DISEÑO RELACIONAL

Teoría del diseño relacional: formaliza un esquema sin redundancia y capaz de preservar información, a través de las **formas normales.**

Un buen diseño relacional tiene:

- Capacidad de preservar la info
- Redundancia mínima: que haya datos duplicados que llevan a la misma info lleva a ciertos problemas.

Dependencias funcionales

una dependencia funcional X → Y, con X,Y ⊂ A es una restricción sobre las posibles tuplas de R que implica que dos tuplas con igual valor del conjunto de atributos X deben también tener igual valor del conjunto de atributos Y.

la → significa X determina Y. para un mismo X, siempre voy a tener un mismo Y

Tanto X como Y pueden ser 1 atributo o un conjunto de atributos.

Básicamente dice que hay una dependencia funcional entre un conjunto de atributos X y un conjunto de atributos Y (X determina a Y). esta dependencia se da cuando dos tuplas tienen el mismo valor en los atributos de X, también tienen que tener el mismo valor en los atributos de Y.

Ej: Si para dos X iguales, obtengo dos Y distintos, X NO determina Y.

ej: el padrón determina el nombre y apellido, porque si tengo dos filas con el mismo padrón, tengo que tener el mismo nombre y apellido. Ahora al revés NO es lo mismo (podría tener dos alumnos != con el mismo nombre)

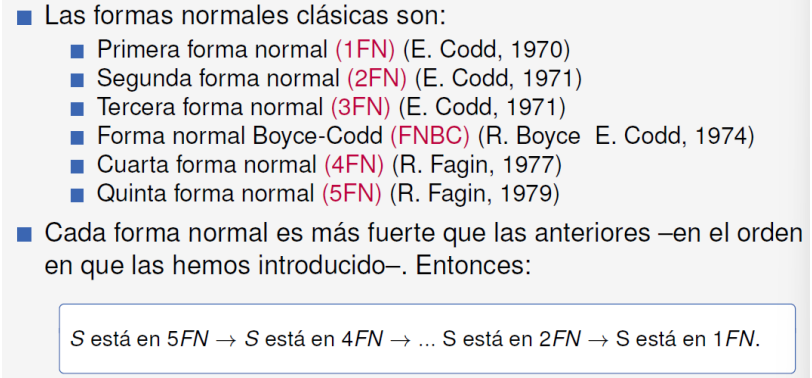
- La dependencia funcional X → Y implica que hay una relación funcional entre los valores de X y los de Y dentro de la base de datos.
- Cuando Y ⊂ X decimos que X → Y es trivial. (el lado derecho esta incluido en el izquierdo)

  Si yo puedo decir que padron determina padron, eso es trivial. Xq si dos filas != tienen el mismo padron obvio q van a tener el mismo padron. Si yo digo nombre+padron determina padron, obviamente es trivial también.

- Las dependencias funcionales se definen a partir de la semántica de los datos. ¡No es posible inferirlas viendo los datos! (osea no puedo definirla con valores actuales de una tabla. Xq por ahí desp yo ingreso algo nuevo y ya no se cumple la dependencia)

Formas normales

Serie de estructuras con las que un esquema de bases de datos puede cumplir o no. Son situaciones en las que esta un esquema de BDD o no. Si las cumple, el esquema están en la X forma normal.



Normalización: proceso a través del cual se convierte un esquema de base de datos en uno equivalente (**i.e: que preserva toda la info**)

Lo que buscamos es:

- Preservar la información
- Eliminar la redundancia
- Evitar las anomalías de ABM

Queremos transformar el esquema en algo con menos redundancia y que evite las anomalías, pero sin perder info.

Ejemplo: la facultad tiene el siguiente esquema relacional donde guarda las notas de los alumnos: ¿Qué problemas le ven a esto?

B	C	D	E	F	G
Padron	**Nombre**	**Apellido**	**Codigo**	**Materia**	**Nota Final**
70000	Juan	Perez	75.15	Base de datos	10
70000	Juan	Perez	75.01	Computación	7
80000	Irma	Gonzaga	75.01	Computación	5
80000	Irma	Gonzaga	75.15	Base de datos	10
90000	Ernesto	Meiren	71.14	Modelos y optimización	8

Anomalías de modificaciones: Si modifico un Nombre, tengo que modificar muchas filas porque están repetidos. Y si me olvido de cambiarlo en otro lado voy a hacer lio. La redundancia no es solo que tengo que hacer una actualización en muchos lados, sino también que: si no la hago en todos los lados, va a estar mal.

Anomalías de altas: No podemos registrar alumnos nuevos de la facu que no hayan cursado nada todavía porque voy a tener que poner un montón de NULLs en materia, y tampoco podría registrar materias sin alumnos.

Anomalías de bajas: Ej: doy de baja modelos y optimización. Si borro esa fila, borro los datos del alumno 9000, cuando lo único que quería registrar era la eliminación de una materia, no la baja de un alumno.

