



The background image shows an airport terminal with several large flight information screens. The leftmost screen displays 'International Flights' with a list of destinations including Paris, London (marked as DELAYED), Moscow, New York, Havana, Incheon, and Tokyo. The middle screen also shows 'International Flights' with destinations like Rome, Paris, Osaka, California, Taipei, Incheon, and Tokyo. The rightmost screen displays 'Domestic Flights' with destinations like Paris, London, Moscow, New York, Incheon, and Tokyo. In the foreground, there are check-in counters and a computer monitor displaying a flight information system interface with a mouse cursor pointing at it.

Time	Destination	Airline	Gate
14:02	PARIS	IGR 401	3
17:41	LONDON DELAYED	KE 2344	24
17:45	MOSCOW	FM 5415	12
18:30	NEW YORK	OW 201	9
20:40	HAVANA	SPM 325	11
22:12	INCHEON	KE 1980	23
22:36	TOKYO	KE 120	15

Time	Destination	Airline	Gate
14:02	ROME	JM 402	1
17:41	PARIS	KE 2344	24
18:30	OSAKA	OW 201	9
17:45	CALIFORNIA	FM 5415	12
20:40	TAIPEI	SPM 325	11
22:12	INCHEON	KE 1980	23
22:36	TOKYO	KE 120	15

Time	Destination	Airline	Gate
14:02	PARIS	IGR 401	3
17:41	LONDON	KE 2344	24
17:45	MOSCOW	FM 5415	12
18:30	NEW YORK	OW 201	9
20:40	HAVANA	SPM 325	11
22:12	INCHEON	KE 1980	23
22:36	TOKYO	KE 120	15

UNIDAD 10. NORMALIZACIÓN DE BASES DE DATOS RELACIONALES

CONTENIDOS:

1.PROBLEMAS DE DISEÑO DE BD RELACIONALES Y NORMALIZACIÓN.

2.APROXIMACIÓN INTUITIVA A LA NORMALIZACIÓN.

2.1. PRIMERA FORMA NORMAL (1FN).

2.2. DEPENDENCIAS FUNCIONALES.

2.3. SEGUNDA FORMA NORMAL (2FN).

2.4. TERCERA FORMA NORMAL (3FN).

2.5. FORMA NORMAL DE BOYCE-CODD (BCFN).

2.6. RESUMEN

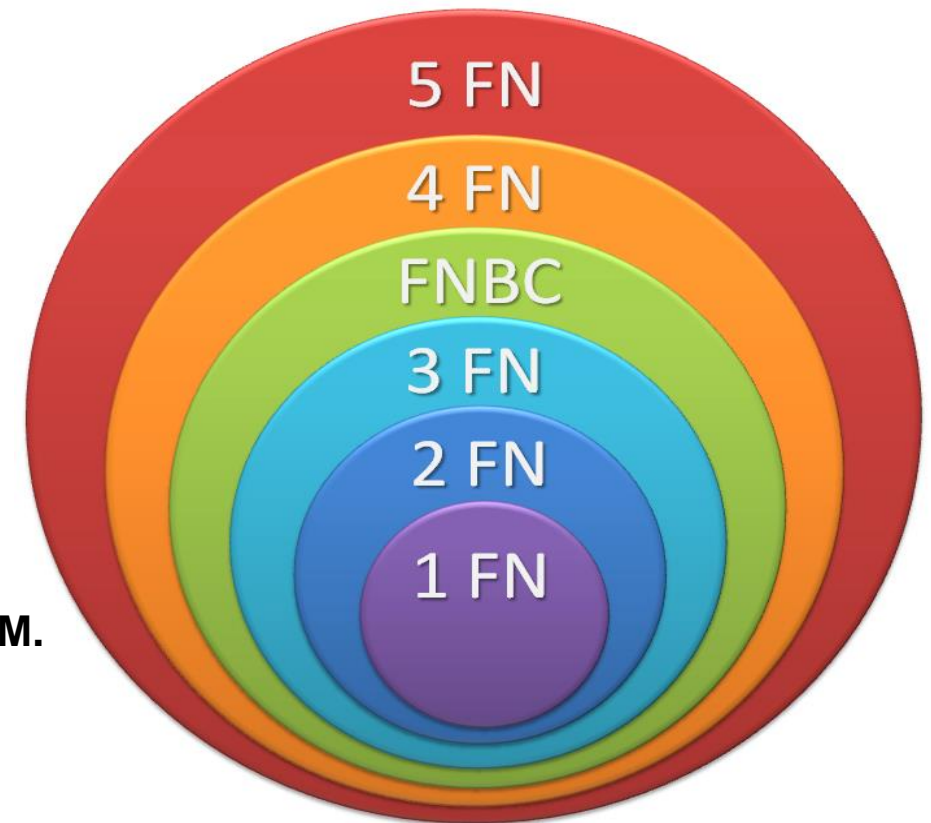
3. OTRAS FORMAS NORMALES.

3.1. DEPENDENCIAS MULTIVALUADAS: FN4 O FNDM.

3.2. REUNIÓN PROYECCIÓN (FN5 O FNRP)

3.3. DOMINIOS Y CLAVES (FN6 O FNDC).

4. METODOLOGÍA.



1. PROBLEMAS DE DISEÑO DE BD RELACIONAL

- El diseño de una BD relacional necesita encontrar una “**buena**” colección de esquemas de relación (tablas).
- **Un mal diseño puede provocar:**
 - Información repetida (redundante).
 - Imposibilidad de representar cierta información.
 - Operaciones no eficientes.
 - Facilitar la aparición de datos inconsistentes.
- **Objetivos del diseño:**
 - Evitar datos redundantes.
 - Asegurar que se representan las relaciones y sus atributos.
 - Facilitar la comprobación de actualizaciones para que no violen las restricciones de integridad.

1. PROBLEMAS DE DISEÑO DE BD RELACIONAL

EJEMPLO: Queremos una BD sobre un banco, sus sucursales y los préstamos solicitados por su clientes. Hacemos el siguiente esquema:

Prestamo = *Nombre-sucursal* + *ciudad-sucursal* + *activo* + *nombre-cliente* + *Numero-prestamo* + *importe*

activo: cantidad de dinero que tiene la sucursal *nombre-sucursal*

importe: cantidad de dinero prestada al cliente

<i>nombre-sucursal</i>	<i>ciudad-sucursal</i>	<i>activo</i>	<i>nombre-cliente</i>	<i>número-préstamo</i>	<i>importe</i>
Centro	Arganzuela	9.000.000	Santos	P-17	1.000
Moralzarzal	La Granja	2.100.000	Gómez	P-23	2.000
Navacerrada	Aluche	1.700.000	López	P-15	1.500
Centro	Arganzuela	9.000.000	Sotoca	P-14	1.500
Becerril	Aluche	400.000	Santos	P-93	500
Collado Mediano	Aluche	8.000.000	Abril	P-11	900
Navas de la Asunción	Alcalá de Henares	300.000	Valdivieso	P-29	1.200
Segovia	Cerceda	3.700.000	López	P-16	1.300
Centro	Arganzuela	9.000.000	González	P-18	2.000
Navacerrada	Aluche	1.700.000	Rodríguez	P-25	2.500
Galapagar	Arganzuela	7.100.000	Amo	P-10	2.200

Insertamos un préstamo concedido por la **oficina Navacerrada** a la señora **Fernández** de 1500€. Hay que rellenar una fila donde vamos a repetir el activo que tiene la sucursal Navacerrada. Y esto de repetir supone:

- **Redundancias:** nombre-sucursal, ciudad-sucursal y activo, se repiten en cada nuevo préstamo → consumo de disco. Se complican las modificaciones (si cambia el activo, hay que actualizar muchas filas) y aumenta la posibilidad de inconsistencias.
- **Valores Null:** No se puede almacenar información de una sucursal si no existe un préstamo o para hacerlo, usamos valores null, pero son más complicados de manejar.

