**Práctica 1 – Apdos-1-2-3: Corregir y Mejorar el diseño de BDs con Dependencias Funcionales (DFs), Dependencias Multivaloradas (DMs) y Formas Normales (FNs)**

### **APARTADO 1.- Reconoce DFs, FNs y consigue un BD en 3FN**

- Conserva el nombre que tienen los atributos.

Dadas estas dos relaciones que forman una BD: donde ***re*** es reserva, ***cl*** es cliente y ***ae*** es aeropuerto

***re(dni\_cl\_re, cod\_vuelo\_cl\_re, nomb\_cl\_re, apell\_cl\_re, telefono\_cl\_re,cuenta\_cl\_re, fecha\_cl\_re, clase\_cl\_re, nume\_plaza\_cl\_re, confirmacion\_cl\_re, cod\_ciudad\_cl\_re, nombre\_ciudad\_cl\_re)***

***vuelo(cod\_vuelo, cod\_avion, plazas\_vuelo\_avion, nomb\_ae\_origen, dire\_ae\_origen, origen\_cod\_ae\_vuelo, destino\_cod\_ae\_vuelo, cod\_pais\_ae\_origen, cod\_ciudad\_ae\_origen, nombre\_ciudad\_ae\_origen, direcciónPostal\_ae\_vuelo\_origen, modelo\_avion, cod\_aerolínea)***

-> Puede que necesites repetir algunos atributos en alguna tabla.

-> Ten en cuenta estos requisitos: El mismo nombre y código de una ciudad puede existir en dos países. En el vuelo debes tener la aerolínea. El código de vuelo es único, e independiente de la línea.

-> Si necesitas asumir otros requisitos (atributos nuevos, DFs nuevas) cuando veas más de una interpretación: escoge la mejor y describe textualmente lo que asumes(los supuestos) en este punto del documento (aquí) a continuación (como los requisitos anteriores).

**Se pide hacer lo siguiente.** Este apartado tiene dos fases:

1.- Encuentra la lista de DFs que hay en las dos relaciones,

-> ver al final de este doc: APTDO 1: AYUDAS par Encontrar DFs y obtener tablas en 3ªFN

DFs para la reserva cliente :

* **dni\_cl\_re** -> nomb\_cl\_re, apell\_cl\_re, telefono\_cl\_re, cuenta\_cl\_re  
  Requisitos: Asumimos que para un dni hay solo un nombre, un apellido, un teléfono y una cuenta cliente.
* **cod\_vuelo\_cl\_re**-> dni\_cl\_re, confirmacion\_cl\_re, fecha\_cl\_re**,** clase\_cl\_re, nume\_plaza\_cl\_re, confirmacion\_cl\_re, cod\_ciudad\_cl\_re, nombre\_ciudad\_cl\_re  
  Requisitos: Asumimos que para una reserva de un cliente preciso, identificada por el código de vuelo, hay una única confirmación de la reserva a un único cliente identificado por su dni, una sola fecha de reserva, una única ciudad (identificada por su nombre y su código) en un único avión identificado por su código, en una única clase y una única plaza en el avión.

DFs para el vuelo :

* **cod\_vuelo** -> cod\_avion, plazas\_vuelo\_avion, origen\_cod\_ae\_vuelo, destino\_cod\_ae\_vuelo, modelo\_avion, cod\_aerolínea  
  Requisitos: Asumimos que un vuelo identificado por su código corresponde a un único avión que hace solo un viaje, de un único punto de origen a un único destino, sin conexiones.
* **origen\_cod\_ae\_vuelo** -> nomb\_ae\_origen, dire\_ae\_origen, cod\_pais\_ae\_origen, cod\_ciudad\_ae\_origen, nombre\_ciudad\_ae\_origen, direcciónPostal\_ae\_vuelo\_origen  
  Requisitos: Asumimos que para un código de aeropuerto hay solo un aeropuerto en el mundo.  
  Esta dependencia existe tanto para el aeropuerto de origen como el aeropuerto de destino, así definimos nuevos atributos (sin \_origen, para aplicarlo a los dos casos).
* **cod\_avion** -> plazas\_vuelo\_avion, modelo\_avion  
  Requisitos: Asumimos que un avión identificado por su código corresponda a un único modelo de avión con su número de plazas determinado.

