Bases de Datos

Clase 7: Dependencias y Formas Normales

Buen diseño de relaciones

- Tenemos nuestro E/R
- Lo sabemos transformar a un esquema relacional
- ¿Siempre estará todo perfecto?

Redundancia en los datos

Guardar la información de forma **redundante**, esto es, más de una vez en la base de datos, puede llevar a causar los siguientes problemas:

- **Información redundante**: Ciertos datos se repiten.
- **Anomalías al modificar**: Si se quiere modificar un dato repetido, entonces todas las copias deben modificarse.
- **Anomalías al insertar**: Puede no ser posible insertar ciertos datos a menos que otros datos (repetidos o no relacionados) se inserten también.
- **Anomalías al eliminar**: Puede no ser posible eliminar ciertos datos a menos que otros datos (repetidos o no relacionados) se eliminen también.

Redundancia en los datos

Ejemplo:

Quizás queremos guardar información de:

- Guías de una agencia de turismo
- Necesitamos su id y nombre
- Número de horas que trabajaron
- Los turistas le ponen un score/evaluación
- Pago por hora depende del score

Guías(<u>gid</u>, nombre, score, horas, valorHora)

Redundancia en los datos

Guía

gid	nombre	score	horas	valorHora
1	Juan	8	40	18000
2	Johanna	8	30	18000
3	Cristian	5	30	15000
4	Pedro	8	32	18000

Problemas con la redundancia

1.- Gasto de espacio

Guía

gid	nombre	score	horas	valorHora
1	Juan	8	40	18000
2	Johanna	8	30	18000
3	Cristian	5	30	15000
4	Pedro	8	32	18000

Guardamos el valor 18000 muchas veces

Problemas con la redundancia

2.- Anomalías de inserción

Guía

gid	nombre	score	horas	valorHora
1	Juan	8	40	18000
2	Johanna	8	30	18000
3	Cristian	5	30	15000
4	Pedro	8	32	18000
5	Pablo	9	34	???

Si conocemos el score, pero no el valorHora, no podemos insertar este guía a la tabla

Problemas con la redundancia

3.- Anomalías de actualización

Guía

gid	nombre	score	horas	valorHora
1	Juan	8	40	17000
2	Johanna	8	30	17000
3	Cristian	5	30	15000
4	Pedro	8	32	17000

Digamos que queremos cambiar el valor hora para el score 8

¡Datos inconsistentes!

Hay que cambiarlo en todas las tuplas con score 8…

Problemas con la redundancia

4.- Anomalías de eliminación

Guía

gid	nombre	score	horas	valorHora
1	Juan	8	40	17000
2	Johanna	8	30	17000
~~3~~	~~Cristian~~	~~5~~	~~30~~	~~15000~~
4	Pedro	8	32	17000

Si elimino a Cristian pierdo información

Solución

Descomponer la tabla

Guía

gid	nombre	score	horas	valorHora
1	Juan	8	40	17000
2	Johanna	8	30	17000
3	Cristian	5	30	15000
4	Pedro	8	32	17000

Solución

Descomponer la tabla

Guías

gid	nombre	score	horas
1	Juan	8	40
2	Johanna	8	30
3	Cristian	5	30
4	Pedro	8	32

Valor

score	valorHora
8	18000
5	15000

- **Score** determina **valorHora**

  Esto se llama una **dependencia funcional**

Restricciones de Integridad

Son condiciones aplicadas en un esquema de base de datos para prevenir la entrada de información incorrecta, asegurando así la calidad y legalidad de los datos almacenados.

Estas restricciones se definen durante la creación del esquema de la base de datos y son verificadas por el sistema de gestión de bases de datos (DBMS) para permitir únicamente instancias legales de datos, evitando cualquier tipo de error.

Dependencia Funcional

Una dependencia funcional en la relación R es:

$$X \rightarrow Y$$

Dónde X e Y son conjuntos de atributos de la relación R

$X \rightarrow Y$ es válida en una relación ⬅➡ para toda tupla se tiene:

$$\pi_X(t_1) = \pi_X(t_2) \quad \rightarrow \quad \pi_Y(t_1) = \pi_Y(t_2)$$

Dependencia Funcional

Ejemplo

Guías(gid, nombre, score, horas, valorHora)

- score → valorHora

Personas(rut, nombre)

- rut → nombre

Películas(pid, título, año, director)

- título, año → director

Lado izquierdo no es necesariamente una llave

Dependencia Funcional

Ejemplo

Programación(cine, película, director, dirección, teléfono, horario, precio)

- cine → teléfono, dirección
- película → director
- cine, película, horario → precio

¿Cuál va a ser la llave?

cine, película, horario

Buenas y malas dependencias

DJE

Depto	Jefe	Empleado
D1	Pérez	Ureta
D1	Pérez	Assad
D2	Correa	Vargas
D1	Pérez	Gómez
D1	Pérez	Camus
...	...	...

ES

Empleado	Salario
Ureta	600
Assad	800
Vargas	800
…	…

- **DJE**: Depto → Jefe

Una dependencia buena

- **ES**: Empleado → Salario (Empleado es llave)

Buenas y malas dependencias

Redundancia

DJE

Depto	Jefe	Empleado
D1	Pérez	Ureta
D1	Pérez	Assad
D2	Correa	Vargas
D1	Pérez	Gómez
D1	Pérez	Camus
...	...	...

Tenemos dos tuplas indicando que Pérez es jefe de D1

Buenas y malas dependencias

Anomalía de inserción

DJE

Depto	Jefe	Empleado
D1	Pérez	Ureta
D1	Pérez	Assad
D2	Correa	Vargas
D1	Pérez	Gómez
D1	Pérez	Camus
...	...	...

Compañía contrata a un empleado, pero no lo asigna a un departamento

No podemos almacenarlo en **DJE**

Buenas y malas dependencias

Anomalía de eliminación

DJE

Depto	Jefe	Empleado
D1	Pérez	Ureta
D1	Pérez	Assad
~~D2~~	~~Correa~~	~~Vargas~~
D1	Pérez	Gómez
D1	Pérez	Camus
...	...	...

El empleado Vargas abandona la empresa, por lo que hay que eliminarlo de **DJE**

¡Al hacer eso eliminamos también al jefe Correa!

Buenas y malas dependencias

El problema es que la asociación entre jefes y empleados se almacena en la misma tabla que la asociación entre jefes y departamentos

También el mismo hecho puede ser almacenado muchas veces, como que jefe está a cargo de qué departamento (ej. Pérez con D1)

Buenas y malas dependencias

Existe dependencia Depto → Jefe
pero Depto **no es llave**

¡Este tipo de situaciones queremos evitar!

¡Porque pueden causar **anomalías**!