1. PROBLEMAS DE DISEÑO DE BD RELACIONAL

DESCOMPOSICIÓN

- El problema de la relación anterior es que se sabe que:
 - Por un lado, una sucursal tiene un único valor de *activo* y se representa así:
 - **nombre-sucursal** → **activo**
 - Por otro lado, una sucursal puede conceder muchos préstamos, por tanto, descomponemos la tabla **PRÉSTAMO** en otras dos (sin mezclar cosas que se repiten con las que no):

OFICINAS = nombre-sucursal + ciudad-sucursal + activo

PRÉSTAMOS = nombre-cliente + número-préstamo + nombre-sucursal
+ importe

PROPIEDADES DESEABLES DE UNA DESCOMPOSICIÓN:

- Todos los atributos de la tabla original (R) deben aparecer en la descomposición (R_1, R_2):
- **DESCOMPOSICIÓN DE UNIÓN SIN PÉRDIDAS.** Para todos los posibles datos en cada tabla (r en la tabla R), los datos de R (llamados r), pueden obtenerse a partir de los de R1 y R2.

$$r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$$

Datos de préstamos por un lado y de oficinas por otro

Símbolo matemático para JOIN

1. PROBLEMAS DE DISEÑO DE BD RELACIONAL

UNA MALA DESCOMPOSICIÓN

Podemos hacer una mala descomposición, incluso manteniendo todos los atributos de la tabla de la que partimos. Ejemplo de descomposición de la tabla $R = A + B$

$$R = A + B \quad R_1 = A \quad y \quad R_2 = B$$

A	B
α	1
α	2
β	1

r

A
α
β

$\Pi_A(r)$

B
1
2

$\Pi_{B(r)}$

$$\Pi_A(r) \bowtie \Pi_B(r)$$

A	B
α	1
α	2
β	1
β	2

Hay datos que no estaban. Esta descomposición no es buena.

1. PROBLEMAS DE DISEÑO DE BD RELACIONAL

OBJETIVOS DE LA NORMALIZACIÓN:

- Decidir cuando una relación R tiene una “buena” forma.
- En el caso de que una relación R no esté en una forma “buena”, indicar cómo descomponerla en un conjunto de relaciones $\{R_1, R_2, \dots, R_n\}$ de forma que
 - **Primero:** Cada nueva relación esté en una forma buena
 - **Segundo:** La descomposición es de unión sin pérdidas.
- Nuestra teoría se basa en:
 - Dependencias funcionales.
 - Dependencias multivaluadas.

Nosotros veremos las primeras, las más usadas en BD de tipo OLTP.

OLTP - On-Line Transactional Processing: Los sistemas OLTP son bases de datos orientadas al procesamiento de transacciones. Una transacción genera un proceso atómico (que debe ser validado con un commit, o invalidado con un rollback), y que puede involucrar operaciones de inserción, modificación y borrado de datos. El proceso transaccional es típico de las bases de datos operacionales.

2. NORMALIZACIÓN INTUITIVA.

- La normalización es una **técnica pensada por E.F.Codd en 1972** para diseñar la estructura lógica de una BD relacional.
- **Es una estrategia de abajo a arriba (down-to-up):** se parte de los atributos y la normalización los va agrupando en tablas.
- La mayoría de metodologías de diseño de BD relacionales, sin embargo, la usan como una etapa posterior al diseño conceptual y al diseño lógico. Una mejora que elimina las dependencias no deseadas de los atributos en cada tabla.
- **Conseguimos:**
 - Evitar anomalías en inserciones, modificaciones y borrados de datos.
 - Mejorar la independencia de los datos.
 - No establecer restricciones artificiales en la estructura de los datos.
- Se basa en el concepto de **Dependencias**. Una dependencia es una propiedad inherente a la semántica de los datos (su significado). Forman parte de las restricciones del usuario del modelo relacional. Hay varios tipos de dependencias:
 - **Dependencias Funcionales:** completa, parcial y transitiva.
 - Dependencias multivaluadas.
 - Dependencias de reunión.

2.1. DEPENDENCIAS FUNCIONALES (DF).

- Dada una tabla T, decimos que el atributo(s) T.y depende funcionalmente del atributo(s) T.x, y se escribe así:

$$T.x \rightarrow T.y$$

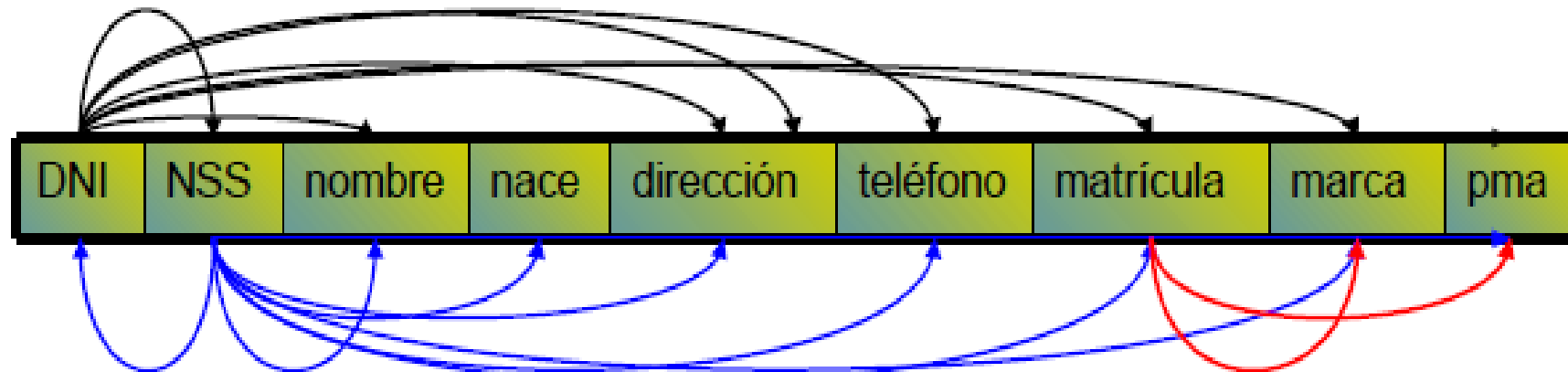
- Y se lee de estas dos formas:
 - T.x determina funcionalmente a T.y
 - T.y depende funcionalmente de T.x
- Si y sólo si un único valor de Y está asociado a cada valor de X en la tabla T.
- **Ejemplo. La tabla PERSONA almacena información de personas y sus vehículos:**

PERSONA = dni + NSS + nombre + nace + dirección + teléfono + matrícula +
marca + PMA

- **DNI \rightarrow nombre**, en el caso de que la tabla no pueda tener estas filas:
 - 55555555A + Jose Luis + ...
 - 55555555A + Mari Carmen + ...
- Si cada vez que aparezca el DNI 55555555A, el nombre debe ser Jose Luis, se dice que nombre depende funcionalmente de dni o que dni determina funcionalmente a nombre (en la tabla PERSONA, un valor de dni no puede tener asociados varios valores de nombre).

