empleados que trabajan en cada proyecto. En estos casos, necesitamos dividir la relación en subconjuntos (o grupos) no permanentes de tuplas. Cada grupo (partición) consistirá en las tuplas que tienen el mismo valor de algún atributo (s), llamado atributo (s) de agrupamiento. Luego podemos aplicar la función a cada uno de esos grupos de forma independiente para producir información resumida sobre cada grupo. SQL tiene una cláusula GROUP BY para este propósito. La cláusula GROUP BY especifica los atributos de agrupación, que también deberían aparecer en la cláusula SELECT, de modo que el valor resultante de aplicar cada función agregada a un grupo de tuplas aparezca junto con el valor de los atributos de agrupación.”

(Elmasri & Navathe, 2011, pp. 126)

“SQL proporciona una cláusula HAVING, que puede aparecer junto con una cláusula GROUP BY. HAVING proporciona una condición en la información de resumen con respecto al grupo de tuplas asociadas con cada valor de los atributos de agrupación. Solo los grupos que satisfacen la condición se recuperan en el resultado de la consulta.”

(Elmasri & Navathe, 2011, pp. 128)

🡺Triggers

“TRIGGER, se puede usar para especificar acciones automáticas que el sistema de base de datos realizará cuando ocurran ciertos eventos y condiciones. Este tipo de funcionalidad generalmente se conoce como bases de datos activas.”

(Elmasri & Navathe, 2011, pp. 131)

🡺Vistas

Elmasri & Navathe define“Una vista como una tabla simple que se deriva de otras tablas, estas tuplas pueden formarse a partir de tablas base o de otras vistas previamente definidas”

(Elmasri & Navathe, 2011, pp. 133)

Dentro de una tabla se va llamando a otra con otros atributos para consultar los datos guardados en cada una de ellas y así la visualización y ordenar como sea más cómodo para la vista y para que el que consulte la base de datos sepa que entidades y que atributos son los que está buscando.

Se hace más efectiva la forma de ver los datos en tablas cuando éstas están siendo referenciadas por otras, ordenándolas para que se encuentre más rápido la información solicitada

10. Defina las operaciones unarias y binarias del álgebra relacional. Describa cual es la propiedad de compatibilidad a la unión.

Las operaciones SELECT y PROJECT son operaciones unarias las cuales operan en relaciones individuales.

 SELECT: esta se usa para elegir un subconjunto de las tuplas en una relación que satisface una condición de selección, en otras palabras, se puede considerar como un filtro que mantiene solo aquellas tuplas que satisfacen la condición especificada

 PROJECT: a diferencia de SELECT este comando selecciona ciertas columnas en vez de filas y descarta otras. Se usa solo si se desean saber ciertos atributos de una relación y así proyectar la relación solamente sobre estos atributos.

Las operaciones binarias son JOIN y DIVISION se denominan así debido a que operan en dos tablas mediante la combinación de tuplas relacionadas

 JOIN: se usa para combinar tuplas relacionadas de dos relaciones en tuplas únicas “más largas”. Esta operación se puede especificar como un producto cartesiano (CARTESIAN PRODUCT) seguida de una operación SELECT. Sin embargo, JOIN es muy importante porque se usa con mucha frecuencia cuando se especifican las consultas a la base de datos.

 DIVISION: esta operación es indicada por ÷, es útil para un tipo especial de consulta que a veces ocurre en aplicaciones de base de datos. Por ejemplo, recuperar los nombres de los empleados que trabajan en todos los proyectos en los que ‘John Smith’ trabaja.

En general, esta operación se aplica a dos relaciones R(Z) ÷ S(X), donde los atributos de R son un subconjunto de los atributos S; es decir, X ⊆ Z. Sea Y el conjunto de atributos de R que no son atributos de S; es decir, Y = Z - X (y por lo tanto Z = X ∪ Y)

(Elmasri & Navathe, 2011, pp. 147-150)

Éstas son operaciones operacionales que despliegan información de acuerdo a los datos ingresados y en base a éstos se harán más tuplas o más columnas para consultar y se hará más completa una base de datos porque toda la información se encuentra en una sola tabla, pero a la vez se complica porque la cantidad de información puede ser demasiada al punto de confundir al que la consulta.

11. Describa los términos entidad, atributo, valor atributo, atributo multivaluado, atributo complejo, atributo llave, conjunto de valores o dominio.

El modelo ER describe los datos como entidades, relaciones y atributos.