2.- Define las tablas necesarias para que toda la BD esté en 3FN (incluyendo las PKs y las FKs)

Tabla cliente:

| **dni\_cl\_re** | nomb\_cl\_re | apell\_cl\_re | telefono\_cl\_re | cuenta\_cl\_re |
| --- | --- | --- | --- | --- |

Tabla reserva:

| **cod\_vuelo\_cl\_re** | dni\_cl\_re | fecha\_cl\_re | clase\_cl\_re | nume\_plaza\_cl\_re | confirmacion\_cl\_re | cod\_ciudad\_cl\_re | nombre\_ciudad\_cl\_re |
| --- | --- | --- | --- | --- | --- | --- | --- |

Tabla vuelo:

| **cod\_vuelo** | cod\_avion | origen\_cod\_ae\_vuelo | destino\_cod\_ae\_vuelo | plazas\_vuelo\_avión | cod\_aerolínea |
| --- | --- | --- | --- | --- | --- |

Tabla aeropuerto:

| **cod\_ae** | nomb\_ae | dire\_ae | cod\_pais\_ae | cod\_ciudad\_ae | nombre\_ciudad\_ae | direcciónPostal\_ae |
| --- | --- | --- | --- | --- | --- | --- |

Tabla avión:

| **cod\_avion** | modelo\_avion | plazas\_vuelo\_avión |
| --- | --- | --- |

Claves primarias (PKs):

* cliente: dni\_cl\_re
* reserva: cod\_vuelo\_cl\_re
* vuelo: cod\_vuelo
* aeropuerto de origen: origen\_cod\_ae\_vuelo
* avion: cod\_avion

Claves foráneas (FKs):

* reserva: dni\_cl\_re (referenciando la tabla Cliente), cod\_vuelo\_cl\_re (referenciando a cod\_vuelo en la tabla Vuelo)
* vuelo: cod\_avion (referenciando la tabla Avión), origen\_cod\_ae\_vuelo y destino\_cod\_ae\_vuelo (referenciando a cod\_ae en la tabla Aeropuerto)

Estas tablas se relacionan de la siguiente manera:

1. Un cliente puede tener varias reservaciones.
2. Un vuelo puede tener varias reservaciones.
3. Un avión puede realizar varios vuelos.
4. Un aeropuerto puede ser origen o destino de varios vuelos.

### **APARTADO 2.- Reconocer la clave primaria (CP), clave ajena (CA), claves candidatas (CC), dependencia funcional (DF) y dependencia multivalorada (DM), datos erróneos. Escribir procedimientos sencillos en PLSQL.**

*-> Utilizando la BDejemploSimple.sql tal y como está definida, sin CP ni CA, realiza los siguientes apartados: (en algunos apartados, debes decidir tu cuáles son las CP para escoger la respuesta)*

**2.0**- *Crea la BDejemplo en Oracle, ejecutando el script* ***BDejemploSimple.sql***tal y como está.

Revisa los comentarios dentro del script para entender el significado de los atributos.

**2.1**- Describe lo indicado en cada apartado: (SIEMPRE contesta cada pregunta por separado)

2.1.x - - 1.- Qué resultado da la siguiente consulta?.

2.- Indica si ese resultado es correcto?

3.- Qué norma no cumple la unión/ join?

4.- Cómo sería correcta? Para comprobar, añade el dni de moroso al principio de la select.

Consulta: quienes son y donde viven los Clientes activos (tabla Cliente) que además son morosos

*Select Cliente.DNI, Cliente.Direccion*

*from Cliente, Moroso*

*where Cliente.Direccion = Moroso.Direccion;*

Devuelve el DNI que no está en Moroso: Hay unión con perdida de información, por hacer la unión de 2 tablas por un atributo que no es CP ni CC en ninguna de las 2 tablas.

**2.1.a** En la tabla Cliente, cuándo podemos considerar CC el atributo Dirección? Describe la respuesta usando la relación que hay entre los valores de los clientes y los valores de las direcciones donde viven.

Cuando el resto de atributos dependan de la dirección, sucede esto Cuando cada cliente viva en una Dirección distinta siempre. Cada fila tiene Dirección distinta.

**2.1.b** Si, en la tabla Puesto, ahora, suponemos que cada título tiene el mismo sueldo para todos los clientes que tengan ese título. 1.- Indica la DF que provoca redundancia. 2.- Inserta filas que provoquen la redundancia (usa “insert”). 3.- Cómo sé si es problemática?. 4.- Cómo la corriges aplicando la teoría?. 5.- En qué FN queda la tabla?. 6.- Indica los axiomas que cumple para estar en esa FN.