2.1. DEPENDENCIAS FUNCIONALES (DF).

- **Dicho de otro modo:** Dada una tabla t , y dos conjuntos de atributos suyos llamados x e y , decimos que $x \rightarrow y$ si cuando dos filas de T tienen igual valor de x , también tienen el mismo valor y .
- **PROPIEDAD:** Todos los atributos de una tabla dependen funcionalmente de la clave primaria de la tabla.
- **DEPENDENCIA FUNCIONAL COMPLETA:** se dice que el conjunto de atributos Y de la tabla T es dependiente funcional por completo del conjunto de atributos X , siempre que $X \rightarrow Y$ y no exista un subconjunto propio de X del que Y dependa funcionalmente.
 - **Ejemplo:** Sea una tabla PERSONA similar a la anterior. PERSONA = dni + NSS + nombre + nace + dirección + teléfono + matrícula + marca + Km
 - $\{dni, nombre\} \rightarrow nace$, pero no es una DF completa, porque hay un subconjunto de $\{DNI, nombre\}$ del que depende nace: $\{DNI\} \rightarrow nombre$ y por tanto $\{DNI\} \rightarrow nace$.
- Las DF se representan mediante un **diagrama de dependencias**. En el esquema anterior, si NSS es una clave alternativa, tendríamos el siguiente diagrama:



2.1. DEPENDENCIAS FUNCIONALES (DF).

OBSERVACIONES:

- La DF es una propiedad semántica. Reconocer las dependencias funcionales es parte del proceso de entender qué significan los datos en nuestra base de datos. El hecho de que $\text{DNI} \rightarrow \text{nombre}$ significa que cada persona tiene un único nombre.
 - Existe una restricción en el mundo real representado por la BD: cada persona tiene un único nombre. Esa restricción debe cumplirse en la BD.
 - La forma de garantizarlo es especificarlo en el esquema, para que el SGBD pueda hacer que se cumpla.
- Como una DF es una restricción de integridad, hay que expresar $\text{DNI} \rightarrow \text{nombre}$ en algún lenguaje:
 - **REGLA DE INTEGRIDAD:** comprueba que para toda fila FX y para toda fila FY (si $\text{FX.dni} = \text{FY.dni}$ entonces $\text{FX.nombre} = \text{FY.nombre}$)
- Las DF representan relaciones de cardinalidad 1:N. La DF **$\text{DNI} \rightarrow \text{nombre}$** puede leerse:
 - Cada nombre puede asociarse a muchos DNI y cada DNI se asocia a un único nombre.
- La integridad referencial es parecida a la DF. Pero una DF se aplica en el interior de una tabla y la referencia puede superar los confines de la tabla.

2.1. DEPENDENCIAS FUNCIONALES (DF).

OBSERVACIONES:

- Asumir una DF implica dar un significado a la información.
- Es posible que dos organizaciones den un significado diferente a la misma información o que incluso a una misma organización le interese interpretarla de diferente manera en dos lugares o escenarios.
- Es decir, **no hay reglas generales a la hora de asumir una DF**. Depende de como se quiera/necesite interpretar los datos.
- Pero no te engañes, eso no significa “*Como no hay reglas fijas, todo vale*”. La realidad es: no hay reglas fijas, pero mi BD sigue unas (tiene un significado de todos los posibles), por tanto, si no las reconoces o no las indicas, tendrás un diseño que genera problemas.

- **EJEMPLO (misma información, diferentes DF):**

PERSONA = DNI + nombre + peso

Una persona (**dni**) solo puede tener un **peso**, es decir en esta tabla dni → peso

EVOLUCIÓN_DE_PACIENTE = DNI + fecha + peso

Un paciente va cambiando de peso de una fecha a otra, por tanto, una misma persona puede tener distintos pesos, es decir, en esta tabla, **peso** no depende de **dni** (un mismo dni, diferentes pesos).

2.2. PRIMERA FORMA NORMAL (1FN).

- El dominio de un atributo es **atómico** si sus elementos se consideran unidades indivisibles.
 - Ejemplos de dominios no atómicos:
 - Partes del nombre de una persona: (nombre, apellido1, apellido2).
 - Código compuesto: identificación de una habitación N301 que tiene 3 partes: N (ala norte), 3 (piso 3), 01 (habitación 01).
 - Una lista de teléfonos: 96311111, 910234567, 95345698, ...
- Una tabla T está en **primera forma normal (1FN)** si los dominios de todos sus atributos son atómicos.

PROBLEMAS SI LAS TABLAS NO ESTÁN EN PRIMERA FORMA NORMAL (1FN)

Los valores no atómicos complican el almacenamiento y permiten que el SGBD relacional no pueda controlar las redundancias (repeticiones) de datos almacenados.

- Para facilitarnos la vida, muchas veces asumiremos que las tablas están en primera forma normal.

2.2. PRIMERA FORMA NORMAL (1FN).

- La atomicidad es más una propiedad de **cómo se usan los dominios de los atributos que de los propios atributos**. Es decir, un mismo atributo, a veces será considerado como atómico (si se utiliza de esta forma) o como compuesto (si en la BD hay trabajos que necesiten usar o manipular de forma aislada alguna de las partes que lo forman).
 - Ejemplo: los strings normalmente podrían considerarse indivisibles.
 - Supongamos que a los estudiantes se les asigna un número de matrícula que son strings de la forma *CS0012* o *EE1127*
 - Si los dos primeros caracteres se extraen, representan el departamento, por tanto el dominio del número de matrícula no es atómico.
 - **Hacer esto es una mala idea**: dejas codificación de la información en manos de usuarios y aplicaciones en vez de en manos del SGBD. Por tanto, vuelves a tener algunos de los problemas que ya se tenían en los sistemas de ficheros. ¿Qué ocurre si un usuario modifica un número de matrícula y deja un departamento que no existe, por ejemplo CD en vez de CS? El SGBD no tiene capacidad para detectar esos errores → el diseño debilita la capacidad de mantener datos coherentes en el sistema.

2.2. PRIMERA FORMA NORMAL (1FN).

CONVERTIR A PRIMERA FORMA NORMAL.

Para que una tabla esté en 1FN sus atributos deben ser atómicos. No deben tener partes $X = (x1, x2 \dots x3)$ ni ser una lista de varios valores $\{\text{teléfonos}\} = \{\text{teléfono1, teléfono2, ...}\}$.

Ej: Jose Miguel de los Prados y todos los Ángeles.
Nombre: 'Jose Miguel'. Par1: 'de los'. Ape1: 'Prados'

PERSONA = dni + nombre(nombre, par1, ape1, par2, ape2) + {teléfonos}

- **Eliminar las partes:** cada parte será un atributo individual.
PERSONA1 = dni + nombre + par1 + ape1 + par2 + ape2 + {teléfonos}
- **Eliminar listas:** Rompemos la tabla original en 2 (como el mapeo de atributos multivaluados).

PERSONA1 = dni + nombre + par1 + ape1 + par2 + ape2

PERSONA2 = dni + teléfono

Clave ajena: dni → PERSONA1(dni)

2.3. SEGUNDA FORMA NORMAL (2FN).

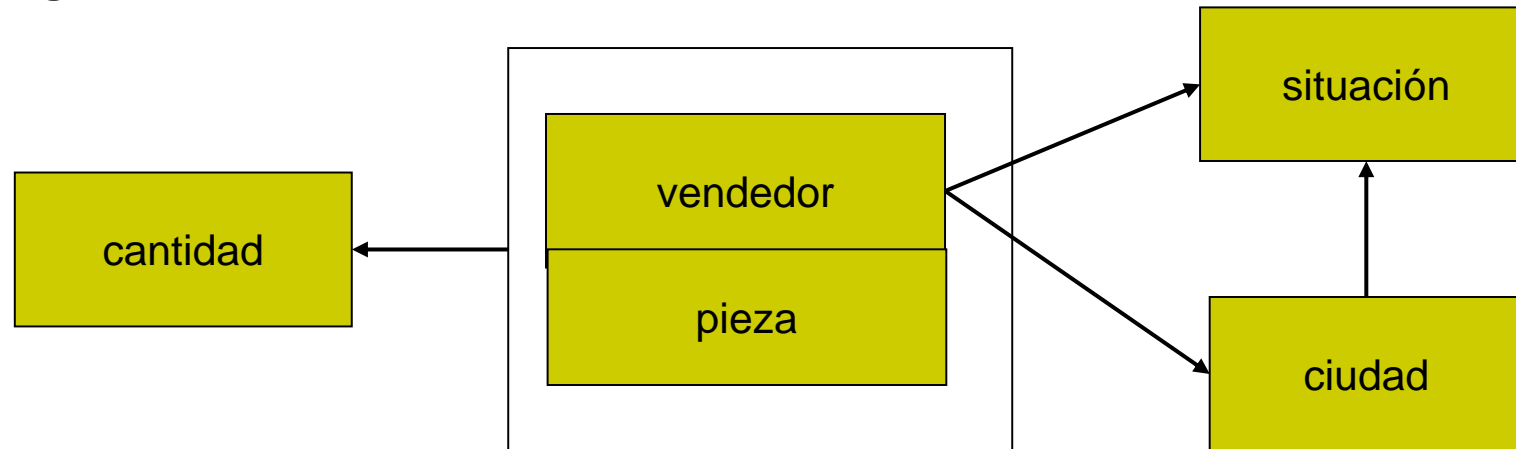
- Vamos a ver un ejemplo de tabla, que estando en 1FN, no está en 2FN y veremos los problemas que esto causa. La tabla es:

VENEDORES = vendedor + situación + ciudad + pieza + cantidad

Clave Primaria: vendedor + pieza

- En este caso, a simple vista podemos ver el problema: estamos mezclando la información de un vendedor con la información que describe un pedido que atiende ese vendedor.

El diagrama de dependencias sería este:



- Este diagrama tiene cierta complejidad

2.3. SEGUNDA FORMA NORMAL (2FN).

- Las redundancias son obvias, si observamos una posible extensión de la tabla:

ANOMALÍAS:

VENDEDORES				
VENDEDOR	SITUACIÓN	CIUDAD	PIEZA	CANTIDAD
V1	20	Alicante	P1	300
V1	20	Alicante	P2	200
V1	20	Alicante	P3	400
V1	20	Alicante	P4	200
V1	20	Alicante	P5	100
V1	20	Alicante	P6	100
V2	10	Castellón	P1	300
V2	10	Castellón	P2	400
V3	10	Torrente	P2	200
V4	20	Valencia	P2	200
V4	20	Valencia	P4	300
V4	20	Valencia	P5	400

- **INSERCIÓN:** No podemos insertar un nuevo vendedor en la BD, hasta que suministre una pieza (la pieza es parte de la clave primaria).
- **BORRADO:** Si borramos la fila de V3, no solo borramos el envío de 200 piezas P2, también borramos la ciudad y la situación del vendedor (perdemos más información de la que queremos borrar).
- **MODIFICACIÓN:** Si el vendedor V1 cambia de ciudad, debemos buscar todas las filas de ese vendedor para modificarlas todas (si no, tenemos inconsistencias).

2.3. SEGUNDA FORMA NORMAL (2FN).

- **PROBLEMA INTUITIVO:** mezcla información de vendedores y pedidos.
- **PROBLEMA TÉCNICO:** hay atributos que dependen funcionalmente de una parte de la clave primaria, no de toda ella.
- **SOLUCIÓN:** separar la información en dos tablas que no tengan ese problema.
- **DEFINICIÓN DE SEGUNDA FORMA NORMAL:** Una tabla está en segunda forma normal, si está en 1FN y además todos los atributos no clave dependen por completo de la clave primaria.
- **CONVERTIR A SEGUNDA FORMA NORMAL:**
 - Si una tabla no está en 2FN, se descompone en un conjunto de tablas equivalentes que sí lo están.
 - **Para que una tabla en 1FN no esté en 2FN, su clave debe ser compuesta.**
 - **El proceso es el siguiente:** Dada la tabla $T = A + B + C + D$ con clave primaria $(A + B)$ y una dependencia funcional $A \rightarrow D$, se sustituye T por:
 - $T1 = A + D$
 - $T2 = A + B + C$ y clave ajena $A \rightarrow T1$
 - La tabla T puede recuperarse si se hace una reunión de clave ajena a primaria de $T2$ a $T1$.

2.3. SEGUNDA FORMA NORMAL (2FN).

EJEMPLO DE TRANSFORMACIÓN A 2FN

En el ejemplo, partimos de la tabla:

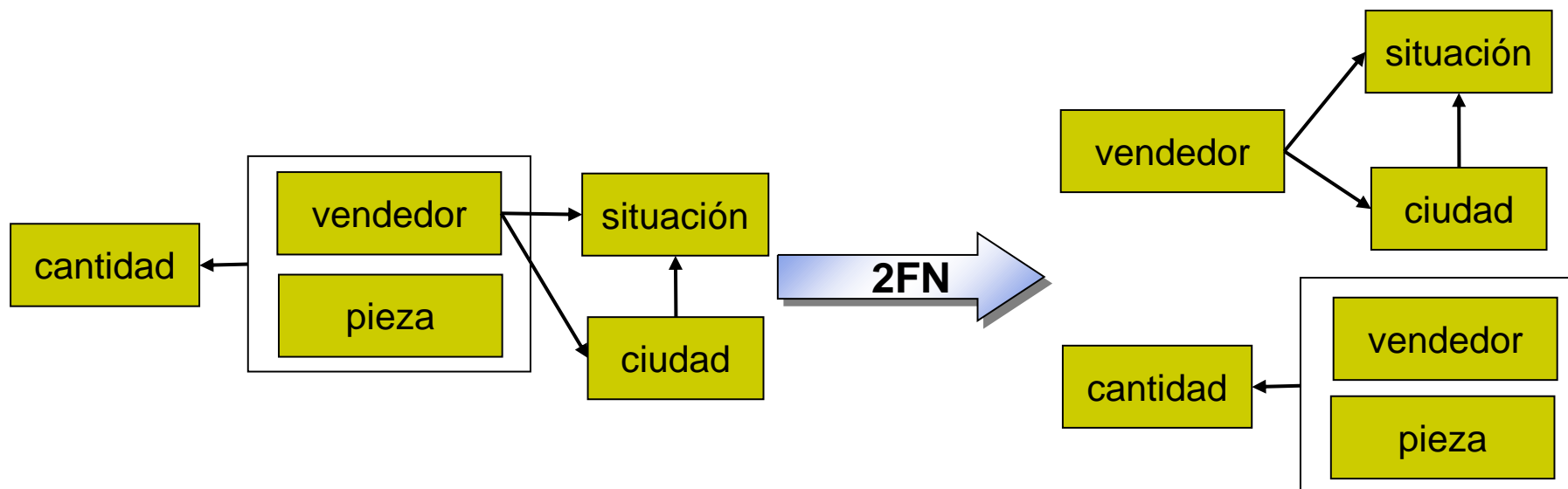
VENDEDOR = vendedor + situación + ciudad + pieza + cantidad

- ¿Está en 1FN? Sí
- ¿Puede no estar en 2FN? Es decir, ¿Su clave primaria es compuesta? Sí. ¿Está en 2FN? No. Su clave primaria es (vendedor + pieza) pero ciudad y situación dependen funcionalmente de vendedor, no de toda la clave primaria. La descomposición es:

VENDEDOR = vendedor + ciudad + situación

PEDIDO = vendedor + pieza + cantidad

Clave ajena: vendedor → VENDEDOR(vendedor)



2.3. SEGUNDA FORMA NORMAL (2FN).

COMPROBAR LAS VENTAJAS.

¿Las anomalías se han solucionado?

