Resumen – Final

Normalización:

1era NF: Una relación está en Primera Forma Normal si:

* Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son simples e indivisibles.
* La relación contiene una clave primaria única.
* La clave primaria no contiene atributos nulos.
* No debe existir variación en el número de columnas.
* Los Campos no clave deben identificarse por la clave (Dependencia Funcional)
* Debe Existir una independencia del orden tanto de las filas como de las columnas, es decir, si los datos cambian de orden no deben cambiar sus significados

Esta forma normal elimina los valores repetidos dentro de una Base de Datos.

2da NF: Una relación está en 2FN si está en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal. Es decir que no existen dependencias parciales. (Todos los atributos que no son clave principal deben depender únicamente de la clave principal).

3era NF: Una relación esta en 3NF si esta en 2NF. La tercera forma normal comprueba las dependencias transitivas, eliminando atributos que no dependen de la clave principal. Para pasar a la tercera forma normal se deben eliminar las DFs que no dependen de la clave principal.

BCNF: En una tabla en 3FN, todos los atributos dependen de una clave, de la clave completa y de ninguna otra cosa excepto de la clave. Se dice que una tabla está en BCNF si y solo si está en 3FN y cada dependencia funcional no trivial tiene una clave candidata como determinante. En términos menos formales, una tabla está en BCNF si está en 3FN y los únicos determinantes son claves candidatas. Esto quiere decir que:

* Si no existen claves candidatas compuestas (con varios atributos), está en BCNF.
* Si existen varias claves candidatas compuestas y éstas tienen un elemento común, puede no estar en BCNF. Sólo si, para cada dependencia funcional en la relación, el determinante es una clave candidata, estará en BCNF.

4ta NF:

5ta NF:

Teoría:

1. **Defina el concepto de dependencia funcional. ¿Qué son las reglas de inferencia? ¿Para que sirven? Enumerarlas**

Una dependencia funcional es una conexión entre uno o más atributos.

**Definición:** Sean X e Y subconjuntos de atributos de una relación. Se dice que Y tiene una dependencia funcional de X, o que X determina a Y, si cada valor de X tiene asociado siempre un único valor de Y.

* Dependencia funcional completa: En una dependencia funcional X -> Y, cuando X es un conjunto de atributos, se dice que la dependencia funcional es completa , si sólo depende de X, y no de ningún subconjunto de X.
* Dependencia funcional elemental: Si se tiene una dependencia completa X => Y, se dice que es una dependencia funcional elemental si Y es un atributo, y no un conjunto de ellos.
* Dependencia funcional trivial: Una dependencia funcional A -> B es trivial cuando B es parte de A. Esto sucede cuando A es un conjunto de atributos, y B es a su vez un subconjunto de A.

Las reglas de inferencia permiten establecer relaciones sintácticas entre las diferentes dependencias funcionales (subconjuntos de atributos de una relación) permitiendo llega, a partir de las mismas, a nuevas dependencias funcionales que son validas ya que estas se infieren de las anteriores.

* 1. Regla reflexiva:

Si Y ⊆ X entonces X →Y

Ej.:

{A, B} → {A}

{A, B} → {B}

* 1. Regla de autodeterminación:

Si X ⊆X entonces X →X

Ej.:

{A} → {A}

{B} → {B}

* 1. Regla de ampliación:

Si X →Y entonces X ∪ Z → Y ∪ Z => X ∪ Z → Y

Ej.:

{A} → {B} ⇒ {A, S} → {B, S}

{A, S} → {B}

* 1. Regla de descomposición:

Si X → Y ∪ Z entonces X → Y y X → Z

Ej.:

{D,B} → {N,T} ⇒ {D,B} → {N}

{D,B} → {T}

* 1. Regla de unión:

Si X → Y , X → Z entonces X → Y ∪ Z

Ej.:

{D,B} → {N}

{D,B} → {T} ⇒ {D,B} → {N,T}

* 1. Regla transitiva (dependencia transitiva) :

Si X →Y e Y →Z entonces X→Z

Ej.:

{A} → {B}

{B} → {C} ⇒ {A} → {C}

Ej.:

{A,B} → {H}

{H} → {S,T} ⇒ {A,B} → {S,T}

* 1. Regla de composición:

Si X →Y , Z →H entonces X U Z → Y U H

Ej.:

{D} → {N}, {B} → {T} ⇒ {D,B} → {N,T}

* 1. Regla pseudotransitiva:

Si X →Y, para W U Y → Z entonces W U X → Z

Ej.:

{A} →{B}, {C,B} → {H} ⇒ {C,A} → {H}

1. **¿Qué propiedades posee el algebra relacional? ¿Cuáles son los operadores tradicionales? Defina la operación división.**

Las relaciones poseen ciertas propiedades, todas ellas consecuencias inmediatas de la definición de relación. Las propiedades dentro de cualquier relación dada son:

* 1. **Propiedad de cierre:** Esta garantiza que cualquier operación que se realice a una o más relaciones mediante los operadores del álgebra relacional, devolverá una nueva relación con todas sus propiedades. En cuanto a la cabecera de las relaciones, la propiedad de cierre garantiza que tanto los atributos de la cabecera como el dominio de cada uno de ellos mantendrán sus características.