Dependencias funcionales en este ejemplo:

{Padron -> nombre, apellido}; {código->materia}; {padron,código-> nota final}

Primera forma normal (1FN)

Un esquema de BDD está en primera forma normal 1FN cuando los dominios de todos sus atributos solo permiten valores atómicos (indivisibles) y monovaluados

Un valor no atómico seria guardarme la dirección compuesta en vez de calle altura y piso por separado

Actualmente se considera que, en el modelo relacional, todos los atributos deben ser monovaluados y atómicos. Con este criterio, todo esquema relacional ya está en 1FN

Para resolverlo si este no fuera el caso: por ejemplo, puede ser que tenga más de un mail

nombre_profesor	mail
Juan Gómez	{jgomez@udbc.com.ar, jgomez94@mibase.com}
Roberta Casas	{rcasas@udbc.com.ar, rcasas@ggmail.com}
Irene Adler	{iadler@udbc.com.ar}

SOLUCION 1: colocar un mail por tupla y repetir el nombre del profesor

nombre_profesor	mail
Juan Gómez	jgomez@udbc.com.ar
Juan Gómez	jgomez94@mibase.com
Roberta Casas	rcasas@udbc.com.ar
Roberta Casas	rcasas@ggmail.com
Irene Adler	iadler@udbc.com.ar

SOLUCION 2: suponer un máximo posible M de mails y tener M atributos distintos reservados para cada mail. Para los que tienen menos de M mails, quedan valores nulos.

nombre_profesor	mail1	mail2
Juan Gómez	jgomez@udbc.com.ar	jgomez94@mibase.com
Roberta Casas	rcasas@udbc.com.ar	rcasas@ggmail.com
Irene Adler	iadler@udbc.com.ar	NULL

Reglas de inferencia de dependencias funcionales

Son 3 reglas para inferir dependencias funcionales a través de otras:

- Axioma de reflexividad: $Y \subset X \Rightarrow X \rightarrow_I Y$
- Axioma de aumento: $\forall W : X \rightarrow Y \Rightarrow XW \rightarrow YW$
- Axioma de transitividad: $X \rightarrow Y \wedge Y \rightarrow Z \Rightarrow X \rightarrow Z$

Aumento: Si X determina Y, X+AlgoMas determinan a Y+AlgoMas. Infiero una dependencia que no conocía en base a una que ya tenia.

Ej: si padron determina apellido, padron+nombre determina apellido+nombre

Transitividad: si X determina a Y, e Y determina Z entonces para un mismo valor de X determino un mismo valor de Z.

Ejercicio:

Muestre que dado el conjunto de dependencias funcionales $F = \{A \rightarrow C, BC \rightarrow E, D \rightarrow B\}$ es posible inferir que $AD \rightarrow E$.

- Si A -> C , AD -> CD por aumento
- Si D->B, CD -> CB por aumento
- Si AD-> CD y CD -> BC (osea CB) y BC -> E, por transitividad, AD -> E

Para hacer esto menos tedioso, llegaron las reglas de Armstrong;

- Regla de unión: $X \rightarrow Y \wedge X \rightarrow Z \Rightarrow X \rightarrow YZ$
- Regla de pseudotransitividad: $\forall W : X \rightarrow Y \wedge YW \rightarrow Z \Rightarrow XW \rightarrow Z$
- Regla de descomposición: $X \rightarrow YZ \Rightarrow X \rightarrow Y \wedge X \rightarrow Z$

La unión y descomposición SOLO se puede hacer del lado derecho, no del lado izquierdo.

Ej: para padron->nombre, apellido, puedo decir que padron->nombre y padron-> apellido. Pero NO puedo decir que para (lado izquierdo) padron, código -> nota, padron->nota y código-> nota. Porque puedo tener un alumno con mas de una nota.

Esto igualmente sigue siendo engorroso, por lo que se llega a la definición de:

Clausuras de conjuntos de df's y atributos

Si probamos que para: *F = {A → C,BC → E,D → B} es posible inferir que AD → E*, podríamos tener un F2 tal que: F2 = *{A → C,BC → E,D → B, AD → E }.* Si comparamos F1 con F2, a nivel de dependencias funcionales, son lo mismo (representan las mismas restricciones)

Estaría bueno saber si dos conjuntos de dependencias son equivalentes entre si o no. Para eso están las ***Clausuras de conjuntos de df's:*** si parto de un conjunto F, el conjunto de clausuras de F, llamado F_+, son todas las df's que puedan inferirse de F.

Si dos conjuntos F1 y F2 tienen la misma clausura, son equivalentes. Entonces, las clausuras me permiten comparar conjuntos de DFs.

Algoritmo para obtener la clausura de un conjunto de atributos

Me deja ver si hay una dependencia implícita entre dos atributos según otras dependencias funcionales + las reglas que vimos.