Anomalías

Objetivo: Eliminar anomalías tratando de minimizar la redundancia, para esto:

- Debemos averiguar las dependencias que aplican
- Descomponer las tablas en tablas más pequeñas

Anomalías

Objetivo: Eliminar anomalías tratando de minimizar la redundancia, para esto:

- **Debemos averiguar las dependencias que aplican**
- Descomponer las tablas en tablas más pequeñas

Dependencias

Observación 1

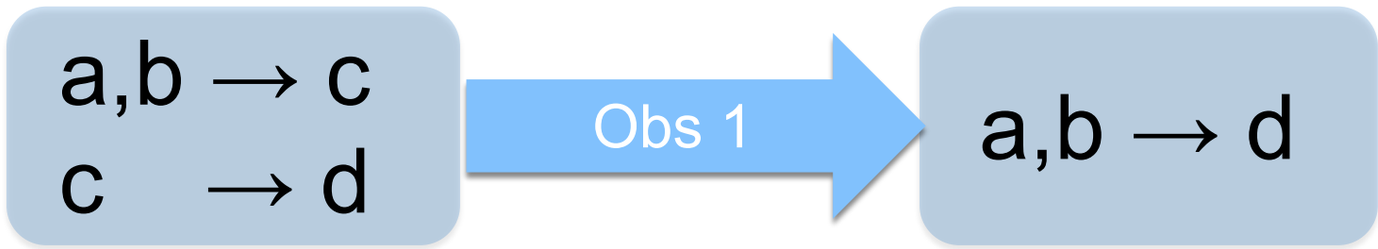
Si tenemos las dependencias:

- $X \rightarrow Y$
- $Y \rightarrow Z$

Podemos deducir que:

- $X \rightarrow Z$

Con X, Y, Z conjuntos de atributos R(a,b,c,d)



Dependencias

Observación 2

Si **Z** ⊆ **Y**, y tenemos la dependencia:

- X → Y

Podemos deducir que:

- X → Z

Con X, Y, Z conjuntos de atributos

R(a,b,c,d)



Dependencias

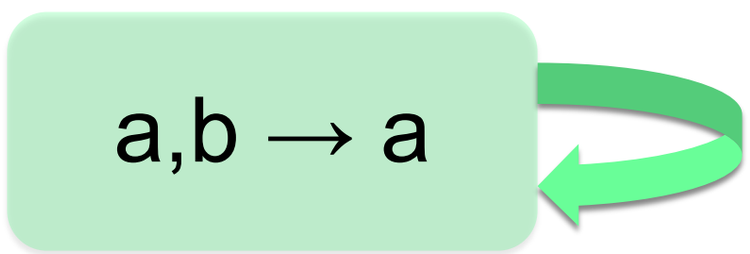
Observación 3

Llamamos dependencia trivial a la dependencia:

$$X \rightarrow Y$$

Si se tiene que **Y** ⊆ **X**

R(a,b,c,d)



Dependencias

Observación 4

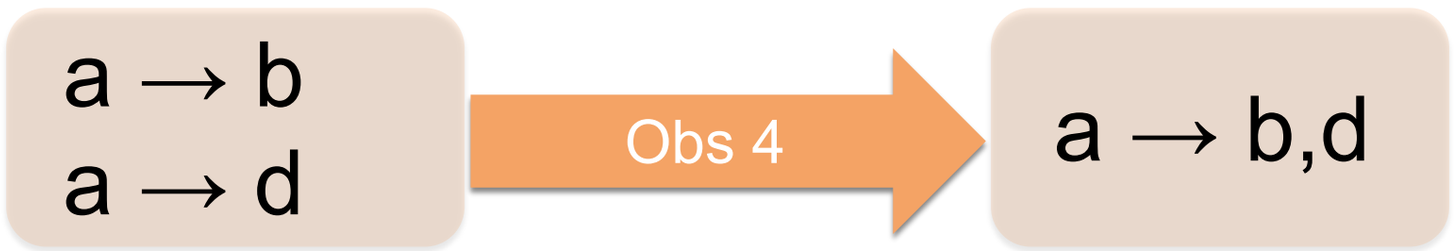
Si tenemos que:

$$X \rightarrow Y, X \rightarrow Z$$

Podemos decir que:

$$X \rightarrow Y, Z$$

R(a,b,c,d)



Dependencias

Observación 5

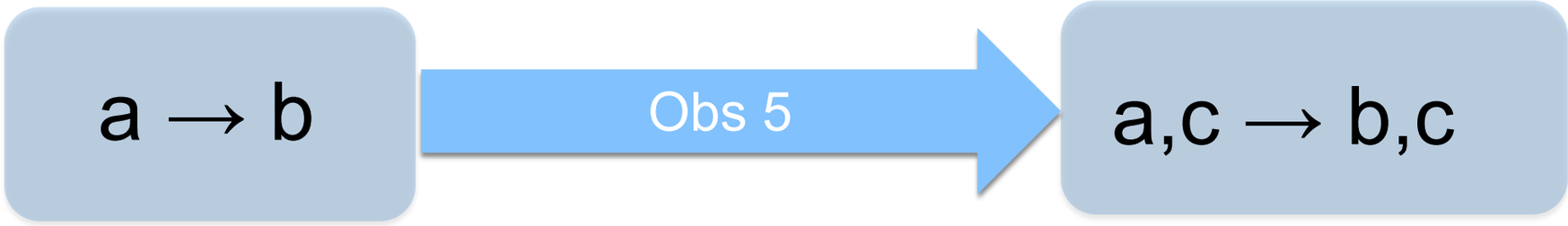
Si tenemos que:

$$X \rightarrow Y$$

Podemos decir que para cada Z:

$$X, Z \rightarrow Y, Z$$

R(a,b,c,d)



Dependencias

Observación 6

Si tenemos X → Y, los atributos **X** son (candidatos a) llave si **Y** contiene a todos los atributos que son parte de la relación y no están en **X**

R(a,b,c,d)



Observación 6: A,B es llave candidata

Dependencias

Ejemplo

Averiguar todas las dependencias de R(a, b, c) si:

- a → b
- b → a, c

Podemos inferir además a → b, c
(por lo tanto, **a** es llave)

Dependencias

Demostración

Supongo que para tuplas t_1, t_2 tengo

$$\pi_a(t_1) = \pi_a(t_2)$$

Como tengo a $\rightarrow$ b, se cumple que $\pi_b(t_1) = \pi_b(t_2)$

Pero b $\rightarrow$ a, c, luego $\pi_{a,c}(t_1) = \pi_{a,c}(t_2)$

Finalmente $\pi_c(t_1) = \pi_c(t_2)$

Importante: con esta idea podemos demostrar que nuestras observaciones 1 - 6 son correctas

Dependencias

Ejercicio

Averiguar todas las dependencias:

Toma(alumno, carrera, ramo, sala, hora)

- alumno → carrera
- carrera, ramo → sala
- ramo → hora

Dependencias

Ejercicio

Averiguar todas las dependencias:

Toma(alumno, carrera, ramo, sala, hora)

- alumno → carrera
- carrera, ramo → sala
- ramo → hora

alumno, ramo → carrera, sala, hora

Anomalías

Objetivo: Eliminar anomalías tratando de minimizar la redundancia, para esto:

- Debemos averiguar las dependencias que aplican
- **Descomponer las tablas en tablas más pequeñas**

Descomposición

Ejemplo: mal diseño

Información: cine, película, director, dirección, teléfono, horario, precio

- cine → dirección, teléfono
- película → director
- cine, película, horario → precio

El peor diseño:

MAL(cine, película, director, dirección, teléfono, horario, precio)

Descomposición

Ejemplo: mal diseño

MAL(cine, película, director, dirección, teléfono, horario, precio)

Redundancia:

- La película determina al director, pero cada vez que dan la película los listamos a ambos
- Listamos la dirección y el teléfono del cine una y otra vez

Descomposición

Ejemplo: mal diseño

MAL(cine, película, director, dirección, teléfono, horario, precio)

Anomalías:

- Si cambiamos una dirección nos volvemos inconsistentes, hay que cambiarla en todas las tuplas
- Si dejamos de mostrar una película perdemos la asociación director – película
- No podemos agregar películas que no se muestran

Descomposición

Ejemplo: buen diseño

Dividimos MAL en 3 tablas

Rels:	Atributos	Dependencias
R_1	cine, dirección, teléfono	cine $\rightarrow$ dirección, teléfono
R_2	cine, pelicula, horario, precio	cine, pelicula, horario $\rightarrow$ precio
R_3	pelicula, director	pelicula $\rightarrow$ director

Descomposición

Ejemplo: buen diseño

Es un buen diseño porque:

- No hay anomalías
- No perdemos dependencias funcionales, pues todas están restringidas a sus respectivas tablas
- No perdemos información:

$$R_1 = \pi_{cine,direccion,telefono}(MAL)$$
$$R_2 = \pi_{cine,pelicula,horario,precio}(MAL)$$
$$R_3 = \pi_{pelicula,director}(MAL)$$
$$MAL = R_1 \bowtie R_2 \bowtie R_3$$

¿Cómo lograr esto para un esquema complejo?