PEDIDO		
VENDEDOR	PIEZA	CANTIDAD
V1	P1	300
V1	P2	200
V1	P3	400
V1	P4	200
V1	P5	100
V1	P6	100
V2	P1	300
V2	P2	400
V3	P2	200
V4	P2	200
V4	P4	300
V4	P5	400

VENDEDOR		
VENDEDOR	SITUACIÓN	CIUDAD
V1	20	Alicante
V2	10	Castellón
V3	10	Torrente
V4	20	Valencia

ANOMALÍAS:

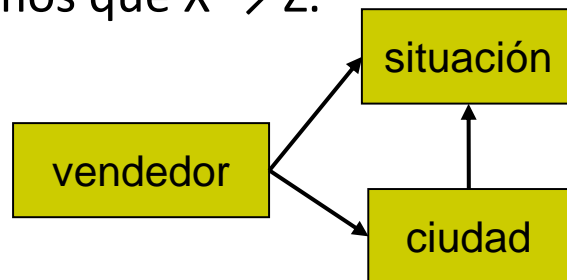
INSERCIÓN: podemos insertar un nuevo vendedor en la BD, sin que suministre una pieza.

BORRADO: Si borramos la fila de PEDIDOS de V3, solo borramos el envío de 200 piezas P2, pero no perdemos la ciudad y situación de V3.

MODIFICACIÓN: Si el vendedor V1 cambia de ciudad, debemos modificar una sola fila.

2.4. TERCERA FORMA NORMAL (3FN).

- Continuamos con el ejemplo anterior. La tabla PEDIDO ya está en 3FN, por tanto la dejaremos tranquila. Pero la tabla VENDEDOR tiene la dependencia funcional de situación respecto de ciudad y es una dependencia funcional completa, pero transitiva (a través de ciudad).
- Como norma general, si se cumplen las DF $X \rightarrow Y$ y $Y \rightarrow Z$, por aplicación de la transitividad, tenemos que $X \rightarrow Z$.



- **LAS DEPENDENCIAS TRANSITIVAS PRODUCEN ANOMALÍAS.**
 - **A. DE INSERCIÓN:** no podemos insertar el hecho de que cierta ciudad tiene una situación, hasta que no tengamos un vendedor en esa ciudad.
 - **A. DE BORRADO:** si eliminamos un vendedor de VENDEDOR, no solo se borra el vendedor, podemos perder también la situación de una ciudad, si es el único vendedor de esa ciudad.
 - **A. DE MODIFICACIÓN:** la situación de una ciudad se repite tanto como vendedores haya en esa ciudad (redundancias). Si cambia la situación, habrá que cambiar todas las filas donde aparezca o tener inconsistencias.

2.4. TERCERA FORMA NORMAL (3FN).

La solución, una vez más, es sustituir VENDEDOR por dos relaciones equivalentes que sí estén en 3FN.

DEFINICIÓN DE TERCERA FORMA NORMAL: Una relación está en tercera forma normal, si está en segunda forma normal y además, todos sus atributos no clave dependen de manera no transitiva de la clave primaria.

- **PASAR UNA TABLA A 3FN:**
 - Dada una tabla T de tipo:
 - $T = A + B + C$ y las DF = $\{ B \rightarrow C \}$
 - La normalización recomienda sustituir T por sus dos proyecciones T1 y T2:
 - $T1 = B + C$
 - $T2 = A + B$ y Clave ajena $B \rightarrow T1$

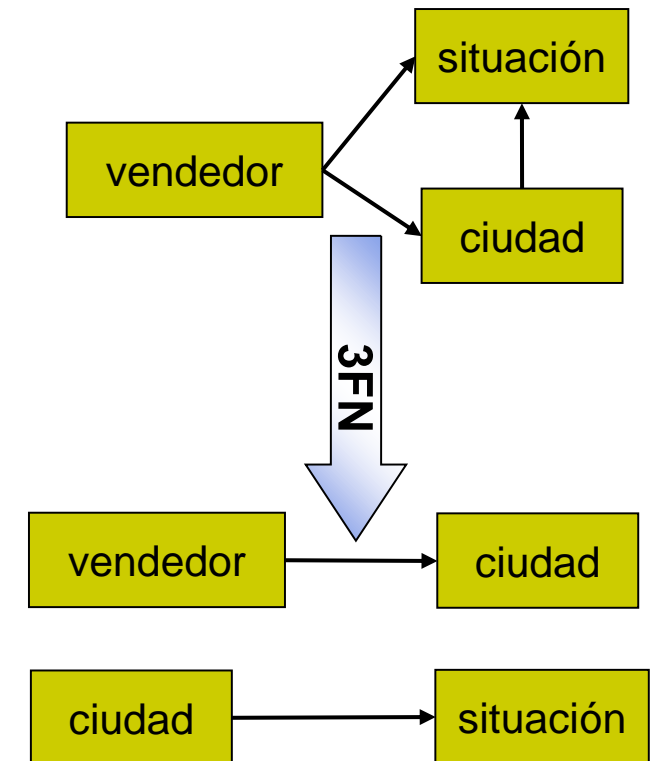
2.4. TERCERA FORMA NORMAL (3FN).

Descomponemos manteniendo la DF ciudad \rightarrow situación. Si lo hacemos de otra forma sería una descomposición con pérdidas. Ejemplo:

- **CASO A: T1 = vendedor + ciudad y T2 = vendedor + situación.** Así convertimos una DF intratabla (una restricción) en una restricción intertablas.
- **CASO B (la correcta): T1 = ciudad + situación y T2 = vendedor + ciudad,** la restricción intertablas es la transitiva, que se cumple si se cumplen las dos intratabla (vendedor \rightarrow ciudad, ciudad \rightarrow situación)

TEOREMA DE RISSANEN: (Ayuda a elegir la ruptura correcta)

- Toda DF en T se puede deducir a partir de las DF en T1 y T2, y
- Los atributos comunes de T1 y T2 forman una clave candidata de al menos una de las dos tablas.
- **CASO A:** Quedan las DF {Vendedor \rightarrow ciudad, vendedor \rightarrow situación} y no se deduce la otra DF.
- **CASO B:** DF = {vendedor \rightarrow ciudad, ciudad \rightarrow situación} y por transitividad, se deduce vendedor \rightarrow situación
 - El atributo común de T1 y T2 es ciudad y es clave candidata (de hecho primaria) en T2. **Es la correcta.**



2.5. FORMA NORMAL BOYCE/CODD (BCFN).

La definición original de 3FN de Codd tenía ciertas deficiencias. No maneja bien el caso de una tabla en la cual:

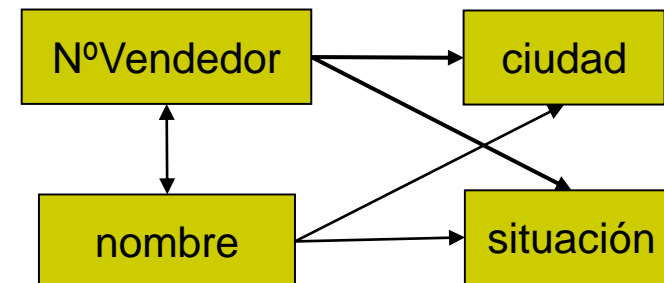
- Hay varias claves candidatas.
- Esas claves candidatas son compuestas.
- Son traslapadas (tienen al menos un atributo en común).
- Boyce y Codd intentando mejorar la 3FN, crearon una nueva forma normal más potente que la 3FN y se llama forma normal de Boyce-Codd. En realidad, raramente se presenta esta necesidad. Para definirla, debemos introducir el concepto de determinante.
- **DETERMINANTE:** atributo del cual depende funcionalmente de forma completa algún otro.

FORMA NORMAL DE BOYCE/CODD: Una relación R está en forma normal de Boyce/Codd, si todo determinante es una clave candidata.