Ej: quiero el AD^+_F para *F = {A → C,BC → E,D → B}*

1. Los atributos pueden determinarse a sí mismos. Osea para el ej seguro tengo **AD^+_F = {A, D}**
2. Mira las dependencias funcionales de F, y si el lado izquierdo esta completamente incluido en el conjunto **AD^+**, agrega el lado derecho.
    - Para A->C, A esta completamente en el conjunto **AD^+_F** así que le agrego C
    - Para D -> B, D esta completamente en el conjunto **AD^+_F** así que agrego B

Clausura de conjunto de DF's != Clausura de conjunto de atributos

1. En la 1ra busco todas las dependencias funcionales posibles usando los axiomas de Armstrong
2. En el 2do son todos los atributos que puedo inferir de un conjunto de dependencias funcionales usando los axiomas (lo obtengo con el algoritmo que vimos recién.

Segunda forma normal (2FN)

Tenemos el siguiente ejemplo: (este ya es 1FN porque todas las tuplas tienen valores atómicos)

nombre_dpto	nombre_profesor	asignatura
Física	Juan Gómez	Física II
Física	Roberta Casas	Física II
Física	Juan Gómez	Física III
Matemática	Roberta Casas	Topología
Matemática	Irene Adler	Álgebra I

Las dependencias funcionales detectadas son:

- **Asignatura -> nombre_depto**

Existen otras dependencias funcionales que pueden deducirse de la anterior

- **{nombre_profesor, asignatura} -> nombre_depto**

- **{nombre_profesor, asignatura} -> asignatura (trivial)**

¿Por qué son dependencias? No hay dos filas con el mismo nombre de profesor y nombre de asignatura que pertenezcan a diferentes departamentos, por ejemplo.

Detectamos que la clave primaria es: {nombre_profesor, asignatura}

El nombre_depto no depende de la clave primaria completa, sino que solo de una parte. Es verdad que la clave primaria entera me determina el nombre de depto, pero si yo tengo SOLO asignatura, también puedo determinar el nombre depto: decimos que **la dependencia PK -> nombre_depto** es una **dependencia funcional parcial**

Dependencia funcional parcial:

Una dependencia funcional X -> Y es **parcial** cuando existe un subconjunto propio A ⊂ X,A≠ X para el cual A → Y. Una dependencia funcional X -> Y es completa si y solo si NO es parcial.

nombre_dpto	nombre_profesor	asignatura
Física	Juan Gómez	Física II
Física	Roberta Casas	Física II
Física	Juan Gómez	Física III
Matemática	Roberta Casas	Topología
Matemática	Irene Adler	Álgebra I

- En el ejemplo, nombre_dpto no tiene dependencia funcional completa de la clave primaria {nombre_profesor, asignatura}.

Atributo primo de una relación: Es aquel que es parte de alguna clave candidata de la relación. En nuestro ejemplo, los atributos primos son en nombre_profesor y la asignatura

Una relación está en **segunda forma normal 2FN** cuando **todos sus atributos NO primos tienen dependencia funcional completa de las claves candidatas**

nombre_dpto	nombre_profesor	asignatura
Física	Juan Gómez	Física II
Física	Roberta Casas	Física II
Física	Juan Gómez	Física III
Matemática	Roberta Casas	Topología
Matemática	Irene Adler	Álgebra I

- ¿Cómo resolvemos la situación en el ejemplo?
  - DocenteAsignatura(nombre_profesor, asignatura)
  - AsignaturaDepartamento(asignatura, nombre_dpto)

Para resolverlo, armo otra tabla: en la 2da pongo el nombre de la asignatura y el único dato con el que se me formaba una dependencia funcional.

En esta nueva forma, en la 1ra tabla no existe ningún atributo no primo, y por ende no existe ningún atributo con dependencias parciales. En la 2da tabla tengo un solo atributo no primo, pero tiene una dependencia funcional completa de la clave candidata (asignatura)

Ejemplo 2 de 2FN

Ejemplo: Base de datos de torneos de tenis individual

TENIS

nombre_torneo	año	ciudad	país	tenista1	tenista2	ronda	set	punt1	punt2
Roland Garros	2016	París	Francia	A. Murray	S. Wawrinka	2-final	1	6	4
Roland Garros	2016	París	Francia	A. Murray	S. Wawrinka	2-final	2	6	2
Roland Garros	2016	París	Francia	A. Murray	S. Wawrinka	2-final	3	4	6
Roland Garros	2016	París	Francia	A. Murray	S. Wawrinka	2-final	4	6	2
Masters de Madrid	2015	Madrid	España	R. Federer	R. Nadal	4-final	1	3	6
Masters de Madrid	2015	Madrid	España	R. Federer	R. Nadal	4-final	2	1	6
Roland Garros	2016	París	Francia	N. Djokovic	A. Murray	Final	1	6	3
Roland Garros	2016	París	Francia	N. Djokovic	A. Murray	Final	2	1	6
Roland Garros	2016	París	Francia	N. Djokovic	A. Murray	Final	3	6	2
Roland Garros	2016	París	Francia	N. Djokovic	A. Murray	Final	4	6	4

Hipótesis: Todos los torneos son por eliminación, de manera que 2 tenistas pueden enfrentarse 1 vez como máximo por torneo.