Formas Normales

1NF

Una relación **R** está en **1NF** si y solo si ninguno de sus atributos tiene relaciones como elementos.

Además, cada registro debe ser único, lo que se asegura mediante la presencia de una llave primaria

Ejemplo de relación que NO está en 1NF:

ID cliente	Cliente	Transacciones
1	Isaac	**ID** / **Fecha** / **Cantidad**: 1 / 2003-10-14 / 100.000

1NF

Una relación **R** está en **1NF** si y solo si ninguno de sus atributos tiene relaciones como elementos.

Ejemplo de relación que SI está en 1NF:

ID cliente	Cliente
1	Isaac

ID cliente	ID transaccion	Fecha	Cantidad
1	1	2003-10-14	100.000

2NF

Una relación **R** está en **2NF** si y solo si está en 1NF y, además, si ninguna llave candidata es funcionalmente dependiente (o parte del set) de otra llave candidata. Básicamente, la relación no tiene dependencias parciales.

Ejemplo de relación que NO está en 2NF:

Fabricante	Modelo	País Fabricante
Forte	X-Prime	Italia
Forte	Ultraclean	Italia
Dent-o-Fresh	EZbrush	USA
Brushmaster	SuperBrush	USA
Kobayashi	ST-60	Japón
Hoch	Toothmaster	Alemania
Hoch	X-Prime	Alemania

1. País fabricante depende funcionalmente de Fabricante.

2. Fabricante es parte (subset) de la llave candidata {Fabricante, Modelo}

En resumen, como País Fabricante es una llave candidata funcionalmente dependiente de un subset de una llave candidata, no se sigue la segunda forma normal.

2NF

Una relación **R** está en **2NF** si y solo si está en 1NF y, además, si ninguna llave candidata es dependiente de otra llave candidata. Básicamente, la relación no tiene dependencias parciales.

Ejemplo de relación que SI está en 2NF:

Fabricante	Modelo
Forte	X-Prime
Forte	Ultraclean
Dent-o-Fresh	EZbrush
Brushmaster	SuperBrush
Kobayashi	ST-60
Hoch	Toothmaster
Hoch	X-Prime

Fabricante	País Fabricante
Forte	Italia
Dent-o-Fresh	USA
Brushmaster	USA
Kobayashi	Japón
Hoch	Alemania

3NF

Una relación **R** está en **3NF** si para toda dependencia funcional no trivial X → Y, **X** es una superllave o **Y** es parte de una llave minimal.

Z es llave minimal si no existe llave **Z'** tal que **Z'** ⊆ **Z**

3NF es menos restrictivo que BCNF ya que permite un poco más de redundancia

3NF

Algoritmo (idea)

¡Solo para una tabla!

IDEA: lado izquierdo mínimal; ¡remover dependencias que se pueden deducir!

INPUT:
- el esquema de la tabla **R**
- dependencias funcionales en un formato estandarizado
- sin dependencias inútiles (no lo especificamos bien)

IDEA: esto me asegura las dependencias! (Cada una en su tabla)

1. Para **cada** dependencia funcional $X \rightarrow Y$ crear una tabla con esquema $X \cup Y$
2. Si al final, los esquemas resultantes $R_1, \ldots, R_n$ no contienen una llave de la tabla **R**, agregar una.

IDEA: esto me asegura que no hay pérdida de información!

3NF

Ejemplo

1. Para cada df $X \rightarrow Y$ crear una tabla con esquema $X \cup Y$

Nombre Comuna	Región	Tasa x m2	RUT	Nombre	Apellido	Rol Lote	M2	Avalúo
A	I	2	111111	Claudio	Gonzalez	34	455	960
A	I	2	111111	Claudio	Gonzalez	35	570	1040
A	I	2	222222	Maria	Zapata	27	895	1790
B	III	1,1	111111	Claudio	Gonzalez	10	150	165
B	III	1,1	333333	Carlos	Fernandez	11	200	220
B	X	1,1	444444	Elena	Abarca	13	150	165
C	V	0,5	555555	Luisa	Muñoz	2	500	250
D	V	3,5	111111	Claudio	Gonzalez	11	100	350
	…	…	…	…	…	…	…	…

NombreComuna, Region → Tasa
RUT → Nombre, Apellido
NombreComuna, Region, Rol → RUT, M2
Tasa, m2 → Avalúo



{NombreComuna, Region, Rol }

3NF

Ejemplo

1. Para cada df $X \rightarrow Y$ crear una tabla con esquema X U Y

Nombre Comuna	**Región**	**Tasa x m2**
A	I	2
A	I	2
B	III	1,1
B	X	1,1
C	V	0,5
	…	…

NombreComuna, Region → Tasa

RUT → Nombre, Apellido
NombreComuna, Region, Rol → RUT, M2
Tasa, m2 → Avalúo



{NombreComuna, Region, Rol }

3NF

Ejemplo

1. Para cada df $X \rightarrow Y$ crear una tabla con esquema $X \cup Y$

Nombre Comuna	Región	Tasa x m2
A	I	2
A	I	2
B	III	1,1
B	X	1,1
C	V	0,5
	…	…

NombreComuna, Region → Tasa

RUT	Nombre	Apellido
111111	Claudio	Gonzalez
222222	Maria	Zapata
333333	Carlos	Fernandez
…	…	…

RUT → Nombre, Apellido

NombreComuna, Region, Rol → RUT, M2

Tasa, m2 → Avalúo



{NombreComuna, Region, Rol }

3NF

Ejemplo

1. Para cada df $X \rightarrow Y$ crear una tabla con esquema $X \cup Y$

Nombre Comuna	**Región**	**Tasa x m2**
A	I	2
A	I	2
B	III	1,1
B	X	1,1
C	V	0,5
	…	…

NombreComuna, Region → Tasa

RUT	**Nombre**	**Apellido**
111111	Claudio	Gonzalez
222222	Maria	Zapata
333333	Carlos	Fernandez
…	…	…

RUT → Nombre, Apellido

Tasa x m2	**M2**	**Avaluo**
34	455	960
35	570	1040
27	895	1790
…	…	…

Tasa, m2 → Avalúo

NombreComuna, Region, Rol → RUT, M2



{NombreComuna, Region, Rol }

3NF

Ejemplo

1. Para cada df $X \rightarrow Y$ crear una tabla con esquema $X \cup Y$

Nombre Comuna	Región	Tasa x m2
A	I	2
A	I	2
B	III	1,1
B	X	1,1
C	V	0,5
	…	…

NombreComuna, Region → Tasa

RUT	Nombre	Apellido
111111	Claudio	Gonzalez
222222	Maria	Zapata
333333	Carlos	Fernandez
…	…	…