Es necesaria corregirla porque hay una DF transitiva DNI -> Titulo -> Sueldo, luego se queda en 2ªFN y debemos tener como minimo 3ªFN para una calidad razonable. Se puede corregir normalizando, descomponiendo en 2 tablas T1(DNI, Titulo) y T2(Titulo, Sueldo) asumiendo que el cliente tiene solo un Título.

En desnormalización se puede considerar no descomponer por ser solo un atributo, y controlar la redundancia con un trigger que se activa cada vez que alguien cambia el sueldo a un puesto, ej.: secretarios.

**2.1.c** Si, en la tabla Puesto, ahora, suponemos que un cliente tiene más de un Título. **1.**- Cuál debe ser la CP? **2.**- Corrige lo necesario en la BD dentro de Oracle con “Alter” **3.**- Inserta filas con datos inventados para que cumpla lo dicho. (usa “insert”).

Para tener identificación única de una fila necesito que la CP sean DNI y Titulo juntos.Sólo así todos los atributos dependen funcionalmente de la clave.

**2.2**- Qué relaciones se tienen que dar (en cuanto al significado de los atributos) en cada apartado, para que exista la DF indicada:

**2.2.X** EJEMPLO resuelto: Si se cumple en Empresa: Cotización Capital,

Solución: cuando las cotizaciones de las empresas determinan el capital que tienen. Es decir, que para dos Empresas con la misma Cotización tendrán el mismo Capital.

**2.2.a** - Tabla Cliente: NombreC -> DNI . 1.- Decir las relaciones. 2.- Comprueba en las filas creadas si se cumplen esas relaciones. 3.- Corrige con update las que no las cumplan, cambiando sus valores (inventa otros).

No puede haber dos clientes con el mismo nombre. Pero sí los hay: las filas del cliente 1,4, y 5 tienen el mismo nombre. Para corregir, hay que inventar nombres únicos para cada cliente. (Si la implemento puedo provocar esa DF poniendo el atributo NombreC como unique).

**2.2.c** - Tabla Compras: Tienda -> NumF , donde NumF es el número de factura. 1.- Decir qué significa la DF. 2.- indica si es razonable esta DF y porqué?

Si cada tienda tuviera un solo NumF: no es razonable que eso suceda.

**2.2.d** - Tabla Compras: NumF -> Tienda 1.- Decir las relaciones. 2.- Indica si es razonable esta DF y porqué?.

Un NumF corresponde a una tienda: solo si no hubiera, para dos tiendas, el mismo NumF, esto es, que la numeración de facturas fuera única para todas las tiendas. Esto no es razonable.

**2.3**- Descubrir la Dependencia y haz un procedimiento para mantener la consistencia de datos. Pasos:

(se asume que los datos actuales de la BD son correctos)

**2.3.a 1.**- Qué dependencia y de qué clase hay en la tabla Invierte, dadas estas reglas de un Broker para invertir: Cada Cliente invierte en varias Empresas, pero está obligado a invertir en los mismos Tipos en cada empresa que invierta; y todas las inversiones del mismo Tipo siempre deben tener la misma cantidad.

Esta situación provoca que:

- cada vez que un Cliente invierte en una empresa nueva, tiene que invertir en todos los tipos en los que había invertido antes.

- También sucede que cada vez que el Cliente quiera invertir en otro Tipo diferente a los que ya tenía, le obligan a invertir en todas las empresas (en las que ya tiene inversiones) con ese Tipo y la misma cantidad que tiene en otras inversiones ese mismo Tipo.

**2.**- Inserta filas con datos inventados para que se cumpla esa dependencia

Es una DM : DNI -->> NombreE , DNI -->> Tipo, cantidad Cuando un cliente decide invertir en un Tipo nuevo, hay que añadir tantas filas como Empresas en las que tiene inversiones. Y cuando se borra un Tipo hay que recorrer toda la tabla borrando las inversiones de ese Tipo. Si hay que borrar una Empresa, hay que borrar las inversiones de cada Tipo.

**--------------------- Práctica 1 - SEMANA 2 --------------**

IMPORTANTE: para estos apartados necesitamos tener la BD como estaba al empezar, para ello:

BORRAR la BD (script *borradb.sql*) y crearla de nuevo (script *BDejemploSimple.sql*) Comprueba que no ha dado ningún error.