1. Vamos a identificar las dependencias funcionales no triviales a partir de la semántica:

- nombre_torneo → {ciudad, pais}
- {nombre_torneo, año, tenista1, tenista2} → {ronda}
- {nombre_torneo, año, tenista1, ronda} → {tenista2}
- {nombre_torneo, año, tenista2, ronda} → {tenista1}
- {nombre_torneo, año, tenista1, tenista2, set} → {punt1, punt2}
- {nombre_torneo, año, tenista1, ronda, set} → {punt1, punt2}
- {nombre_torneo, año, tenista2, ronda, set} → {punt1, punt2}

2. Identificamos la clave primaria
   {nombre_torneo, año, tenista1, tenista2, set}

3. Identificamos otras claves candidata

   {nombre_torneo, año, tenista1, ronda, set}
   {nombre_torneo, año, tenista2, ronda, set}

4. Busco los Atributos primos: nombre_torneo, año, tenista1, tenista2, set, ronda
5. Infiero los Atributos NO primos: ciudad, país.

6. Me fijo si estoy en 2FN. Para estarlo, los atributos NO primos deben tener dependencias funcionales completas.

En este ej, los atributos no primos ciudad, país tienen una df con nombre_torneo. Así que tienen una df parcial, porque solo dependen de parte de la clave candidata.

7. Resuelvo las DFs parciales realizando una descomposición en dos tablas: paso a otra tabla los atributos NO primos y la parte de la CK de la que dependen.

Hacemos en una tabla el nombre_torneo, ciudad país, y en la otra todos los que estaban menos los de la derecha de la DF (osea los no primos)

Descomposición

Torneos(nombre_torneo, ciudad, país)
Partidos(nombre_torneo, año, tenista1, tenista2, set, ronda, punt1, punt2)

Ejemplo: Base de datos de torneos de tenis individual

TORNEOS

nombre_torneo	ciudad	país
Roland Garros	París	Francia
Masters de Madrid	Madrid	España

PARTIDOS

nombre_torneo	año	tenista1	tenista2	ronda	set	punt1	punt2
Roland Garros	2016	A. Murray	S. Wawrinka	2-final	1	6	4
Roland Garros	2016	A. Murray	S. Wawrinka	2-final	2	6	2
Roland Garros	2016	A. Murray	S. Wawrinka	2-final	3	4	6
Roland Garros	2016	A. Murray	S. Wawrinka	2-final	4	6	2
Masters de Madrid	2015	R. Federer	R. Nadal	4-final	1	3	6
Masters de Madrid	2015	R. Federer	R. Nadal	4-final	2	1	6
Roland Garros	2016	N. Djokovic	A. Murray	Final	1	6	3
Roland Garros	2016	N. Djokovic	A. Murray	Final	2	1	6
Roland Garros	2016	N. Djokovic	A. Murray	Final	3	6	2
Roland Garros	2016	N. Djokovic	A. Murray	Final	4	6	4

DESCOMPOSICION

Teorema: si cuando descompongo la relación en dos, si la intersección de los atributos es superclave de alguna de las relaciones, entonces no hay perdida de info.

1. yo descompongo y me fijo qué atributos quedaron en ambas tablas. (en nuestro caso, es únicamente nombre_torneo)
2. Si nombre_torneo es superclave de alguna de las dos relaciones, entonces no pierdo info.

En este caso, como nombre_torneo es CK para una de las tablas (tabla Torneos), no hay perdida de info, porque al ser CK, en esa tabla esta una única vez la fila por valor.

Definición matemática de descomposición:

Relación universal: una relación R(A1,A2, ...,An) que engloba todos los atributos del mundo real que nuestro modelo lógico representa

Dada una relación universal R(A1,A2, ...,An) y un conjunto de dependencias funcionales F definidas sobre ella, decimos que un conjunto de relaciones

{R1(B11,B12, ...,B1n1), ..., $R_m(B_{m1},B_{m2}, ...,Bmn_m)$} es una descomposición de R cuando todos los atributos de la relación R se conservan. Osea:

$$\bigcup_{i=1}^{n} A_i = \bigcup_{i=1}^{m} \bigcup_{j=1}^{n_i} B_{ij}$$

Entre todos Ri, si yo agarro todos los atributos que hay, y los uno entre todos los Ri, me quedo con los mismos atributos que los originales. Osea:todos los atributos de la relación original, tienen que estar en **al menos un Ri.** A veces están en mas de uno, pero no pueden desaparecer completamente, y no pueden aparecer atributos nuevos.

En nuestro ejemplo, si yo agarro todos los atributos de torneo, y todos los atributos de partido (como es unión nombre_torneo no queda repetido), llego a tener todos los atributos que tenia originalmente.

Propiedades de las descomposiciones

1. **Preservación de información:** de las dos tablas, yo puedo volver a la tabla original con un Join por nombre_torneo y llego a la tabla original sin perder info.

Si una descomposición cumple que, para toda instancia posible de R, la junta de las proyecciones sobre los Ri permite recuperar la misma instancia de relación, entonces decimos que la descomposición preserva la información.