RUT → Nombre, Apellido

Tasa x m2	M2	Avaluo
34	455	960
35	570	1040
27	895	1790
…	…	…

Tasa, m2 → Avalúo

Nombre Comuna	Región	RUT	Rol Lote	M2
A	I	111111	34	455
	…	…	…	…

NombreComuna, Region, Rol → RUT, M2



{NombreComuna, Region, Rol }

3NF

Ejemplo

2. Si al final, los esquemas resultantes R_1,…, R_n no contienen una llave de la tabla original, agregar una.

Nombre Comuna	**Región**	**Tasa x m2**
A	I	2
A	I	2
B	III	1,1
B	X	1,1
C	V	0,5
	…	…

NombreComuna, Region → Tasa

RUT	**Nombre**	**Apellido**
111111	Claudio	Gonzalez
222222	Maria	Zapata
333333	Carlos	Fernandez
…	…	…

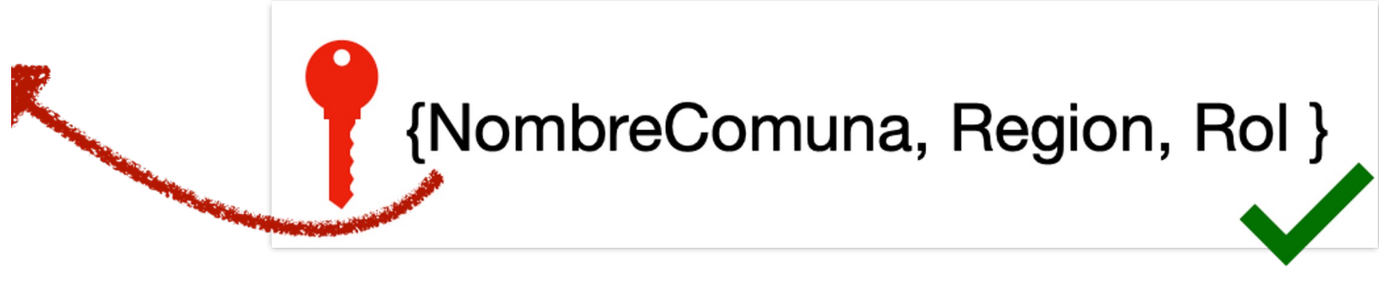
RUT → Nombre, Apellido

Tasa x m2	**M2**	**Avaluo**
34	455	960
35	570	1040
27	895	1790
…	…	…

Tasa, m2 → Avalúo

Nombre Comuna	**Región**	**RUT**	**Rol Lote**	**M2**
A	I	111111	34	455
	…	…	…	…

NombreComuna, Region, Rol → RUT, M2



Relación BCNF

Boyce-Codd Normal Form (BCNF)

Causa de anomalías: $X \rightarrow Y$ cuando X no es una super llave

Una relación **R** está en **BCNF** si para toda dependencia funcional no trivial $X \rightarrow Y$, **X** es una super llave

Un esquema está en BCNF si todas sus relaciones están en BCNF

¿Cuándo estoy en BCNF?

Alumnos(<u>aid</u>, rut, nombre, apellido, carrera, correo)

Llaves:

- **aid**
- **rut**
- **correo**

Dependencias:

- **aid -> todo**
- **rut -> todo**
- **correo -> todo**
- **aid,rut -> todo**
- **aid,nombre -> todo**

…

superllaves

¡Basta concentrarnos en las llaves! -- ¿Por qué?

¿Cuándo estoy en BCNF?

Para BCNF hay que tomar en cuenta las dependencias:

- llave → resto (ojo: llave candidata, posiblemente más de una)
- notSuperLlave → algo (ya no estoy en BCNF)

AlumnoTelefono(<u>rut</u>, <u>teléfono</u>, nombre, apellido):

- rut, telefono -> todo (esto es la única llave, todo OK)

- rut -> nombre,apellido (como rut no es superLlave, entonces el esquema no está en BCNF)

BCNF

¿Cómo lograr BCNF?

BCNF se logra mediante descomposiciones

Ya vimos una: de MAL a tres tablas

BCNF

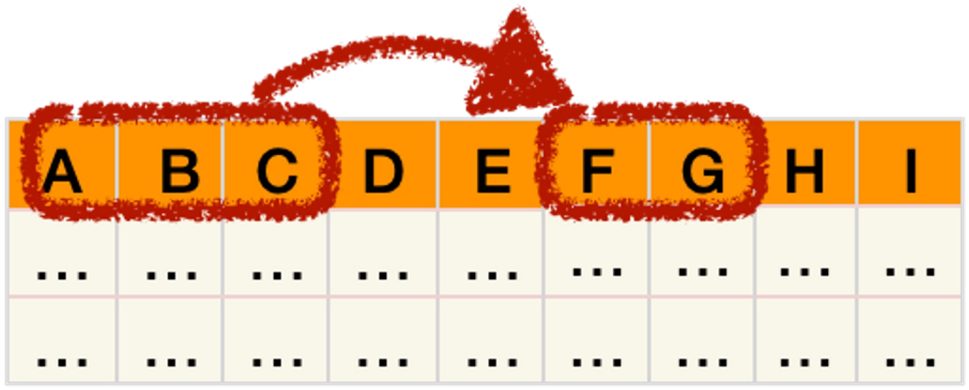
Algoritmo

INPUT: - el esquema de la tabla R(A) con A un conjunto de atributos

- dependencias funcionales estandarizadas

1. Tomar una dependencia funcional $X \rightarrow Y$, dónde X no es una super llave y descomponer en 2 tablas con esquemas:

R1(X,Y) y R2(A – Y)



$A,B,C \rightarrow F,G$

BCNF

Algoritmo

INPUT: - el esquema de la tabla R(A) con A un conjunto de atributos

- dependencias funcionales estandarizadas

1. Tomar una dependencia funcional $X \rightarrow Y$, dónde X no es una super llave y descomponer en 2 tablas con esquemas:

R1(X,Y) y R2(A – Y)

A	B	C	F	G
…	…	…	…	…
…	…	…	…	…

A	B	C	D	E	H	I
…	…	…	…	…	…	…
…	…	…	…	…	…	…

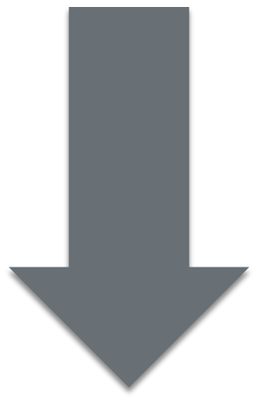
2. Repetir con las tablas que aun no están en BCNF

BCNF

Algoritmo

S(a,b,c,d,e,f)

a,b → c,e rompe BCNF



S1(a,b,c,e)

S2(a,b,d,f)

BCNF

Algoritmo

$x_1,\ldots,x_n \rightarrow y_1,\ldots, y_k$

Se repite el proceso anterior hasta que no haya más violaciones de BCNF

Optimización: elegir la mayor cantidad de y’s posibles

BCNF

Ejemplo

¿Cómo descomponemos **R** para lograr BCNF?