🡺Entidad

“El objeto básico que representa el modelo ER es una entidad, que es una cosa en el mundo real con una existencia independiente. Una entidad puede ser un objeto con una existencia física (por ejemplo, una persona, un automóvil, una casa o un empleado en particular) o puede ser un objeto con una existencia conceptual (por ejemplo, una empresa, un trabajo o un curso universitario).”

(Elmasri & Navathe, 2011, pp. 203)

🡺Atributo

“Cada entidad tiene atributos, las propiedades particulares que lo describen.”

(Elmasri & Navathe, 2011, pp. 203)

🡺Valor Atributo

“El valor de un atributo compuesto es la concatenación de los valores de sus atributos simples, componentes.

\*Los atributos compuestos se pueden dividir en subpartes más pequeñas, que representan atributos más básicos con significados independientes.”

(Elmasri & Navathe, 2011, pp 205-206)

🡺Atributo multivaluado

“Un atributo multivalor puede tener límites inferior y superior para limitar el número de valores permitidos para cada entidad individual. Por ejemplo, el atributo Colores de un automóvil puede estar restringido para tener entre uno y tres valores, si suponemos que un automóvil puede tener tres colores como máximo.”

(Elmasri & Navathe, 2011, pp. 206)

🡺Atributo complejo

“Atributos complejos. Observe que, en general, los atributos compuestos y multivalor se pueden anidar arbitrariamente. Podemos representar el agrupamiento arbitrario agrupando los componentes de un atributo compuesto entre paréntesis () y separando los componentes con comas, y mostrando atributos multivalor entre llaves {}. Tales atributos se llaman atributos complejos.”

(Elmasri & Navathe, 2011, pp. 206)

🡺Atributo llave

“Atributos clave de un tipo de entidad. Una restricción importante en las entidades de un tipo de entidad es la restricción de clave o exclusividad en los atributos. Un tipo de entidad generalmente tiene uno o más atributos cuyos valores son distintos para cada entidad individual en el conjunto de entidades. Tal atributo se denomina atributo clave, y sus valores se pueden usar para identificar cada entidad de manera única.”

(Elmasri & Navathe, 2011, pp. 208)

🡺Conjunto de valores o dominio

“Conjuntos de valores (dominios) de atributos. Cada atributo simple de un tipo de entidad se asocia con un conjunto de valores (o dominio de valores), que especifica el conjunto de valores que se pueden asignar a ese atributo para cada entidad individual.”

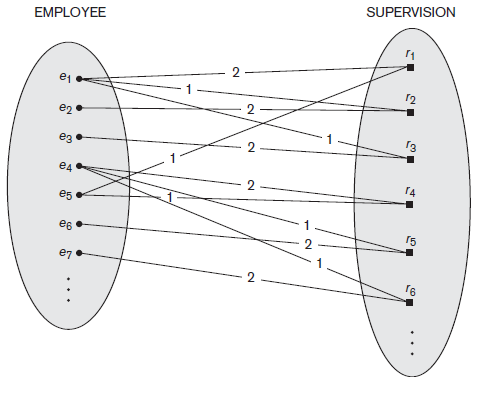
(Elmasri & Navathe, 2011, pp. 209).

Pienso que es importante conocer estos conceptos porque al momento de hacer la base de datos debes saber que datos estás apunto de colocar como llaves, atributos y sus respectivos valores, para que al momento que los ingreses, no cometas un error de no saber cómo declarar cada uno de los datos que vas a ingresar a la base.

12. ¿Qué significa ser una relación recursiva?, dé algunos ejemplos.

“Puede suceder que el mismo tipo de entidad participa más de una vez en un tipo de relación en roles diferentes. En tales casos, el nombre del rol se vuelve esencial para distinguir el significado del rol que desempeña cada entidad participante. Tales tipos de relación se llaman relaciones recursivas.

Ejemplo: El tipo de relación SUPERVISIÓN relaciona a un empleado con un supervisor, donde tanto las entidades empleadoras como las supervisoras son miembros de este conjunto de entidades EMPLEADO. Por lo tanto, el tipo de entidad EMPLEADO participa dos veces en SUPERVISIÓN: una vez en el rol de supervisor (o jefe), y una vez en el rol de supervisado (o subordinado). Cada instancia de relación ri en SUPERVISIÓN asocia dos entidades de empleado ej y ek, una de las cuales desempeña el papel de supervisor y la otra la función de supervisado. En la figura que se muestra abajo, las líneas marcadas con '1' representan al rol de supervisor, y los marcados '2' representan el rol de supervisado; por lo tanto, e1 supervisa e2 y e3, e4 supervisa e6 y e7, y e5 supervisa e1 y e4. En este ejemplo, cada instancia de relación debe estar conectada con dos líneas, una marcada con '1' (supervisor) y el otro con '2' (supervisado).” (Elmasri&Navathe, 2011, p.215)



Ser recursiva se refiere a que se utiliza así misma para que de forma única se llegue a un destino y no se repita más de una vez.