**2.3.b (mantengo desnormalizada)** Implementa un procedimiento *insertoConsistente*, en PLSQL, que vigile la consistencia de datos y que mantenga la dependencia del apartado 2.3.a automáticamente cada vez que se inserte una nueva inversión (sus datos son parámetros del proc), actualizando o creando las filas necesarias.

Utiliza, como base para implementar el archivo */para-Apdo-2.3.b/INSERTOCONSISTENTE-*

*Esqueleto.sql* . Para probar el procedimiento, ejecuta */para-Apdo-2.3.b/prueba-insertconsistente.sql*.

Está en INSERTOCONSISTENTE-Esqueleto.sql

**2.3.c (normalizo la tabla)** Cómo podrías arreglar la dependencia del apartado anterior dividiendo en tablas? Describe las tablas, sin implementar nada.

Haciendo dos tablas según la DM :

* Invierte-NombreE(DNI, NombreE)
* Invierte-Tipo(DNI,Cantidad,Tipo)

Así ya se ve que, cada vez que invierta en una nueva empresa, solo inserta en Invierte-NombreE.

**2.3.d** (para NOTA) En otra situación, nos dicen que el número de tarjeta NumT está compuesto por pais, banco, número de Cuenta Local de ese banco. Quedando la tabla así Compras:

CO(DNI, NumF, Pais, Banco, CuentaLocal , Fecha, Tienda, Importe)

**1.-** Describe textualmente qué DFs hay ahora en esa tabla, justificando cada una con el significado que tiene.

La tabla "Compras" tiene los siguientes Dependencias Funcionales (DFs):

1. **DNI, NumF -> Fecha, Tienda, Importe**   
   Esta dependencia funcional indica que dado el DNI y el número de factura, se puede determinar la fecha de la compra, la tienda donde se realizó y el importe de la misma. Es decir, cada compra está asociada a un cliente y a una factura, y a su vez, se realiza en una fecha y en una tienda determinada con un importe específico.
2. **NumF -> Pais, Banco, CuentaLocal**  
   Cada número de factura corresponde a una determinada entidad bancaria de un país específico y un número de cuenta local en esa entidad.
3. **DNI, NumF -> Pais, Banco, CuentaLocal, Fecha, Tienda, Importe**Esta DF significa que el número de tarjeta NumT (compuesto por Pais, Banco y CuentaLocal) es único para cada compra (NumF) realizada por un cliente (DNI). Esto implica que los datos del país, banco y cuenta local están determinados por el número de tarjeta, y por lo tanto, la fecha, la tienda y el importe de la compra están también determinados.
4. **Pais, Banco, CuentaLocal -> DNI**   
   Esta dependencia funcional indica que la información del DNI del comprador está determinada por el país, banco y número de cuenta local utilizados en la transacción. (no en el sentido opuesto porque un cliente identificado por su DNI puede tener cuentas en varios países)

**2.-** Obtener el mínimo conjunto de DFs solo de esa tabla, aplicando los apartados B y C del algoritmo en el documento **tengo-demasiadas-DFs-y-otros-problemas-cierre-minimal.pdf**, tal y como hicimos en clase con el EJ-FN1-f. (solo usando el conjunto de DFs en Compras).

Para obtener el mínimo conjunto de DFs, aplicando los apartados B y C del algoritmo, se debe realizar lo siguiente:

B. Para ver si en una DF sobra algún atributo del lado IZQ, podemos seguir los pasos indicados:

Paso B.1.

1. DNI, NumF -> Fecha
2. DNI, NumF -> Tienda
3. DNI, NumF -> Importe
4. NumF -> Pais
5. NumF -> Banco
6. NumF -> CuentaLocal
7. Pais, Banco, CuentaLocal -> DNI

Paso B.2.

De cada DF del resultado de B.1 : hacemos el cierre del lado izq X con las tres DFs originales y averiguamos si se puede quitar algún atributo del lado izq.

C. Para ver si alguna DF se puede inferir con las otras, podemos seguir los pasos indicados:

* DNI -> NumF, Pais, Banco, CuentaLocal, Fecha, Tienda, Importe: no se puede inferir con las otras.
* NumF -> Pais, Banco, CuentaLocal: se puede inferir con la otra DF, ya que el número de tarjeta determina el país, el banco y el número de cuenta local asociados a esa tarjeta. Por lo tanto, podemos eliminar esta DF sin pérdida de información.