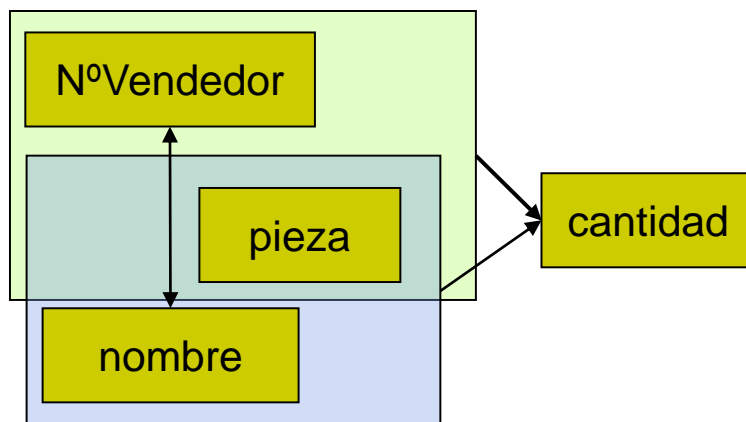
- **Otra forma de decirlo:** sólo salen flechas desde las claves candidatas en los diagramas de dependencias funcionales.
- **PASAR A FNBC:** Sea $T = A + B + C + D$ una tabla donde $(A+C)$ y $(B+C)$ son claves candidatas y $A \rightarrow B$. (Luego A es un determinante pero no es clave candidata, aunque sí están en 3FN). Descomponer en:
 - **$T1 = A + B$ y $T2 = A + C + D$ o bien $T1 = A + B$ y $T2 = B + C + D$**

2.5. FORMA NORMAL BOYCE/CODD (BCFN).

EJEMPLO 1: vendedor = N°Vendedor + nombre + ciudad + situación donde N°Vendedor y nombre son claves candidatas. ¿Está en BCFN? No está en 3FN y por eso no está en BCFN, pero solo salen flechas de los determinantes.

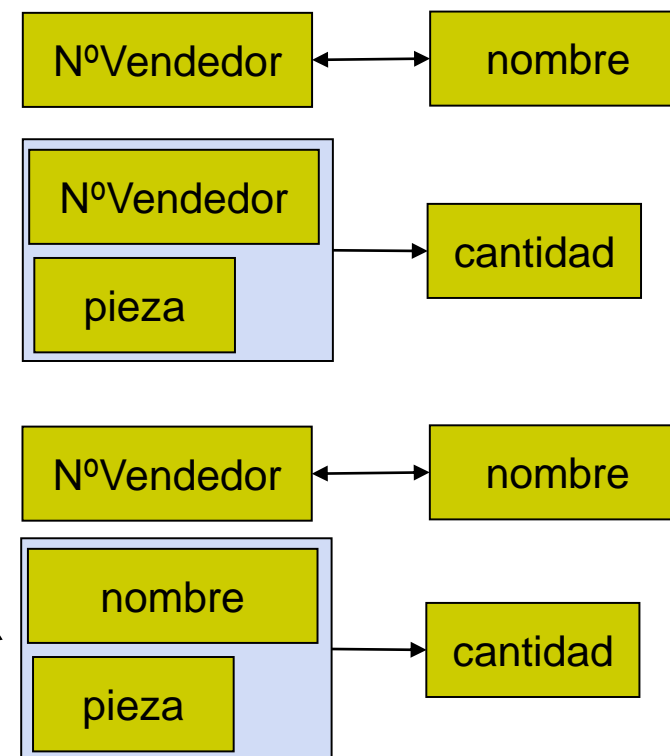


EJEMPLO 2: vendedor = N°Vendedor + nombre + pieza + cantidad donde (N°Vendedor + pieza) es la clave primaria y (nombre + pieza) es clave alternativa. ¿Está en FNBC?



FNBC

FNBC

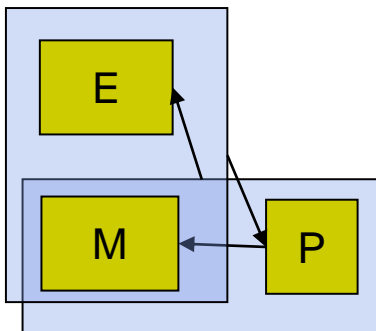


2.5. FORMA NORMAL BOYCE/CODD (BCFN).

EJEMPLO

Tenemos la tabla **EMP** = **E** + **M** + **P** que significa “el estudiante *E* estudia la materia *M* impartida por el profesor *P*”.

- Para cada materia, cada estudiante tiene un solo profesor { $M+E \rightarrow P$ }.
- Cada profesor imparte una sola materia (una materia la dan varios profesores): { $P \rightarrow M$ }.
- Un profesor solo da una materia a cada alumno: { $P+M \rightarrow E$ }
- Haz el diagrama de dependencias funcionales.



EMP		
ESTUDIANTE	MATERIA	PROFESOR
Gómez	Mates	Flores
Gómez	Física	Rosas
Pérez	Mates	Flores
Pérez	Física	Ramos

- ¿Cuáles son las claves candidatas? (E+M) y (M+P)
- ¿Está en 3FN? ¿Hay atributos no clave con DF transitivas? No. Luego en 3FN.
- ¿Está en BCFN? No. P no es determinante y determina a M.
- Descomponer en FNBC:
 - T1 = P + M y T2 = P + E
- ¿Algún problema?
 - Si. Perdemos la DF $P \rightarrow M$ al no poder deducirla a partir de { $E+M \rightarrow P$ }

Perdemos información: una restricción a cambio de no tener ciertas anomalías al modificar

2.6. RESUMEN.

- Nuestro proceso de diseño consiste en un primer diseño conceptual:
 - Basado en DER.
- Traducir ese esquema conceptual al modelo relacional.
- Por último, repasar el diseño de cada tabla, en busca de dependencias entre atributos, que se nos hayan escapado y que puedan generar redundancias (y sus problemas: inconsistencias e ineficiencias).
 - Intentaremos dejar cada tabla en FNBC o en 3FN (para sistemas de explotación -consultas y modificaciones- transaccionales es lo mínimo normalmente).
 - Nuestro acercamiento a la Normalización es intuitivo, porque, aunque así no tenemos garantías de subsanar todas las anomalías:
 - Al venir de un diseño previo, casi todo estará bien.
 - **El acercamiento matemático tiene ventajas: es exhaustivo y permite formalizar** lo que denominamos buenas prácticas. Esto permite a su vez crear algoritmos (y programas) que hagan el trabajo. Pero aun así, los algoritmos y los programas desconocen la semántica de cada atributo y eso debe seguir proporcionándolo el diseñador.

2.6. RESUMEN.

RESUMIENDO:

- **1FN: Una tabla T está en primera forma normal si los dominios de todos sus atributos son atómicos.**
- **2FN un atributo no es determinado por una parte de la clave primaria.**
 - Si la clave primaria no es compuesta, la tabla está en 2FN.
 - Si $T = a_0 + a_1 + a_2 + a_3$ y $a_1 \rightarrow a_2$ No está en 2FN.
- **3FN: No hay dependencias entre atributos no clave.**
 - Si no hay más de un atributo no clave, está en 3FN.
 - Si $R = a_0 + a_1 + a_2$ con $a_1 \rightarrow a_2$ no está en 3FN.
- **BCFN: un atributo (no clave alternativa) determina a parte de la primaria.**
 - Si está en 3FN y la clave primaria es simple, está en FNBC.
 - Si $R = a_0 + a_1 + a_2 + a_3$ con $a_2 \rightarrow a_1$ y a_2 no es alternativa, no en BCFN.

3.1. DEPENDENCIAS MULTIVALUADAS.

4FN

Una tabla con una dependencia multivalor es una donde la existencia de dos o más relaciones independientes (muchos a muchos) causa redundancia y es esta redundancia la que suprime la 4FN.