13. Discuta los tipos principales de restricciones en la especialización y generalización.

“La *especialización* es el proceso de definir un conjunto de subclases de un tipo de entidad; este tipo de entidad se llama la superclase de la especialización, por ejemplo, el conjunto de subclases {SECRETARIO, INGENIERO, TÉCNICO} es una especialización de la superclase EMPLEADO que distingue entre las entidades de los empleados en función del tipo de trabajo de cada entidad empleado.

Para la generalización podemos pensar en un proceso inverso de abstracción en el que suprimimos las diferencias entre varios tipos de entidades, identificamos sus características comunes y las generalizamos en una sola superclase de la cual los tipos de entidades originales son subclases especiales, por ejemplo, consideremos los tipos de entidad CAR y TRUCK, ya que tienen varios atributos comunes, se pueden generalizar en el tipo de entidad VEHÍCULO.

Existen las restricciones de disjunción que especifica que las subclases de la especialización deben ser disjuntas, e integridad que puede ser total o parcial.   
Una restricción de especialización total especifica que cada entidad en la superclase debe ser miembro de al menos una subclase en la especialización y una especialización parcial, permite a una entidad no pertenecer a ninguna de las subclases.

Ambas (disjunción e integridad) son independientes, por lo tanto, tenemos las siguientes cuatro restricciones posibles en la especialización:

* Disjuntos, total
* Disjunto, parcial
* Superposición, total
* Superposición, parcial

Por supuesto, la restricción correcta se determina a partir del significado del mundo real que se aplica a cada especialización. En general, una superclase que se identificó a través del proceso de generalización generalmente es total, porque la superclase se deriva de las subclases y, por lo tanto, contiene solo las entidades que están en las subclases.” (Elmasri&Navathe, 2011, p.248-254).

Pienso que esta es una manera de organizar los datos por prioridad de manera que todos los datos lleven diferente prioridad.

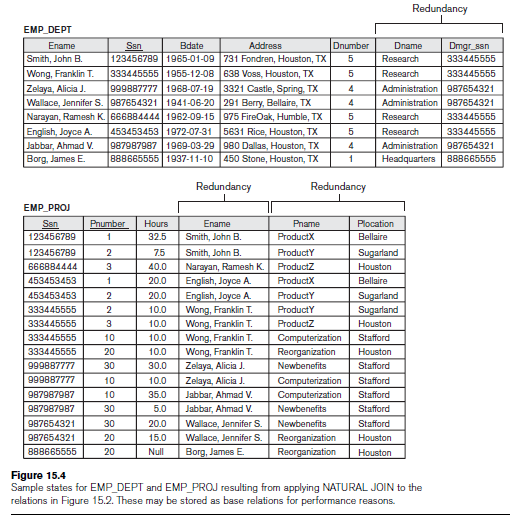
14. Describa las anomalías de inserción, borrado y actualización, ilustre con ejemplos.

**“*Anomalías de inserción:*** Las anomalías de inserción se pueden diferenciar en dos tipos, ilustrados por los siguientes ejemplos basados ​​en la relación EMP\_DEPT:

* Para insertar una nueva tupla de empleado en EMP\_DEPT, debemos incluir los valores de atributo para el departamento para el que trabaja el empleado, o valores NULL (si el empleado aún no trabaja para un departamento). Por ejemplo, para insertar una nueva tupla para un empleado que trabaja en el departamento número 5, debemos ingresar todos los valores de atributo del departamento 5 correctamente para que sean consistentes con los valores correspondientes para el departamento 5 en otras tuplas en EMP\_DEPT. En el diseño de la Figura 15.2, no tenemos que preocuparnos por este problema de coherencia porque ingresamos solo el número de departamento en la tupla de empleados; todos los demás valores de atributo del departamento 5 se registran solo una vez en la base de datos, como una sola tupla en la relación DEPARTAMENTO.
* Es difícil insertar un departamento nuevo que todavía no tenga empleados en la relación EMP\_DEPT. La única forma de hacerlo es colocar valores NULL en los atributos para el empleado. Esto viola la integridad de la entidad para EMP\_DEPT porque Ssn es su clave principal. Además, cuando se asigna el primer empleado a ese departamento, ya no necesitamos esta tupla con valores NULL. Este problema no ocurre en el diseño de la figura 15.2 porque un departamento se ingresa en la relación de DEPARTAMENTO ya sea que haya empleados o no trabaje para ello, y cada vez que se asigna un empleado a ese departamento, se inserta una tupla correspondiente en EMPLOYEE.