Por lo tanto, el mínimo conjunto de DFs para la tabla Compras es:

* DNI → NumF
* NumF → Pais, Banco, CuentaLocal

**3.-** Obtener una clave primaria (CP o PK) aplicando el apartado E del mismo documento.

Para obtener una clave primaria de la tabla Compras, aplicamos el apartado E del documento:

E.1: Obtenemos el conjunto mínimo de DFs, que ya hemos calculado en el paso C2: DNI → NumF y NumF → Pais, Banco, CuentaLocal.

E.2: Hacemos el cierre de cada atributo de la tabla con respecto a este conjunto de DFs:

* DNI+ = {DNI, NumF, Pais, Banco, CuentaLocal, Fecha, Tienda, Importe}
* NumF+ = {NumF, Pais, Banco, CuentaLocal}

Si ningún otro atributo de la tabla está incluido en el cierre de DNI o NumF, entonces ambos son candidatos a clave primaria. En este caso, podemos elegir cualquiera de los dos como clave primaria. Por ejemplo, podríamos tomar DNI como clave primaria, ya que es un identificador único de los compradores y por lo tanto garantiza la unicidad de cada fila en la tabla.

**APARTADO 3.- Formas Normales (FN): Reconocerlas, y pasar de una FN a la siguiente**

-> *Usando de nuevo la BDEjemplo original* resuelve los siguientes apartados. Tendrás que determinar las clave primaria antes de hacer el apartado:

**3.1** Queremos hacer que la tabla Cliente NO esté en la 1ª FN: 1.- indica cuál es la CP. 2.- Qué relaciones deben cumplir los atributos para que suceda eso. 3.- *Da ejemplos de filas que cumplan lo dicho.*

La CP es: DNI

Por ejemplo si un cliente tiene varios teléfonos, así el atributo Teléfono es multivalorado luego No está en 1ªFN

**3.2** Introduce una DF en la tabla Invierte para que NO esté en la 2ª FN: 1.- Indica cuál es la CP. 2.- Qué relaciones deben cumplir los atributos para que suceda eso (define la CP que usas). 3.- *Da ejemplos de filas que cumplan lo dicho.* 4.- Cuál sería el arreglo?

La CP es: DNI, NombreE, Tipo.

DF: Para un Tipo siempre es la misma Cantidad. Tipo → Cantidad

No está en 2ªFN porque esta DF depende parcialmente de la CP.

Arreglo: Hacer dos tablas: invierte-a(DNI, NombreE, Tipo) , Tipo-C(Tipo, Cantidad)

Desnormalización: Como es solo un atributo, no sería muy problemático dejar que se repitiera, si a cambio tenemos que acceder a dos tablas frecuentemente en lugar de una sola.

**3.3** Introduce una DF en la tabla Empresa para que NO esté en la 3ª FN: 1.- Indica cuál es la CP. 2.- Qué relaciones deben cumplir los atributos para que suceda eso (define la CP que usas). 3.- *Da ejemplos de filas que cumplan lo dicho.* 5.- Cuál sería el arreglo?

La CP es NombreE.

La DF: Cotización → Capital siempre una cotización tiene el mismo capital.

No está en 3ªFN porque hay una DF transitiva: NombreE → Cotización → Capital

Arreglo: Hacer dos tablas : empresa-1(NombreE, Cotizacion) y CotizacionE (Cotizacion Capital)

Desnormalización: Como es solo un atributo, no sería muy problemático dejar que se repitiera, si a cambio tenemos que acceder a dos tablas frecuentemente en lugar de una sola.

**3.4** Introduce una DF en la tabla Tarjeta para que NO esté en la FNBC: 1.- Indica cuál es la CP. 2.- Qué relaciones deben cumplir los atributos para que suceda eso (define la CP que usas). 3.- *Da ejemplos de filas que cumplan lo dicho.* 4.- Cuál sería el arreglo?

La CP es NumT, Organizacion.

La DF es TipoT -> Organizacion , no está en FNBC porque una parte de la clave Y depende de un atributo X no superclave.

Arreglo: Hacer dos tablas, pero puede perder la DF que hacía la clave primaria, por lo que no es recomendable en este caso.