Permutaciones de envíos de pizzas

<u>Restaurante</u>	<u>Variedad de Pizza</u>	<u>Área de envío</u>
Vincenzo's Pizza	Corteza gruesa	Springfield
Vincenzo's Pizza	Corteza gruesa	Shelbyville
Vincenzo's Pizza	Corteza fina	Springfield
Vincenzo's Pizza	Corteza fina	Shelbyville
Elite Pizza	Corteza fina	Capital City
Elite Pizza	Corteza rellena	Capital City
A1 Pizza	Corteza gruesa	Springfield
A1 Pizza	Corteza gruesa	Shelbyville
A1 Pizza	Corteza gruesa	Capital City
A1 Pizza	Corteza rellena	Springfield
A1 Pizza	Corteza rellena	Shelbyville
A1 Pizza	Corteza rellena	Capital City

Ejemplo: como la tabla tiene una clave única y ningún atributo no clave, no viola ninguna forma normal hasta FNBC. Pero debido a que las variedades de pizza que un restaurante ofrece son independientes de las áreas a las cuales el restaurante envía, hay redundancia en la tabla: nos dicen 3 veces que *A1 Pizza* ofrece la *Corteza rellena*, y si *A1 Pizza* comienza a producir pizzas de *Corteza de queso* entonces hay que añadir múltiples filas de *A1 Pizza*, una para cada una de las *Áreas de envío*. En términos formales, esto se describe como que *Variedad de pizza* está teniendo una dependencia multivaluada en *Restaurante*.

3.1. DEPENDENCIAS MULTIVALUADAS.

Para satisfacer la 4FN, debemos poner los hechos sobre las *variedades de pizza* ofrecidas en una tabla diferente de los hechos sobre *áreas de envío*:

Variedades por restaurante

<u>Restaurante</u>	<u>Variedad de pizza</u>
Vincenzo's Pizza	Corteza gruesa
Vincenzo's Pizza	Corteza fina
Elite Pizza	Corteza fina
Elite Pizza	Corteza rellena
A1 Pizza	Corteza gruesa
A1 Pizza	Corteza rellena

Áreas de envío por restaurante

<u>Restaurante</u>	<u>Área de envío</u>
Vincenzo's Pizza	Springfield
Vincenzo's Pizza	Shelbyville
Elite Pizza	Capital City
A1 Pizza	Springfield
A1 Pizza	Shelbyville
A1 Pizza	Capital City

No siempre es deseable la 4FN para BD de tipo OLTP

3.1. DEPENDENCIAS MULTIVALUADAS.

DEFINICIÓN DE DEPENDENCIA MULTIVALUADA

Sea T una tabla donde $T = X + Y + Z$. La dependencia multivaluada $X \twoheadrightarrow Y$ existe en T si los pares de filas $f1$ y $f2$ de T , tal que $f1[X] = f2[X]$ tienen las filas $f3$ y $f4$ en T tales que:

- $f1[X] = f2[X] = f3[X] = f4[X]$
- $f3[Y] = f1[Y]$
- $f3[T-X-Y] = f2[T-X-Y]$
- $f4[Y] = f2[Y]$
- $f4[T-X-Y] = f1[T-X-Y]$

Estudio de 1992 de Margaret S. Wu: la enseñanza se detiene justo antes de la 4FN debido a la creencia de que las tablas que la violan son raras en aplicaciones empresariales. En las BD de 40 empresas, más del 20% tenían tablas que violan 4FN pero satisfacen las anteriores.

En otras palabras se puede decir que $X \twoheadrightarrow Y$ si dado un valor de X , hay un conjunto de valores de Y asociados y este conjunto de valores de Y NO está relacionado (ni funcional ni multifuncionalmente) con los valores de $T - X - Y$ (donde T es el esquema de la tabla), es decir Y es independiente de los atributos de $T-X-Y$.

Una dependencia multivaluada de la forma $X \twoheadrightarrow Y$, es trivial cuando el conjunto de atributos $\{X, Y\}$ conforma el total de los atributos del esquema.

3.1. DEPENDENCIAS MULTIVALUADAS.

DEFINICIÓN DE 4FN

Una tabla T está en 4FN, si está en 3FN o BCFN, y además no tiene dependencias multivaluadas no triviales.

En el ejemplo anterior:

T = restaurante + variedad_pizza + área_envío

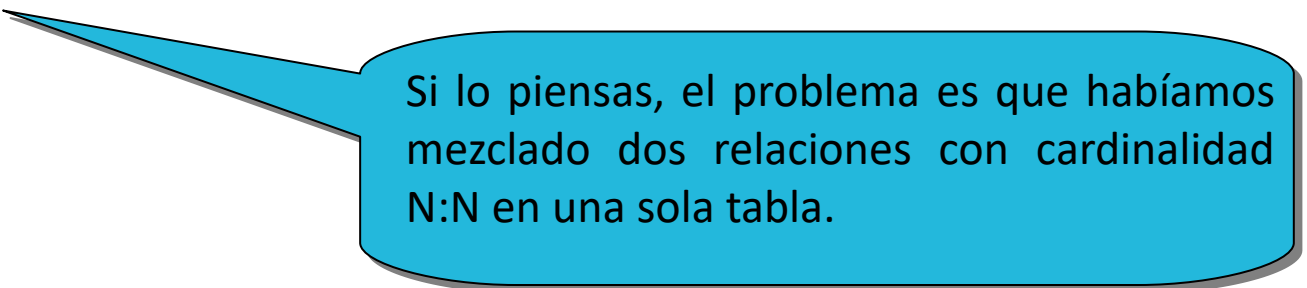
variedad_pizza -->> área_envío

T no está en 4FN aunque sí está en BCFN.

La solución es romper la tabla conservando la dependencia.

T1 = restaurante + variedad_pizza

T2 = restaurante + área_envío



Si lo piensas, el problema es que habíamos mezclado dos relaciones con cardinalidad N:N en una sola tabla.

3.2. PROYECCIÓN-REUNIÓN (PRFN).

LA QUINTA FORMA NORMAL.

Conocida como **forma normal de proyección-reunión**, reduce redundancias en las BD relacionales que guardan hechos multi-valores aislando semánticamente relaciones múltiples relacionadas. Una tabla se dice que está en 5NF si y solo si está en 4FN y cada dependencia de unión (join) en ella es implicada por las llaves candidatas.

Psiquiatra-para-Asegurador-para-Condición

<u>Psiquiatra</u>	<u>Asegurador</u>	<u>Condición</u>
Dr. James	Healthco	Ansiedad
Dr. James	Healthco	Depresión
Dr. Kendrick	FriendlyCare	OCD
Dr. Kendrick	FriendlyCare	Ansiedad
Dr. Kendrick	FriendlyCare	Depresión
Dr. Lowenstein	FriendlyCare	Esquizofrenia
Dr. Lowenstein	Healthco	Ansiedad
Dr. Lowenstein	Healthco	Demencia
Dr. Lowenstein	Victorian Life	Trastorno de conversión

Ejemplo: El psiquiatra puede ofrecer tratamiento reembolsable a los pacientes que sufren de la condición y que son asegurados por el asegurador. En ausencia de cualquier regla que restrinja las combinaciones válidas posibles de psiquiatra, asegurador y condición, la tabla de 3 atributos *Psiquiatra-Asegurador-Condición* es necesaria para modelar la situación correctamente.

3.2. PROYECCIÓN-REUNIÓN (PRFN).

- Sin embargo, **supón que se aplica la regla siguiente:** *Cuando un psiquiatra es autorizado a ofrecer el tratamiento reembolsable a los pacientes asegurados por P y el psiquiatra puede tratar la condición C, entonces -si el asegurador P cubre la condición C- debe ser cierto que el psiquiatra puede ofrecer el tratamiento reembolsable a pacientes que sufren C y están asegurados por P.*

Con estas restricciones es posible dividir la tabla en 3 partes. Esta disposición ayuda a quitar redundancia. Supón que el Dr. Jaime se convierte en un proveedor de tratamientos para la aseguradora Mafriend. En la disposición actual tendríamos que añadir 2 nuevas filas puesto que el Dr. Jaime puede tratar 2 condiciones cubiertas por Mafriend: ansiedad y depresión.