**Anomalías de eliminación:** El problema de las anomalías de eliminación se relaciona con la segunda situación de anomalía de inserción recién discutida. Si eliminamos de EMP\_DEPT una tupla de empleados que representa al último empleado que trabaja para un departamento en particular, la información relacionada con ese departamento se pierde de la base de datos. Este problema no ocurre en la base de datos de la figura 15.2 porque las tuplas de DEPARTAMENTO se almacenan por separado.

**Anomalías de modificación:** En EMP\_DEPT, si cambiamos el valor de uno de los atributos de un departamento en particular, por ejemplo, el gerente de departamento, debemos actualizar las tuplas de todos los empleados que trabajan en ese departamento; de lo contrario, la base de datos se volverá inconsistente. Si no actualizamos algunas tuplas, se mostrará que el mismo departamento tiene dos valores diferentes para el administrador en diferentestuplas de empleados, lo que sería incorrecto.”(Elmasri&Navathe, 2011, p.507-509).



Creo que estas anomalías son importantes corregirlas ya que hay datos que no se pueden quedar así solamente, y para que la ejecución de SQL y en la visualización de la tabla sea correcta.

15. ¿Qué es una dependencia funcional?

“Una dependencia funcional es una restricción entre dos conjuntos de atributos de la base de datos Supongamos que nuestro esquema de base de datos relacional tiene n atributos A1, A2, ..., An; pensemos que toda la base de datos está descrita por un esquema de relación única universal R = {A1, A2, ..., An}. No implicamos que realmente almacenaremos la base de datos como una tabla universal única; utilizamos este concepto solo en el desarrollo de la teoría formal de las dependencias de datos.

Una dependencia funcional, denotada por X → Y, entre dos conjuntos de atributos X e Y que son subconjuntos de R especifican una restricción sobre las posibles tuplas que pueden formar un estado de relación r de R. La restricción es eso, para cualesquiera dos tuplas t1 y t2 en r que tengan t1 [X] = t2 [X], también deben tener t1 [Y] = t2 [Y].

Esto significa que los valores del componente Y de una tupla en r dependen de, o están determinados por, los valores del componente X; alternativamente, los valores del componente X de una tupla determinan de manera única (o funcional) los valores del componente Y.

También decimos que existe una dependencia funcional de X a Y, o que Y depende funcionalmente de X. La abreviatura de dependencia funcional es FD o f.d. El conjunto de atributos X se llama el lado izquierdo del FD, y Y se llama el lado derecho” (Elmasri&Navathe, 2011, p.513-514)

Una dependencia funcional es una conexión entre uno o más atributos X → Y, donde los valores de Y dependen de X, al igual que los valores de X determinan de manera única los valores de Y.

16. ¿A qué se refiere el término una relación no normalizada?

“Una relación no normalizada en la que los esquemas de relación dados basados ​​en sus FD y claves primarias tienen redundancia, anomalías de inserción, eliminación y de modificación.” (Elmasri&Navathe, 2011, p.517).

Es importante comprender este concepto porque todas las tablas con información son redundantes y muestran casi siempre la misma información o repetida varias veces cuando solo podría verse una sola vez.

17.- Defina la primera, segunda y tercera forma normal

1FN: La relación no debe tener multivalor atributos o relaciones anidadas. Requiere formar nuevas relaciones para cada multivalor atributo o relación anidada.

2FN: Para relaciones donde la clave primaria contiene atributos múltiples, el atributo no llave debe ser funcionalmente depende de una parte de la clave principal. El proceso es descomponer y establecer una nueva relación para cada tecla parcial con su atributo dependiente (s). Es importante mantener una relación con la clave primaria original y cualquier atributo que sea completamente funcionalmente dependiente de él.

3FN: La relación no debe tener una no llave como atributo funcionalmente determinado por otro atributo no clave (o por un conjunto de atributos no clave). Es decir, no debe haber dependencia transitiva de una no llave como atributo en la clave principal. Es necesario descomponer y establecer una relación que incluya a los atributos no clave que funcionalmente determinar (n) otro atributo no clave (s). (Elmasri&Navathe, 2011, p.525).