2. **La preservación de dependencias funcionales**: Diremos que la descomposición preserva las dependencias funcionales cuando toda dependencia funcional X → Y en R puede inferirse a partir de dependencias funcionales definidas en los Ri.

Una descomposición de R que cumple con ambas propiedades se denomina descomposición equivalente de R.

A medida que avanzamos en la normalización, se minimiza la **redundancia de datos**, una propiedad deseable en todo esquema de base de datos.

Tercera Forma Normal (3FN)

Ejemplo supermercado

VENTAS

nro_factura	cliente	nro_item	cod_producto	nombre_producto	cantidad	precio_unit
0003-45821	Lionel Pessari	1	249	Suprabond 500mg	2	87.00
0003-45821	Lionel Pessari	2	230	Tersuave azul 4l	1	270.00
0003-45821	Lionel Pessari	3	115	Brocha 5cm	2	90.00
0003-45822	Claudia Serrano	1	258	Alba p/Exteriores 3l	2	225.00
0003-45822	Claudia Serrano	2	116	Brocha 10cm	2	130.00
0003-45823	Claudia Serrano	1	330	Cetol 2l	1	315.00

Lo que SI puede pasar es que haya un nuevo ítem con un código que se halla dado en la factura anterior. Ej: que agarre otro paquete de fideos, se pase como item 4 de Lionel y el cod_prod sea 249.

1. Identificamos las dependencias funcionales no triviales a partir de la semántica:
   - nro_factura → cliente
   - {nro_factura, nro_item} → {nombre_producto, cod_producto, cantidad, precio_unit}
   - cod_producto → nombre_producto

2. Identificamos la clave primaria:
   - {nro_factura, nro_item}

   No puede ser nro_factura, cod_prodicto por lo que digimos de los fideos antes.

3. Busco otras claves candidatas: no hay
4. Esta en 1FN: Si. Porque los atributos son atómicos, etc.
5. ¿Está en 2FN?

Miremos los atributos NO primos: cliente, cod_producto, nombre_producto, cantidad, precio_unit.

Tengo un atributo NO primo: cliente, que depende parcilamente de la CK (lo puedo deducir con nro_factura solo, sin nro_item. Entonces, como tengo una DF parcial de un atributo primo, NO está en 2FN

6. Lo llevo a 2FN

Voy a tener que descomponer en una tabla con (nro_factura,cliente) y otra con todos los otros atributos menos el cliente.

■ Normalización:

Descomposición a 2FN

ClienteFactura(nro_factura, cliente)
DetalleFactura(nro_factura, nro_item, cod_producto, nombre_producto, cantidad, precio_unit)

CLIENTE FACTURA

nro_factura	cliente
0003-45821	Lionel Pessari
0003-45822	Claudia Serrano

DETALLE FACTURA

nro_factura	nro_item	cod_producto	nombre_producto	cantidad	precio_unit
0003-45821	1	249	Suprabond 500mg	2	87.00
0003-45821	2	230	Tersuave azul 4l	1	270.00
0003-45821	3	115	Brocha 5cm	2	90.00
0003-45822	1	258	Alba p/Exteriores 3l	2	225.00
0003-45822	2	116	Brocha 10cm	2	130.00
0003-45823	1	330	Cetol 2l	1	315.00

Obvs: todas las dependencias funcionales que había se mantienen, pero una de las dependencias muestra que **un atributo no primo puede deducirse a partir de otro atributo no primo:**

cod_producto → nombre_producto

entonces, decimos que nombre_producto tiene "dependencia transitiva" en la clave primaria, lo que NO es deseable.

Dependencia transitiva

- Una dependencia funcional $X \rightarrow Y$ es transitiva cuando existe un conjunto de atributos Z que satisface dependencias $X \rightarrow Z$ y $Z \rightarrow Y$, siendo $Z \rightarrow Y$ no trivial, X$\rightarrow$ Y no trivial, y $Z \not\rightarrow X$.

DETALLEFACTURA

nro_factura	nro_item	cod_producto	nombre_producto	cantidad	precio_unit
0003-45821	1	249	Suprabond 500mg	2	87.00
0003-45821	2	230	Tersuave azul 4l	1	270.00
0003-45821	3	115	Brocha 5cm	2	90.00
0003-45822	1	258	Alba p/Exteriores 3l	2	225.00
0003-45822	2	116	Brocha 10cm	2	130.00
0003-45823	1	330	Cetol 2l	1	315.00

- En el ejemplo, *nombre_producto* tiene dependencia transitiva en la clave primaria porque $\{nro\_factura, nro\_item\} \rightarrow cod\_producto$ y $cod\_producto \rightarrow nombre\_producto$.
- Observación: Toda dependencia funcional parcial no trivial es transitiva.

3FN: Decimos que una relación está en tercera forma normal (3FN) cuando no existen dependencias transitivas CKi → Y de atributos no primos (i.e. Y/⊂ Ui CKi), con CKi clave candidata.