R(a, b, c, d)

$a \rightarrow b$, $b \rightarrow c$

Podemos deducir $a \rightarrow c$, $a \rightarrow b$, c

BCNF

Ejemplo

Descomposición 1:

R(a, b, c, d)

- a → b
- b → c
- a → c
- a → b, c

R(a, b, c, d)

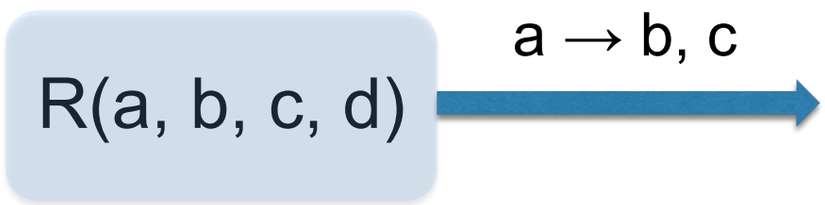
BCNF

Ejemplo

Descomposición 1:

R(a, b, c, d)

- $a \rightarrow b$
- $b \rightarrow c$
- $a \rightarrow c$
- $a \rightarrow b, c$



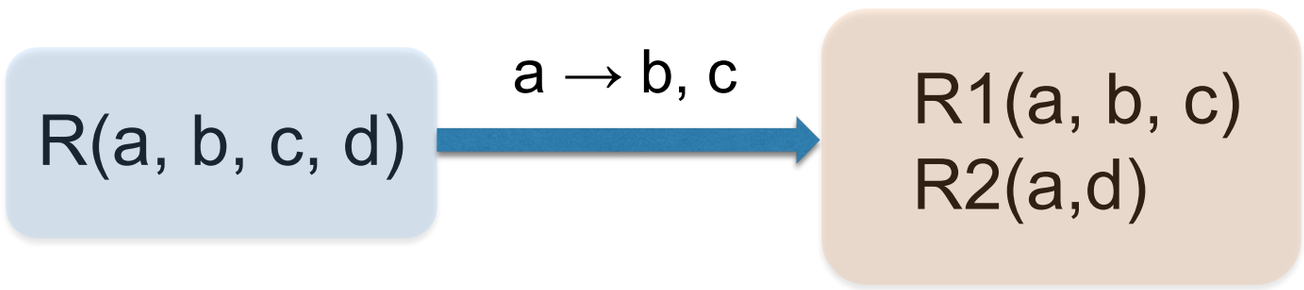
BCNF

Ejemplo

Descomposición 1:

R(a, b, c, d)

- a → b
- b → c
- a → c
- a → b, c



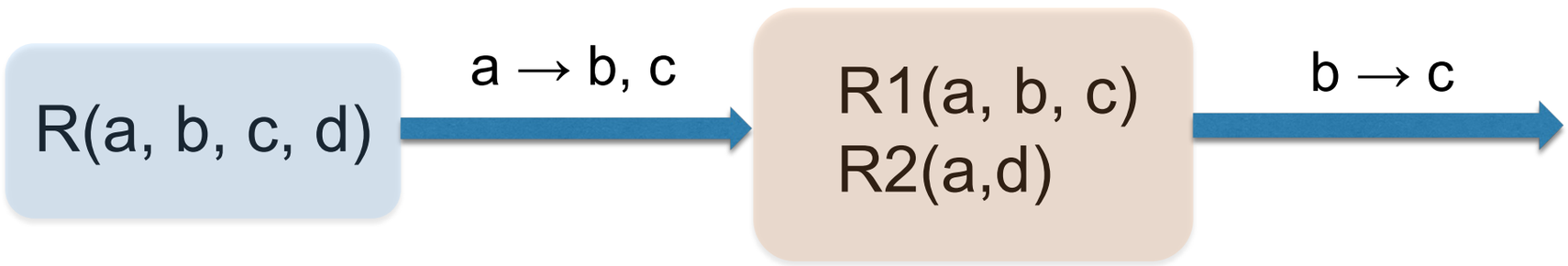
BCNF

Ejemplo

Descomposición 1:

R(a, b, c, d)

- a → b
- b → c
- a → c
- a → b, c



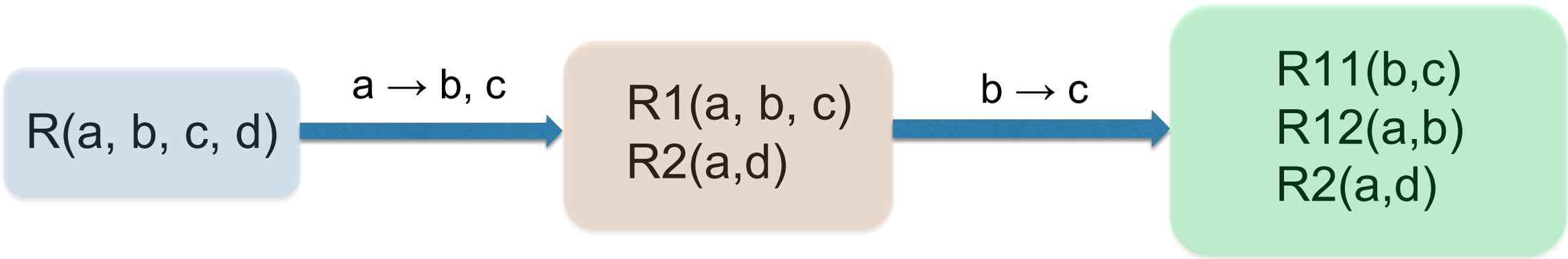
BCNF

Ejemplo

Descomposición 1:

R(a, b, c, d)

- a → b
- b → c
- a → c
- a → b, c



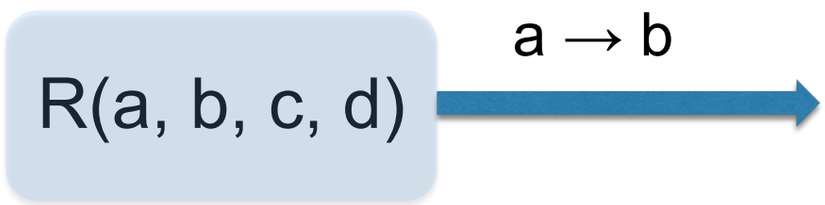
BCNF

Ejemplo

Descomposición 2:

R(a, b, c, d)

- $a \rightarrow b$
- $b \rightarrow c$
- $a \rightarrow c$
- $a \rightarrow b, c$



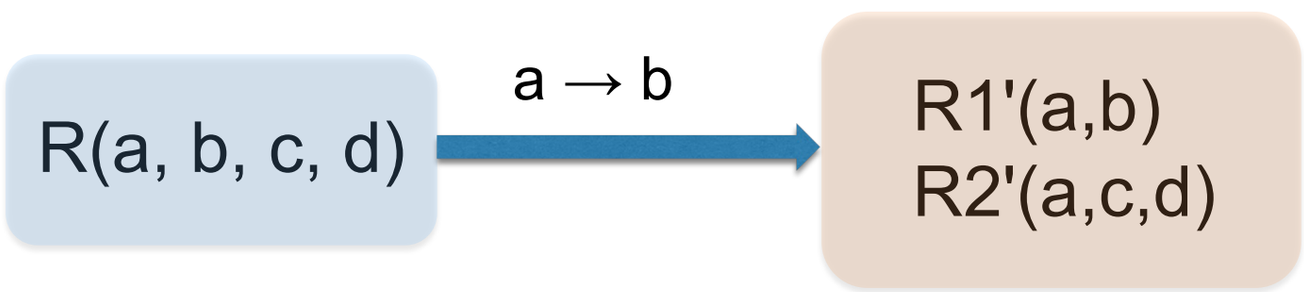
BCNF

Ejemplo

Descomposición 2:

R(a, b, c, d)

- a → b
- b → c
- a → c
- a → b, c



BCNF

Ejemplo

Descomposición 2:

R(a, b, c, d)