Estas formas normales son las de toda la vida y se aplican casi siempre en la normalización de datos porque al separar en varias tablas las llaves y descomponer cada atributo para que al final quede de una manera que no tenga muchos datos y tenga las referencias a cada una de las demás tablas generadas por la descomposición.

18.- Defina la forma normal de Boyce-Codd.

Un esquema de relación R está en BCNF si siempre que sea una dependencia funcional no trivial X → A contenida en R, luego X es una super llave de R. (Elmasri&Navathe, 2011, p.529).

Está definición nos dice que esta forma normal se utiliza para evitar aún más la redundancia.

19.- ¿Qué es una dependencia multivaluada?

Una dependencia multivaluada X →→ Y especificada en el esquema de relación R, donde X e Y son subconjuntos de R, especifica la siguiente restricción en cualquier estado de relación r de R: si dos tuplas t1 y t2 existen en r tal que t1[X]=t2[X], entonces dos tuplas t3 y t4 también deberían existir en r con las siguientes propiedades, donde usamos Z para denotar (R - (X∪Y)):

t3[X]=t4[X]=t1[X]=t2[X].

t3[Y]=t1[Y] y t4[Y]=t2[Y].

t3[Z]=t2[Z] y t4[Z]=t1[Z].

Siempre que X →→ Y se cumple, decimos que X multidetermina Y. Debido a la simetría en la definición, cada vez que X →→ Y se mantiene en R, también lo hace X→→Z. Por lo tanto, X→→Y implica X →→ Z, y por lo tanto a veces se escribe como X →→ Y | Z. (Elmasri&Navathe, 2011, p.533).

Quiere decir que si en una relación existen 2 tuplas que sean iguales, entonces existen dos tuplas más que sean iguales dependiendo de las combinaciones anteriores, para desúes aplicar la teoría de dependencias funcionales como la transitiva etc para ver qué tupla determina a otra.

20.- Defina la cuarta forma normal, ¿cuándo es violada esta forma y cuándo es típicamente aplicable?

R está en 4NF con respecto a un conjunto de dependencias F (que incluye dependencias funcionales y dependencias de valores múltiples) si, para cada dependencia trivalente no trivial X →→ Y en F, X es una super llave para R. (Elmasri&Navathe, 2011, p.534).

Tengo entendido que ésta forma normal se utiliza para eliminar la redundancia todavía más aunque generan muchas tablas por descomposición de datos.

21.- Defina la dependencia reunión y la quinta forma normal.

Una dependencia de unión (JD), denotada por JD (R1, R2, ..., Rn), especificada en esquema de relación R, especifica una restricción en los estados r de R. La restricción establece que cada estado legal r de R debe tener una descomposición de unión no aditiva en R1, R2, ..., Rn, a través de algunas operaciones del álgebra relacional. Tal dependencia se llama trivial porque tiene la propiedad de unión no aditiva para cualquier relación estado r de R y por lo tanto no especifica ninguna restricción en R.

Un esquema de relación R está en la quinta forma normal (5NF) (o proyecto-unión forma normal (PJNF) con respecto a un conjunto F de funcional, multivalor, y unir dependencias si, por cada dependencia de unión no trivial JD (R1, R2, ..., Rn) en F + (es decir, implícito en F), cada Ri es una super llave de R. (Elmasri&Navathe, 2011, p.535).

Un esquema relacional se presentan las restricciones de los estados de los 2 conceptos de la vida real que se estén relacionando, y la quinta forma normal se utiliza al igual que la cuarta forma normal para reducir aún más la redundancia en las tablas, pero se usa poco por la gran cantidad de tablas que se generan.

22.- ¿Cuál es el significado de la propiedad de cerradura de un conjunto de dependencias funcionales, ilustre con un ejemplo?

"Definición. Formalmente, el conjunto de todas las dependencias que incluyen F y todas las dependencias que se pueden inferir de F se llaman cierre de F; se denota por F +.