Entonces lo que buscamos solucionar en las 3FN es solucionar las dependencias **transitivas.**

Una definicion equivalente es que para toda dependencia funcional no trivial X → Y, o bien X es superclave, o bien Y – X contiene solo atributos primos:

Tenemos el problema cuando el lado izquierdo NO es superclave y el lado derecho NO es primo

Superclave: un conjunto de atributos que permiten identificar unívocamente a las filas, pero NO necesariamente es minimal como la CK

En nuestro ejemplo, el esquema NO esta en 3FN porque cod_producto → nombre_prod viola: el lado izquierdo NO es superclave y el lado derecho es NO primo.

Solución:

1. Una nueva tabla con la que viola la transitidad: en nuestro ejemplo tenemos una nueva tabla con código_prod, nombre_prod

DETALLEFACTURA

nro_factura	nro_item	cod_producto	nombre_producto	cantidad	precio_unit
0003-45821	1	249	Suprabond 500mg	2	87.00
0003-45821	2	230	Tersuave azul 4l	1	270.00
0003-45821	3	115	Brocha 5cm	2	90.00
0003-45822	1	258	Alba p/Exteriores 3l	2	225.00
0003-45822	2	116	Brocha 10cm	2	130.00
0003-45823	1	330	Cetol 2l	1	315.00

- PK = $\{nro\_factura, nro\_item\}$
- ¿Cómo se resuelve la situación?
  - DetalleFactura(nro_factura nro_item cod_producto cantidad precio_unit)
  - Productos(cod_producto, nombre_producto)

Y me queda un esquema de 3 tablas, contando la tabla de antes que era CLIENTE_FACTURA, donde resolví las dependencias parciales y las dependencias transitivas

Volvamos al ejemplo de los tenistas:

PARTIDOS

nombre_torneo	año	tenista1	tenista2	ronda	set	punt1	punt2
Roland Garros	2016	A. Murray	S. Wawrinka	2-final	1	6	4
Roland Garros	2016	A. Murray	S. Wawrinka	2-final	2	6	2
Roland Garros	2016	A. Murray	S. Wawrinka	2-final	3	4	6
Roland Garros	2016	A. Murray	S. Wawrinka	2-final	4	6	2
Masters de Madrid	2015	R. Federer	R. Nadal	4-final	1	3	6
Masters de Madrid	2015	R. Federer	R. Nadal	4-final	2	1	6
Roland Garros	2016	N. Djokovic	A. Murray	Final	1	6	3
Roland Garros	2016	N. Djokovic	A. Murray	Final	2	1	6
Roland Garros	2016	N. Djokovic	A. Murray	Final	3	6	2
Roland Garros	2016	N. Djokovic	A. Murray	Final	4	6	4

¿Hay dependencias transitivas de atributos no primos? No.

- {nombre_torneo, año, tenista1, tenista2, set} → punt1
- A su vez: {nombre_torneo, año, tenista1, tenista2} → ronda
- Y luego: {nombre_torneo, año, tenista1, ronda, set} → punt1
- Pero ronda es parte de una clave candidata.

Por lo tanto, está en tercera formal normal.

Forma normal Boyce-Codd (FNBC)

Pero… hay un tipo de redundancia que aún no eliminamos…

CURSADA

alumno	materia	profesor
Dante Micelli	Zoología	Edmundo Ribeiro
Dante Micelli	Botánica	José Cestoni
Dante Micelli	Anatomía General I	Pedro González
Alberto Deheza	Botánica	José Cestoni
Alberto Deheza	Zoología	Viviana Díaz
Carla Hernández	Zoología	Edmundo Ribeiro
Carla Hernández	Anatomía General I	Pedro González
Carla Hernández	Botánica	José Cestoni
Leticia Humboldt	Botánica	Héctor Larraza
Leticia Humboldt	Zoología	Viviana Díaz

Hipótesis: Cada materia es dictada por muchos profesores, pero un estudiante sólo cursa con uno de ellos. La universidad tiene la restricción de que un profesor sólo puede dictar una materia.

1. Identificamos las dependencias funcionales no triviales a partir de la semántica:

- {alumno, materia} → profesor
- profesor → materia

la redundancia esta en que como un profesor dicta una única materia, por cada fila que tengo a un profesor tengo muchas veces la materia (ejemplo Viviana Diaz con Zoologia)

2. Identificamos la clave primaria:
   - {alumno, materia}
   a. Aunque hay otras claves candidatas:
      - {alumno, profesor}
3. La “materia” podría deducirse con parte de la clave candidata, y sin embargo la estamos repitiendo...

la redundancia está en que como un profesor dicta una única materia, por cada fila que tengo a un profesor tengo muchas veces la materia (ejemplo Viviana Diaz con Zoologia)

pero las dos DFs que marcamos están en 3FN. Pero hay redundancia igual…

La forma normal Boyce-Codd impide que esto suceda prohibiendo que existan dependencias transitivas de una clave candidata, inclusive de atributos primos.

- {alumno, profesor} → profesor → materia

Una relación está en forma normal Boyce-Codd (FNBC) cuando no existen dependencias transitivas CK → Y, con CK clave candidata

Es decir, eliminamos la posibilidad de tener dependencias transitivas X → Y en las que Y es un atributo primo.