- a → b
- b → c
- a → c
- a → b, c



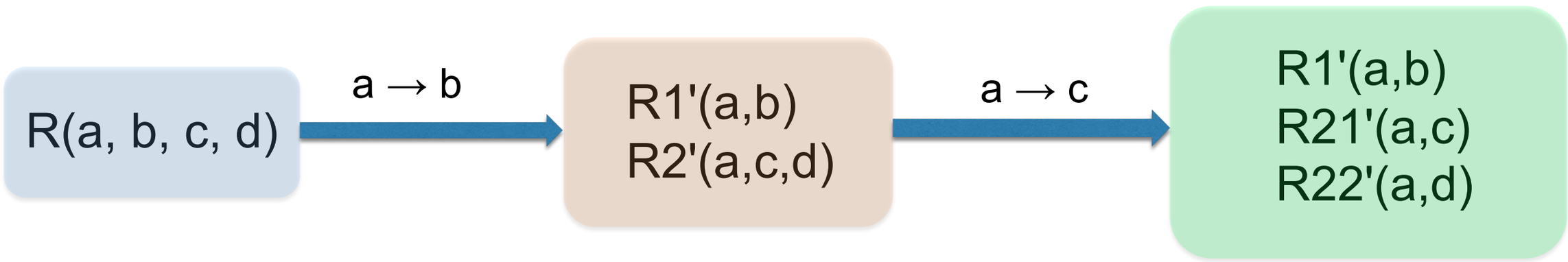
BCNF

Ejemplo

Descomposición 2:

R(a, b, c, d)

- a → b
- b → c
- a → c
- a → b, c



BCNF

Ejemplo

Descomposición 1:

R11(b,c)
R12(a,b)
R2(a,d)

Descomposición 2:

R1'(a,b)
R21'(a,c)
R22'(a,d)

BCNF

Ejemplo

Descomposición 1:

R11(b,c)
R12(a,b)
R2(a,d)

Descomposición 2:

R1'(a,b)
R21'(a,c)
R22'(a,d)

Pero: el join natural es equivalente

Pérdida de Información

¡La descomposición no puede perder información!

Producto	nombre	precio	categoría
	Canon T3	300	fotografía
	Nokia 5000	400	fotografía
	Galaxy IV	400	celular

nombre	categoría
Canon T3	fotografía
Nokia 5000	fotografía
Galaxy IV	celular

precio	categoría
300	fotografía
400	fotografía
400	celular

Pérdida de Información

¡La descomposición no puede perder información!

Al hacer el Join:

Producto	nombre	precio	categoría
	Canon T3	300	fotografía
	Canon T3	400	fotografía
	Nokia 5000	300	fotografía
	Nokia 5000	400	fotografía
	Galaxy IV	400	celular

Descomposición sin pérdida

R(A, B, C) descompuesta en R1(A, B) y R2(A, C) es sin pérdida de información si para toda instancia de R:

$$R_1 \bowtie R_2 = R$$

Descomposición sin pérdida

Teorema

Para todo esquema con relación **R**(X, Y, Z) y dependencia funcional X → Y, para X, Y, Z conjuntos de atributos disjuntos, se tiene que la descomposición en **R1**(X, Y) y **R2**(X, Z) con X → Y es **sin pérdida de información**

Problemas con BCNF

Nuestra descomposición siempre va a ser sin pérdida de información, sin embargo puede ocurrir lo siguiente:

R(a,b,c,d,e)

- DF1: a,b,c → d,e
- DF2: d,e → b

llaves:

- a,b,c
- d,e,a,c

¡Hay una violación de BCNF: d,e → b!

Problemas con BCNF

¡Hay que descomponer usando DF2!

R(a,b,c,d,e)

- DF1: a,b,c → d,e
- DF2: d,e → b



R1(d,e,b)

R2(a,c,d,e)

Observaciones:

- Tabla R1 permite validar que se cumple DF2
- ¡No hay una tabla que permite validar DF1!

Problemas con BCNF

¡Hay que descomponer usando DF2!

R(a,b,c,d,e)

- DF1: a,b,c → d,e
- DF2: d,e → b



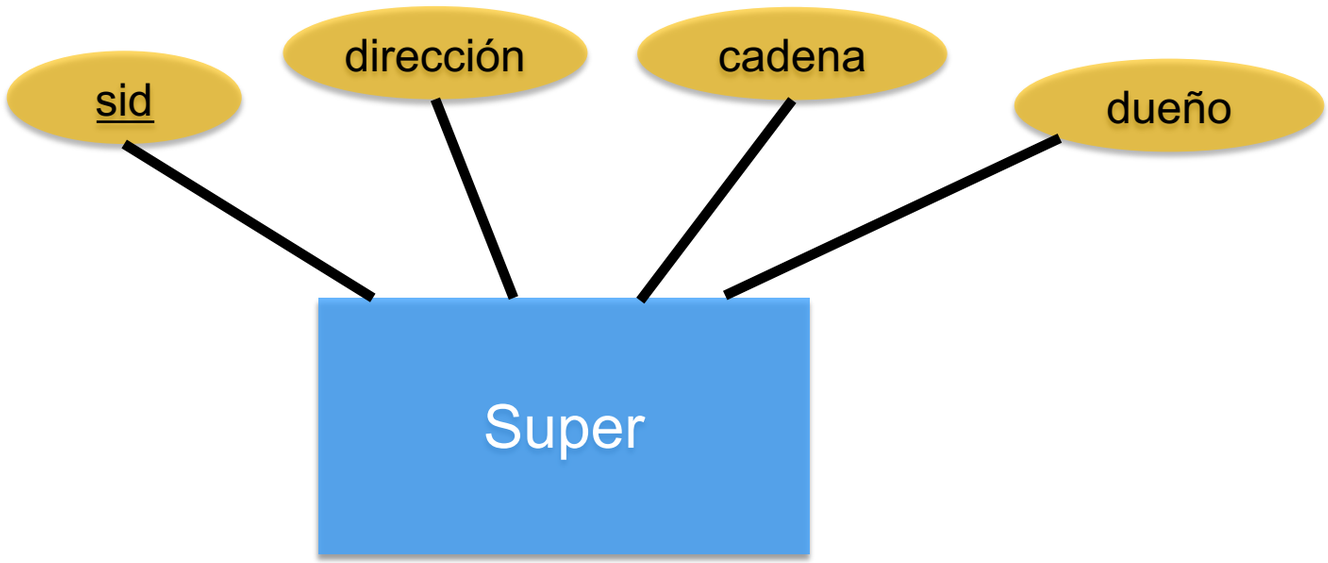
R1(d,e,b)

R2(a,c,d,e)

Observaciones:

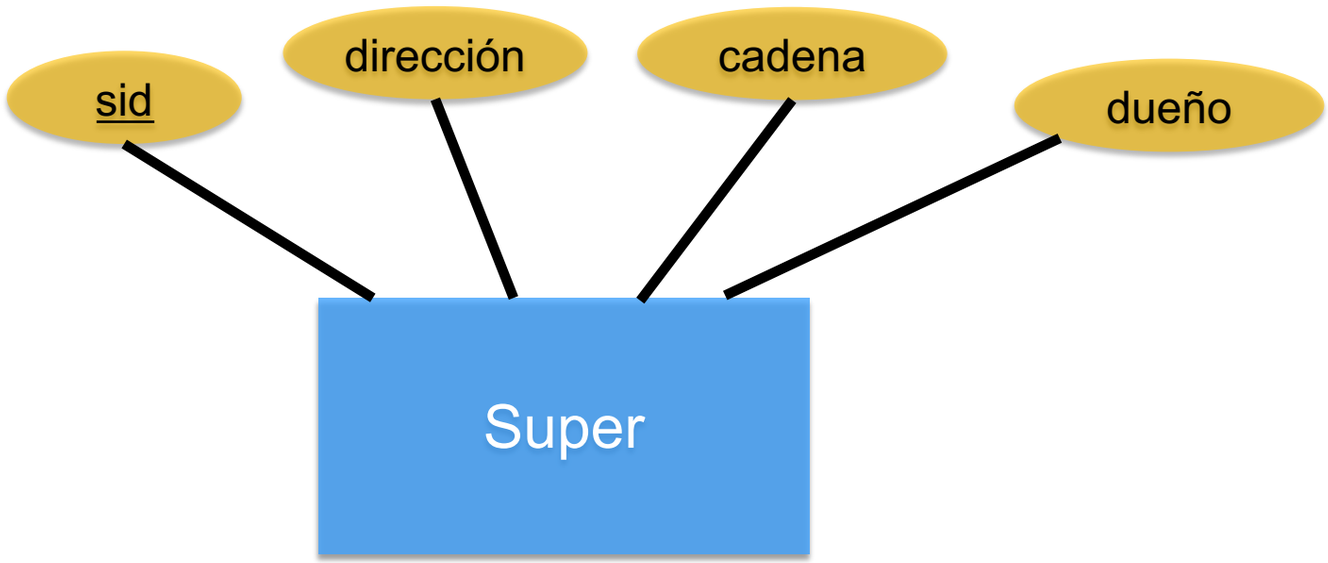
- Tabla R1 permite validar que se cumple DF2
- ¡No hay una tabla que permite validar DF1!
- ¡Hay que hacer el join de R1 y R2 para validar DF1!

Si nos “equivocamos” en E/R



Super(<u>sid</u>, dirección, cadena, dueño)

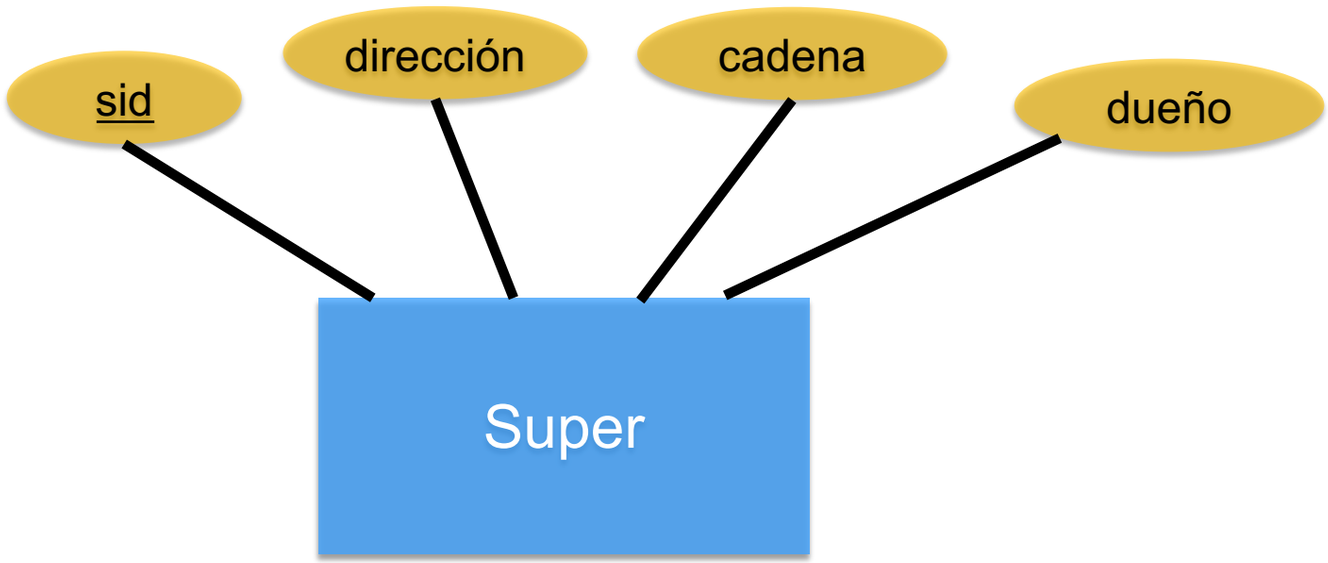
Si nos “equivocamos” en E/R



Super(<u>sid</u>, dirección, cadena, dueño)

- cadena → dueño

Si nos “equivocamos” en E/R



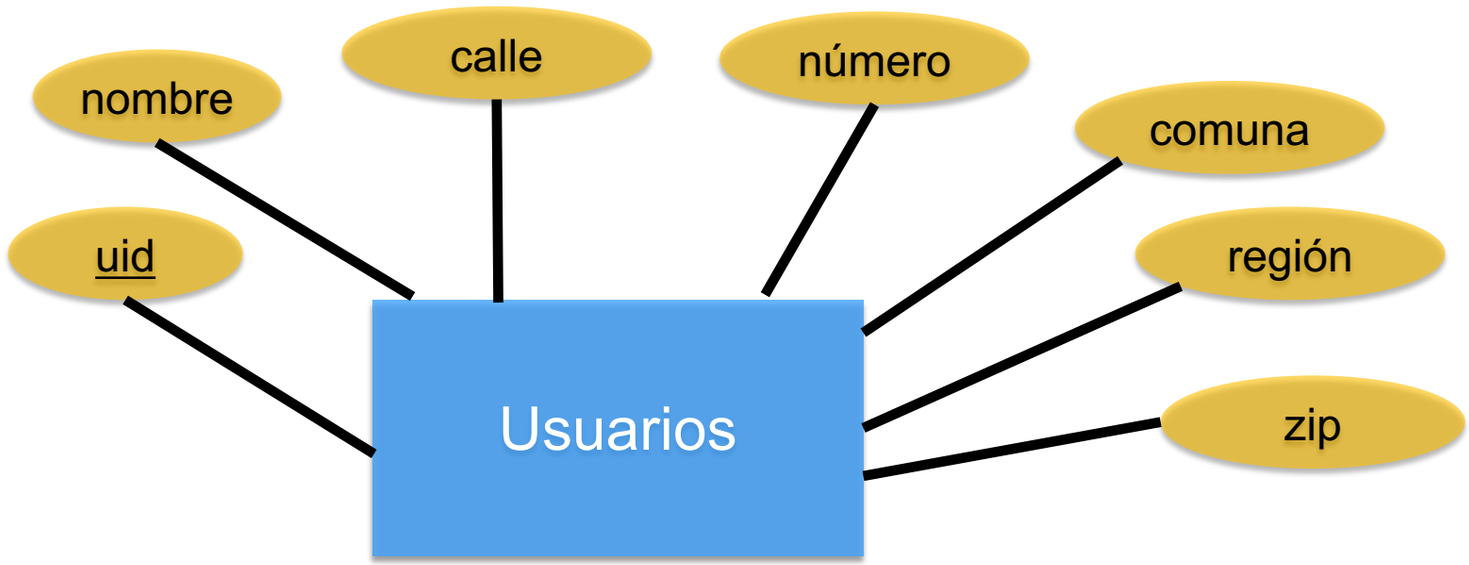
Super(<u>sid</u>, dirección, cadena, dueño)

- cadena → dueño

Super(<u>sid</u>, dirección, cadena)
Dueños(<u>cadena</u>, dueño)