Por ejemplo, supongamos que especificamos el siguiente conjunto F de funcionalidad obvia dependencias en el esquema de relación en la figura 15.3 (a):

F = {Ssn → {Ename, Bdate, Address, Dnumber}, Dnumber → {Dname, Dmgr\_ssn}}

Algunas de las dependencias funcionales adicionales que podemos inferir de F son las siguientes:

Ssn → {Dname, Dmgr\_ssn}

Ssn → Ssn

Dnúmero → Dname

Un FD X → Y se deduce de un conjunto de dependencias F especificado en R si X → Y mantiene en cada relación legal el estado r de R; es decir, siempre que r satisfaga todas las dependencias en F, X → Y también se cumple en r. El cierre F + de F es el conjunto de todos los funcionales dependencias que se pueden inferir de F. Determinar una forma sistemática de inferir dependencias, debemos descubrir un conjunto de reglas de inferencia que se pueden usar para inferir nuevas dependencias de un conjunto dado de dependencias. Consideramos algunos de estos las siguientes reglas de inferencia. Usamos la notación F | = X → Y para indicar que la función la dependencia X → Y se deduce del conjunto de dependencias funcionales F. "

(Elmasri & Navathe, 2011, pp.545-546)

Los conjuntos de dependencias funcionales se reconocen por poner entre paréntesis los atributos involucrados y subrayando las llaves de estas dependencias.

23.- ¿Que es un conjunto mínimo de dependencias funcionales?

"De manera informal, una cobertura mínima de un conjunto de dependencias funcionales E es un conjunto de funcionalidades dependencias F que satisface la propiedad de que cada dependencia en E está en el cierre F + de F. "

"Definición. Una cobertura mínima de un conjunto de dependencias funcionales E es mínima conjunto de dependencias (en la forma canónica estándar y sin redundancia) eso es equivalente a E. "(Elmasri& Navathe, 2011, pp. 549-550)

Cada dependencia tendrá que cumplirse para que esté en el cierre y sean llaves candidatas y no deben de contener redundancia para que sean equivalentes a un conjunto de dependencias.

24.- ¿Que se entiende la condición de preservación de atributos en una descomposición?

"Sería útil si cada dependencia funcional X → Y se especifica en F apareció directamente en uno de los esquemas de relación Ri en la descomposición D o podría inferirse de las dependencias que aparecen en algunos Ri.

Informalmente, esta es la condición de preservación de dependencia. Queremos preservar las dependencias porque cada dependencia en F representa una restricción en la base de datos".

(Elmasri& Navathe, 2011, pp. 552-553)

Cada restricción es importante ya que de no cumplirlas se puede deshacer toda la base y al momento de descomponer las tuplas la condición ya no existe y los datos se perderían.

25.- ¿Por qué las formas normales por si solas son insuficientes como una condición para un buen diseño de un esquema de bases de datos?

"Los formularios normales por sí solos no son suficientes para que el mejor diseño de la base de datos sea la combinación de las formas normales, al menos, hasta 3-NF o BCNF.

"Otro objetivo es tener cada relación individual Ri en la descomposición D estar en

BCNF o 3NF. Sin embargo, esta condición no es suficiente para garantizar una buena base de datos diseño en sí mismo. Debemos considerar la descomposición de la relación universal como un todo, además de mirar las relaciones individuales". (Elmasri& Navathe, 2011, pp. 552).

Esto tiene sentido ya que cada forma normal va haciendo que cada uno de los atributos sean separados en tablas diferentes para evitar redundancia, pero no garantiza que sea una buena base de datos ya que podría ser excesiva cantidad de tablas generadas para que cumpla una condición.

26.- ¿Que es la propiedad de preservación de dependencia para una descomposición, porqué esta es importante?

"Sería útil si cada dependencia funcional X → Y se especifica en F apareció directamente en uno de los esquemas de relación Ri en la descomposición D o podría inferirse de las dependencias que aparecen en algunos Ri. Informalmente, esta es la condición de preservación de dependencia. Queremos preservar las dependencias porque cada dependencia en F representa una restricción en la base de datos". (Elmasri& Navathe, 2011, pp. 552-553).

A mi parecer es importante porque hablando de normalización si una propiedad no se cumple, al momento de descomponer la base se puede encontrar con ciertas restricciones que no nos dejarían cumplir con la dependencia original.

27.- ¿Cuál es la propiedad de reunión sin pérdida de una descomposición?

“Otra propiedad que debe tener una descomposición D es la unión no aditiva propiedad, que asegura que no se generan tuplas espurias cuando un NATURAL JOIN se aplica a las relaciones resultantes de la descomposición.  
Definición. Formalmente, una descomposición D = {R1, R2,..., Rm} de R tiene la pérdida (no aditiva) unir propiedad con respecto al conjunto de dependencias F en R si, para cada estado de relación r de R que satisface F, se cumple lo siguiente, donde \* es el UNIÓN NATURAL de todas las relaciones en D: \* (πR1 (r),..., πRm (r)) = r.”(Elmarsi & Navathe, 2011, pp553-554).