Dicho de otra forma, una relación está en FNBC cuando para toda dependencia funcional no trivial X → Y, X es superclave.

El problema que resuelve la FNBC se da cuando en una relación existen varias claves candidatas que se solapan. Si no se solapan, FNBC = 3FN

¿Cómo se resuelve la situación anterior? Descomponiendo. Armo una nueva tabla.

- Inscripciones(alumno, profesor)
- Cursos(materia, profesor)

¡Pero observemos que perdimos la dependencia funcional {alumno, materia} → profesor! Osea tabla por tabla no tengo forma de validar. Esta forma a veces genera pérdidas de DFs.

INSCRIPCIONES

alumno	profesor
Dante Micelli	Edmundo Ribeiro
Dante Micelli	José Cestoni
Dante Micelli	Pedro González
Alberto Deheza	José Cestoni
Alberto Deheza	Viviana Díaz
Carla Hernández	Edmundo Ribeiro
Carla Hernández	Pedro González
Carla Hernández	José Cestoni
Leticia Humboldt	Héctor Larraza
Leticia Humboldt	Viviana Díaz

CURSOS

materia	profesor
Zoología	Edmundo Ribeiro
Botánica	José Cestoni
Anatomía General I	Pedro González
Zoología	Viviana Díaz
Botánica	Héctor Larraza

Ejemplo 2 resolución de dependencias para FNBC:

Ejemplo: Base de datos de torneos de tenis individual

PARTIDOS

nombre_torneo	año	tenista1	tenista2	ronda	set	punt1	punt2
Roland Garros	2016	A. Murray	S. Wawrinka	2-final	1	6	4
Roland Garros	2016	A. Murray	S. Wawrinka	2-final	2	6	2
Roland Garros	2016	A. Murray	S. Wawrinka	2-final	3	4	6
Roland Garros	2016	A. Murray	S. Wawrinka	2-final	4	6	2
Masters de Madrid	2015	R. Federer	R. Nadal	4-final	1	3	6
Masters de Madrid	2015	R. Federer	R. Nadal	4-final	2	1	6
Roland Garros	2016	N. Djokovic	A. Murray	Final	1	6	3
Roland Garros	2016	N. Djokovic	A. Murray	Final	2	1	6
Roland Garros	2016	N. Djokovic	A. Murray	Final	3	6	2
Roland Garros	2016	N. Djokovic	A. Murray	Final	4	6	4

- La "ronda" puede deducirse con sólo una parte de la clave primaria, y sin embargo la estamos repitiendo en cada set.
- La FNBC impide que esto suceda prohibiendo que existan dependencias parciales de una clave candidata, inclusive de atributos primos.
- Una dependencia que nos molesta es {nombre_torneo, año, tenista1, tenista2}→ *ronda*, porque {nombre_torneo, año, tenista1, tenista2} no es superclave.

Lo resolvemos de la siguiente forma:

- Torneos(nombre_torneo, ciudad, país)
- Rondas(nombre_torneo, año, tenista1, tenista2, ronda)
- Partidos(nombre_torneo, año, tenista1, tenista2, set, punt1, punt2)

TORNEOS

nombre_torneo	ciudad	país
Roland Garros	París	Francia
Masters de Madrid	Madrid	España

RONDAS

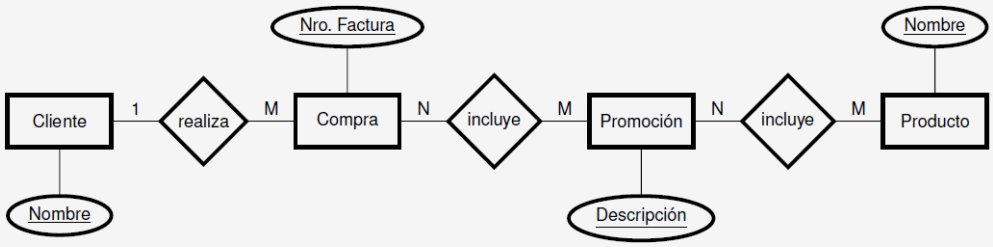
nombre_torneo	año	tenista1	tenista2	ronda
Roland Garros	2016	A. Murray	S. Wawrinka	2-final
Masters de Madrid	2015	R. Federer	R. Nadal	4-final
Roland Garros	2016	N. Djokovic	A. Murray	Final

PARTIDOS

nombre_torneo	año	tenista1	tenista2	set	punt1	punt2
Roland Garros	2016	A. Murray	S. Wawrinka	1	6	4
Roland Garros	2016	A. Murray	S. Wawrinka	2	6	2
Roland Garros	2016	A. Murray	S. Wawrinka	3	4	6
Roland Garros	2016	A. Murray	S. Wawrinka	4	6	2
Masters de Madrid	2015	R. Federer	R. Nadal	1	3	6
Masters de Madrid	2015	R. Federer	R. Nadal	2	1	6
Roland Garros	2016	N. Djokovic	A. Murray	1	6	3
Roland Garros	2016	N. Djokovic	A. Murray	2	1	6
Roland Garros	2016	N. Djokovic	A. Murray	3	6	2
Roland Garros	2016	N. Djokovic	A. Murray	4	6	4

Dependencias Multivaluadas

■ Observemos el siguiente caso de un supermercado:



Hipótesis: Por cada compra sólo se puede adquirir una vez cada promoción.