Con la nueva disposición, solo necesitamos agregar una sola fila (en la tabla Psiquiatra-Asegurador).

Psiquiatra-para-Condición	
<u>Psiquiatra</u>	<u>Condición</u>
Dr. James	Ansiedad
Dr. James	Depresión
Dr. Kendrick	OCD
Dr. Kendrick	Ansiedad
Dr. Kendrick	Depresión
Dr. Kendrick	Trastorno emocional
Dr. Lowenstein	Esquizofrenia
Dr. Lowenstein	Ansiedad
Dr. Lowenstein	Demencia
Dr. Lowenstein	Trastorno de conversión

Psiquiatra-para-Asegurador

<u>Psiquiatra</u>	<u>Asegurador</u>
Dr. James	Healthco
Dr. Kendrick	FriendlyCare
Dr. Lowenstein	FriendlyCare
Dr. Lowenstein	Healthco
Dr. Lowenstein	Victorian Life

Asegurador-para-Condición

<u>Asegurador</u>	<u>Condición</u>
Healthco	Ansiedad
Healthco	Depresión
Healthco	Demencia
FriendlyCare	OCD
FriendlyCare	Ansiedad
FriendlyCare	Depresión
FriendlyCare	Trastorno emocional
FriendlyCare	Esquizofrenia
Victorian Life	Trastorno de conversión

3.3. DOMINIO CLAVE (DKFN).

Hay otra clase de restricciones, aún más generales llamadas **formas normales de dominio clave**.

- El problema de estas restricciones es que es difícil de razonar con ellas y no existen reglas de inferencia cerradas y completas.
- Se usan muy raramente aunque alcanzarlas sería lo más.

Ejemplo: el dominio del atributo **persona_rica** consiste en los nombres de toda la gente rica. El dominio para **tipo_rico** es {'Millonario excéntrico', 'Multimillonario excéntrico', 'Millonario malvado', 'Multimillonario malvado'} y el dominio de **valor_net** es todos los números enteros $\geq 1.000.000$. Hay una restricción que liga *tipo_rico* a *valor_net*, incluso aunque no podemos deducir uno del otro. La restricción dicta que un *Millonario excéntrico* o *malvado* tendrá un valor neto de 1.000.000 a 999.999.999, mientras que un *Multimillonario excéntrico* o *malvado* tendrá 1.000.000.000 o más.

Esta restricción no es de dominio ni de clave, y las restricciones de dominio y clave no evitan combinaciones anómala de *tipo_rico* / *valor_net*. La violación de la DKNF se elimina alterando el dominio *tipo_rico* para hacer que sea consistente con solo dos valores, 'Malvado' y 'Excéntrico' (el estatus de persona rica como millonario o multimillonario está implícito en *valor_net*. DKNF es difícil de alcanzar en la práctica.

Persona rica

<u>Persona rica</u>	Tipo de persona rica	Valor neto en dólares
Steve	Millonario excéntrico	124,543,621
Roderick	Multimillonario malvado	6,553,228,893
Katrina	Multimillonario excéntrico	8,829,462,998
Gary	Millonario malvado	495,565,211

4. METODOLOGÍA.

- Hemos asumido que partimos de un esquema *R donde*
 - *R* se puede crear al convertir un diagrama entidad-relación a un conjunto de tablas.
 - *R* podría ser una única relación conteniendo todos los atributos que existirán en la BD (**relación universal**).
 - La Normalización romperá *R* en relaciones más pequeñas.
- *R* podría ser el resultado de un diseño sobre la marcha de relaciones, que queremos comprobar/convertir a una forma normal determinada.

4. METODOLOGÍA.

DIAGRAMA ENTIDAD-RELACIÓN (DER)

- Cuando un DER se hace cuidadosamente, identificando todas las entidades correctamente, las tablas generadas a partir de él, no necesitan normalización.
- Sin embargo, en un diseño real (imperfecto) hay DF's de atributos no clave entre los atributos de cada entidad.
- Las DF's entre atributos de las relaciones del DER son posibles pero raras (la mayoría son relaciones binarias).
- Por tanto, la normalización actúa aquí como una etapa de mejora de ciertas imperfecciones que se nos hayan colado en el diagrama.

4. METODOLOGÍA.

RELACIÓN UNIVERSAL



- La tabla $r_1 \bowtie r_2 \bowtie \dots \bowtie r_n$ es llamada *relación universal* porque tiene a todos los atributos del universo del discurso definidos por

$$R_1 \cup R_2 \cup \dots \cup R_n$$

- La relación universal podría necesitar valores null y tener filas incompletas.
- Una descomposición particular define una forma de restringir la información incompleta que es aceptable para nuestra BD.
 - Las anteriores necesitan al menos un atributo que no sea null.
- La relación universal necesita nombres de atributos únicos asumiendo que tienen **roles únicos**
 - Ejemplo. *Cliente_nombre*, *sucursal_nombre*
- Reutilizar nombres es lo normal en SQL porque hasta los nombres de tablas pueden tener prefijos para eliminar ambigüedades.

4. METODOLOGÍA.

DESNORMALIZAR UN ESQUEMA

- **A veces es interesante usar esquemas poco normalizados por motivos de rendimiento.**
 - **Ejemplo:** mostrar los clientes, sus cuentas y saldos requiere join de cuentas y ahorrador.
 - **Alternativa 1: Usar relaciones sin normalizar** **datawarehouse**
 - Localiza rápido → consultas +rápidas.
 - Consume más almacenamiento y tiempo en modificaciones.
 - Exige más trabajo a programadores y facilita cometer errores (redundancias → inconsistencias y +trabajo).
 - **Alternativa 2: usar vistas materializadas**
 - Ahorra la programación y sus errores, pero tiene las mismas desventajas y ventajas.
 - **Alternativa 3: normalizar.** **OLTP**
 - Lecturas -rápidas.
 - Modificaciones: +rápidas y -almacenamiento.
 - Menos trabajo a programadores y facilita controlar más errores.

4. METODOLOGÍA.

OTROS ASPECTOS DEL DISEÑO

Algunos aspectos de un buen diseño de una BD relacional no los garantiza la normalización, si no el sentido común y la experiencia:

- **Ejemplo de diseño:** En vez de hacer beneficios(*empresa, año, importe*), usar:
 - Las tablas *beneficios_2000, beneficios_2001, etc.*, todas con el mismo esquema (*empresa, importe*).
 - Está en FNBC, pero las consultas de varios años son difíciles y cada año necesita una tabla.
 - *Empresa_año*(*empresa, beneficios_2000, beneficios_2001, ...*)
 - También en FNBC, pero cada año hay que aumentar el nº de atributos.
 - Estaría bien en hojas de cálculo o herramientas de análisis de datos, no en un ambiente de explotación de base de daos relacional.
- **Otro ejemplo:** se necesita rellenar el valor de un atributo con una lista de posibles valores, por ejemplo {*bicicleta, moto, monopatín*}. Posibilidades:
 - **Que lo teclee el usuario:** al final acabarás teniendo valores similares (quieren ser el mismo pero no lo son: *Bicileta, Vicicletas, bicitetas, bici, bicis, BICI, ...*) ¿Si luego necesitas contar cuantos valores bicicleta hay en 30000 filas?
 - **Usar una tabla para los valores:** Gastas una tabla más, pero aseguras que los valores son correctos mediante la integridad referencial. Pesado, pero seguro.