Pérdida se refiere a la pérdida de información y no a la pérdida de tuplas. Si no se tiene la propiedad de unión si pérdida, y después de aplicar la proyección (π) y Natural Join (\*) esas tuplas adicionales representan información errónea.

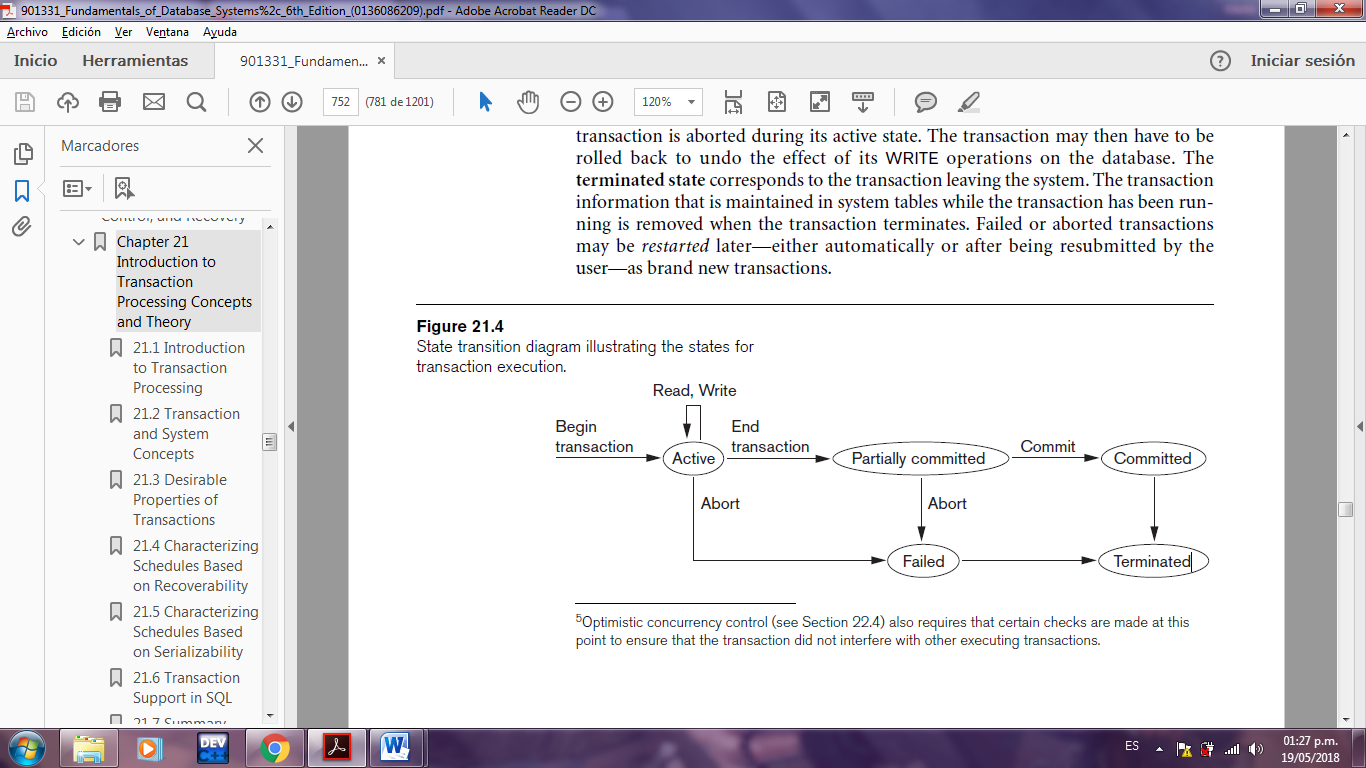
28.- ¿Qué significa el término optimización heurística? Describa las principales heurísticas que se aplican durante la optimización de una consulta.

Navathe define una optimización heurística como: “[...] Una representación de consulta optimizada, que corresponde a la estrategia de ejecución de consulta.

Una de las principales reglas heurísticas es aplicar operaciones SELECT y PROJECT antes aplicando JOIN u otras operaciones binarias, porque el tamaño del archivo resultante de una operación binaria, como JOIN, suele ser una función multiplicativa del tamaño de los archivos de entrada. Las operaciones SELECT y PROJECT reducen el tamaño de un archivo y, por lo tanto, debe aplicarse antes de una combinación u otra operación binaria.”(Elmarsi & Navathe, 2011, pp700-701).

Siento que éste tipo de optimización es muy eficiente ya que al final de cada JOIN el archivo generado puede ser muy grande y usando la heurística reducen dicho tamaño y es más optimizado ya que al momento de buscar un dato, se encontrará de manera más rápida porque la información es menor.