PROMOCIONES VENDIDAS

nro_factura	nombre_cliente	descripción_promo	nombre_producto
0249-19855	Juana Auzqui	Fiesta-Pancho	Pack salchichas x6
0249-19855	Juana Auzqui	Fiesta-Pancho	Pack pan de viena x6
0249-19855	Juana Auzqui	Fiesta-Pancho	Mayonesa 250gr
0034-20329	Bernardo Lühn	Vajilla Reluciente	Esponjas x2
0034-20329	Bernardo Lühn	Vajilla Reluciente	1 detergente Universo
0034-20329	Bernardo Lühn	Vajilla Reluciente	1 antigrasa Universo
0034-20329	Bernardo Lühn	Vajilla Reluciente	Repasadores x3
0058-91330	Bernardo Lühn	Fiesta-Pancho	Pack salchichas x6
0058-91330	Bernardo Lühn	Fiesta-Pancho	Pack pan de viena x6
0058-91330	Bernardo Lühn	Fiesta-Pancho	Mayonesa 250gr

Dada una relación R(A), la dependencia multivaluada $X \twoheadrightarrow Y$ es una restricción sobre las posibles tuplas de R que implica que para todo par de tuplas t1, t2 tales que t1[X] = t2[X], deberían existir otras dos tuplas t3 y t4 que resulten de intercambiar los valores de Y entre t1 y t2. En otras palabras, tales que:

- t3[X] = t4[X] = t1[X]= t2[X]
- t3[Y] = t1[Y] y t4[Y] = t2[Y]
- $t3[A-(X \cup Y)] = t2[A-(X \cup Y)]$ y $t4[A-(X \cup Y)] = t1[A-(X \cup Y)]$

Clave de la relacion: {nro_factura, descripcion_promo, nombre_producto}.

No es cierto que “nombre_producto” dependa funcionalmente de “descripción_promo”. Sin embargo, tenemos información redundante porque *los productos que integran cada promo son siempre los mismos independientemente de quienes compran la promo.*

Este tipo de redundancia se captura por el concepto de **dependencia multivariada**

Por una cuestión de simetría, si $X \twoheadrightarrow Y$ entonces también vale que $X \twoheadrightarrow A - (X \cup Y)$.

Observemos que en el ejemplo anterior:

- descripcion_promo $\twoheadrightarrow$ nombre_producto
- descripcion_promo $\twoheadrightarrow$ {nombre_factura, nombre_cliente}.

Las dependencias multivaluadas en las que $X \cup Y = A$ ó $Y \subset X$ son triviales.

Cuarta forma normal

Una relación R está en cuarta forma normal cuando para toda dependencia multivaluada no trivial $X \twoheadrightarrow Y$, X es superclave.

Propiedad: si R está en 4FN, entonces R está en FNBC porque:

- Toda dependencia funcional es una dependencia multivaluada: $X \rightarrow Y \Rightarrow X \twoheadrightarrow Y$

- Luego, si un esquema está en 4FN, no puede haber una df no trivial X → Y en la que X no sea superclave

Es común que las dependencias multivaluadas provengan de la existencia de atributos multivaluados en el modelo conceptual, o de interrelaciones N-N no capturadas.

Solución al ejemplo:

1. Normalizamos para llevar a FNBC eliminando la dependencia funcional parcial nro_factura → nombre_cliente.

   Para hacer esto, descomponemos en:

   Descomposición a FNBC

   ClienteFactura(nro_factura, nombre_cliente)
   PromoProdFactura(nro_factura, descripción_promo, nombre_producto)

2. Eliminamos la dependencia multivariada descripcion_promo →→ nombre_producto descomponiendo en:

   Descomposición a 4FN

   Promociones(descripción_promo, nombre_producto)
   ClientesFactura(nro_factura, nombre_cliente)
   PromocionesFactura(nro_factura, descripción_promo)

   Luego:

PROMOCIONES

descripción_promo	nombre_producto
Fiesta-Pancho	Pack salchichas x6
Fiesta-Pancho	Pack pan de viena x6
Fiesta-Pancho	Mayonesa 250gr
Vajilla Reluciente	Esponjas x2
Vajilla Reluciente	1 detergente Universo
Vajilla Reluciente	1 antigrasa Universo
Vajilla Reluciente	Repasadores x3

CLIENTESFACTURA

nro_factura	nombre_cliente
0249-19855	Juana Auzqui
0034-20329	Bernardo Lühn

PROMOCIONESFACTURA

nro_factura	descripción_promo
0249-19855	Fiesta-Pancho
0034-20329	Vajilla Reluciente
0058-91330	Fiesta-Pancho