**Operadores tradicionales del algebra relacional:** Los operadores tradicionales son:

* Unión (UNION)
* Intersección (INTERSECT)
* Diferencia (MINUS)
* Producto cartesiano (TIMES)

**La operación de división se defina como:** Una operación que retorna una relación agrupada, en cierta manera conceptual, en conjuntos de tuplas. Formando un conjunto para cada valor distinto del o los atributos que definan el agrupamiento. Y agrega a cada conjunto formado, al realizar alguna operación con operadores totales, el atributo conteniendo en el resultado de la operación especificada.

* Operación unaria.
* Dominio de la operación:
  + Definido por los dominios del criterio de agrupamiento y los totalizadores agregados.

Sintaxis:

SUMMARIZE relacion BY atributo ADD totalizador(atributo) AS nuevo-atributo

Ej:

SUMMARIZE relacion BY atributo1 ADD COUNT(atributo1) AS cantidad

1. **¿Puede una relacion que se encuentra en BCNF, poseer anomalías de actualización como consecuencia de una DF? Justifique.**
2. **¿Cuáles son las propiedades de las relaciones?**
   1. **No existen tupias duplicadas:** Esta propiedad surge del hecho de que el cuerpo de la relación es un conjunto matemático (de tuplas) y en consecuencia no incluyen elementos duplicados. Esta primera propiedad sirve para ilustrar el punto de que una relación y una tabla no son lo mismo, ya que una tabla puede contener filas duplicadas, mientras que una relación no puede contener una tupla duplicada. (Ya que si una "relación" contiene tupias duplicadas, entonces, por definición, ¡no es una relación!)
   2. **Las tupias están en desorden, de arriba hacia abajo:** Esta propiedad también surge del hecho de que el cuerpo de la relación es un conjunto matemático y por definición, los conjuntos no están ordenados. Esta segunda propiedad sirve también para ilustrar la idea de que una relación y una tabla no son lo mismo, ya que las filas de una tabla tienen un ordenamiento (de arriba hacia abajo o viceversa), mientras que las tupias de una relación no lo tienen.
   3. **Los atributos están en desorden, de izquierda a derecha:** Esta propiedad surge del hecho de que el encabezado de una relación también es un conjunto (de atributos). Por lo tanto, no existe algo como "el primer atributo" o "el segundo atributo" (etcétera) y tampoco hay "el siguiente atributo" (de nuevo, no existe el concepto de "el siguiente"), en otras palabras, siempre se hace referencia a los atributos por nombre, nunca por posición.
   4. **Cada tupla contiene exactamente un valor para cada atributo:** Esta propiedad surge inmediatamente de la definición de una tupia: una tupia es un conjunto de n componentes o pares ordenados. Se dice que una relación que satisface esta cuarta propiedad está normalizada o, lo que es equivalente, está en la primera forma normal. Esta propiedad en particular podría parecer muy obvia y de hecho lo es, en especial debido a que todas las relaciones están normalizadas de acuerdo con la definición.

**SQL:**

1. **La siguiente sentencia proyectaria los articulos mas caros. Como modificaria la misma si no pudiese utlizar funciones de totales?**

select NroArticulo, Descripcion from articulos where precio = (select Max(precio) from articulos)

**Respuesta:**

select articulos.NroArticulo, articulos.Descripcion from articulos where precio =(select top 1 articulos.precio from articulos group by precio order by precio desc)

1. **La siguiente sentencia proyectaria las facturas que superen los $100 del total. Indicar si esta correctamente formulada, si no corregirla.**

select nroFactura, sum(precio\*cantidad) as total from itemFactura where total>100 group by NroFactura.

**Respuesta:**

La sentencia es incorrecta porque no se puede referenciar a un campo que es un alias en una clausula where. La sentencia correcta seria:

select \* FROM (select nroFactura, sum(precio\*cantidad) as total from itemFactura group by nroFactura) as facturaTotal where total>100

1. **¿Que proyecta la siguiente consulta? Si no es correcta explique porque**

Select nombre from clientes where nombre like ‘\_[AOE]%[ S]’

**Respuesta:**

Se proyectan los nombres de los clientes que empiezen con cualquier cosa y luego tengan una A, O, E como segundo caractres y luego deben finalizar con un espacio o con la lestra S.

**NOTA:**

Las diferentes wilcard del like son:

* El % - Representa 0, 1 o multiples caracteres.
* El \_ - Representa un unico carácter
* Los [] – Todo lo que este dentro de los corchetes será buscado explícitamente y equivale a un caracter. Tambien se pueden poner rangos, Ej.:
  + [AEO5] – Un carácter que puede ser A,E,O,5.
  + [A-C] – El rango de letras entre A y C que es A,B,C.
* El ! – Representa una negación de algo. Es lo contrario a lo que se pone. Tambien se pueden usar rangos, ej.:
  + [!AEO5] – Puede ser cualquier carácter que no sea A,E,O,5.
  + [!A-C] – Son las letras que no estran dentro del rango entre A y C o sea, todas menos A,B,C.

1. **Indicar si es correcta o no la siguiente consulta. Teniendo en cuenta que el idCliente es PK en clientes y clave foranea en facturas. Indicar que proyectaria.**

select nroFactura, fecha, id\_cliente as codigo, clientes.nombre from facturas f, (select \* from clientes where….

**Respuesta:**

Esta sentencia es valida y va a devolver un producto cartesiano entre los nombres de los clientes y la tabla factura o nulo dependiendo de la clausula where.