29.- ¿Elabore un diagrama de estados que describa los estados típicos de las transacciones durante su ejecución?



“Diagrama de transición del estado que ilustra los estados para ejecución de transacción.

■ BEGIN\_TRANSACTION. Esto marca el comienzo de la ejecución de la transacción.

■ READ OR WRITE. Estos especifican operaciones de lectura o escritura en la base de datos elementos que se ejecutan como parte de una transacción.

■ END\_TRANSACTION. Esto especifica que las operaciones de transacción READ AND WRITE han terminado y marca el final de la ejecución de la transacción. Sin embargo, en este punto, puede ser necesario verificar si los cambios introducidos por la transacción se puede aplicar permanentemente a la base de datos (comprometida) o si la transacción debe abortarse porque viola la serializabilidad o por alguna otra razón.

■ COMMIT\_TRANSACTION. Esto señala un final exitoso de la transacción por lo que cualquier cambio (actualización) ejecutado por la transacción puede ser seguro comprometido con la base de datos y no se deshará.

■ ROLLBACK (o ABORT). Esto indica que la transacción ha finalizado sin éxito, para que cualquier cambio o efecto que la transacción pueda haber aplicado a la base de datos debe deshacerse. “(Elmarsi & Navathe, 2011, pp751-752).

En mi opinión un diagrama muy eficiente ya que es muy cómodo para el entendimiento de lo que ocurre en una transacción sin que quede duda.

*30.- ¿Qué es un sistema de bitácora de uso? ¿Cuáles son los registros típicos en el sistema de bitácora? ¿Qué son los puntos de confirmación de transacciones y porqué son importantes?*

“El registro es un archivo secuencial, de solo un agregado, que se guarda en el disco, por lo que no se ve afectado por ningún tipo de falla a excepción del disco o falla catastrófica.

Típicamente, uno (o más) búferes de memoria principal contienen la última parte del archivo de registro, por lo que el registro las entradas se agregan primero al búfer de memoria principal. […]Además, el archivo de registro del disco se respalda periódicamente para archivar almacenamiento (cinta) para proteger contra fallas catastróficas.

Los puntos de confirmación de transacciones solo se utilizan para identificar cada transacción.

Los protocolos para la recuperación que evitan retrocesos en cascada que incluyen casi todos los protocolos prácticos, no requiere que las operaciones de LECTURA estén escritas al registro del sistema. Sin embargo, si el registro también se usa para otros fines, como auditar (realizar un seguimiento de todas las operaciones de la base de datos), entonces tales entradas pueden ser incluidas. “(Elmarsi & Navathe, 2011, p753).

Es importante saber que en cómo en cualquier procedimiento, se debe llevar un registro de lo que se está haciendo, como en este caso las transacciones ya en que este registro, nos daremos cuenta de qué está ocurriendo en el proceso y si hay errores cómo proseguir para corregirlos.

*31.- Describa las propiedades ACID de una transacción*.

“[..] Deben ser aplicados por el control de simultaneidad y los métodos de recuperación de la DBMS.

■ Atomicidad. Una transacción es una unidad atómica de procesamiento; debe ser cualquiera realizado en su totalidad o no realizado en absoluto.

■ Conservación de consistencia. Una transacción debe preservar la coherencia, lo que significa que si se ejecuta completamente de principio a fin sin interferencia de otras transacciones, debe tomar la base de datos de un estado consistente a otro.

■ Aislamiento. Una transacción debería aparecer como si se estuviera ejecutando de forma aislada de otras transacciones, a pesar de que muchas transacciones se están ejecutando de forma aislada de otras transacciones, aunque muchas transacciones se estén ejecutando simultáneamente. Es decir, la ejecución de una transacción no debe interferirse con cualquier otra transacción que se ejecute simultáneamente.

■ Durabilidad o permanencia. Los cambios aplicados a la base de datos por un cometido la transacción deben persistir en la base de datos. Estos cambios no deben ser perdidos debido a cualquier falla. “(Elmarsi & Navathe, 2011, pp754-755).

Siento que es muy importante conocer estas propiedades ya que al ser aplicadas en la realidad podrían ayudar a saber qué hacer en cada caso y como resolver un problema aplicando todos los conocimientos adquiridos acerca de este tema.

**BIBLIOGRAFÍA:** Elmasri, Ramez & Navathe, Shamktant (2011).*Fundamentals of Database Systems*, EEUU.Addison-Wesley.