BCNF

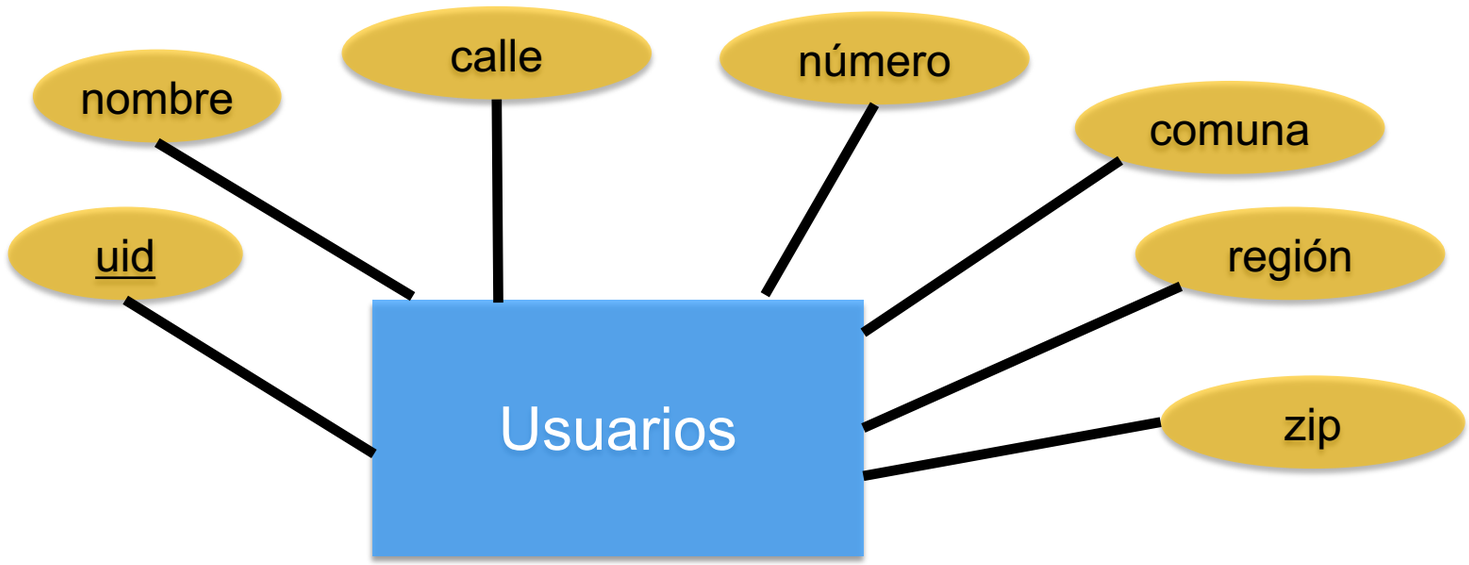
Si nos “equivocamos” en E/R



Usuarios(<u>uid</u>, nombre, calle, número, comuna, región, zip)

- zip → calle, comuna, región

Si nos “equivocamos” en E/R



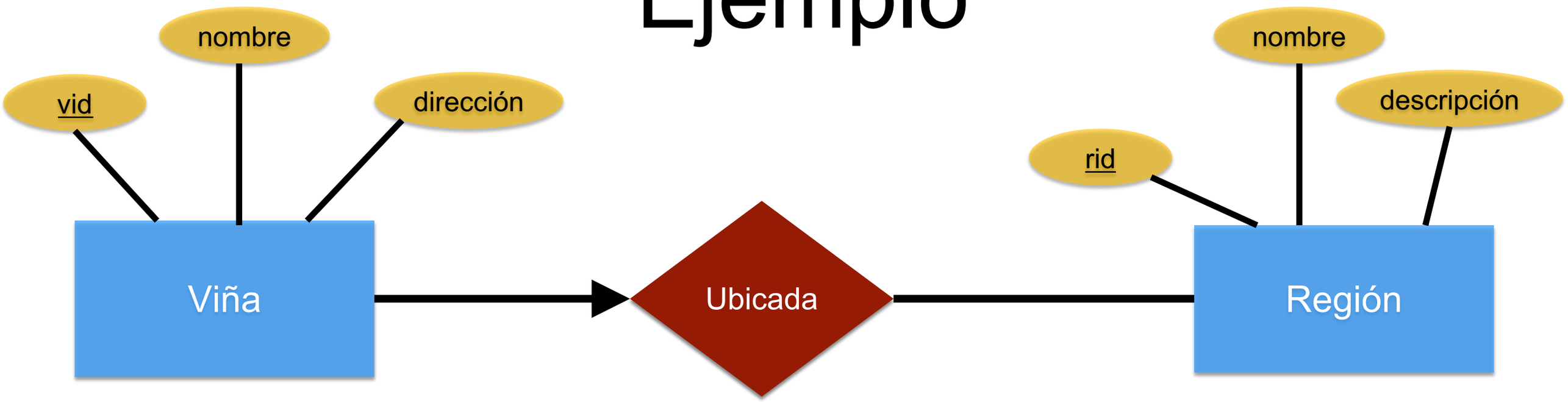
Usuarios(<u>uid</u>, nombre, calle, número, comuna, región, zip)

- zip → calle, comuna, región

Usuarios(<u>uid</u>, nombre, número, zip)
CódigoPostal(<u>zip</u>, calle, comuna, región)

BCNF

Ejemplo



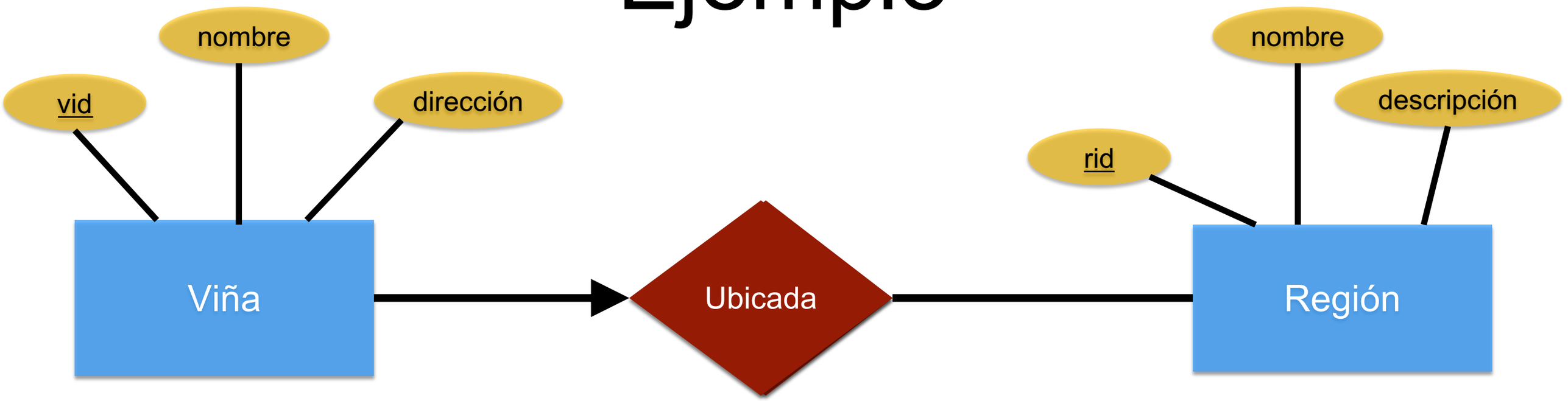
Viña(<u>vid</u>, nombre, dirección)

Región(<u>rid</u>, nombre, descripción)

Ubicada(<u>vid</u>, rid)

BCNF

Ejemplo



Viña(vid, nombre, dirección, rid)

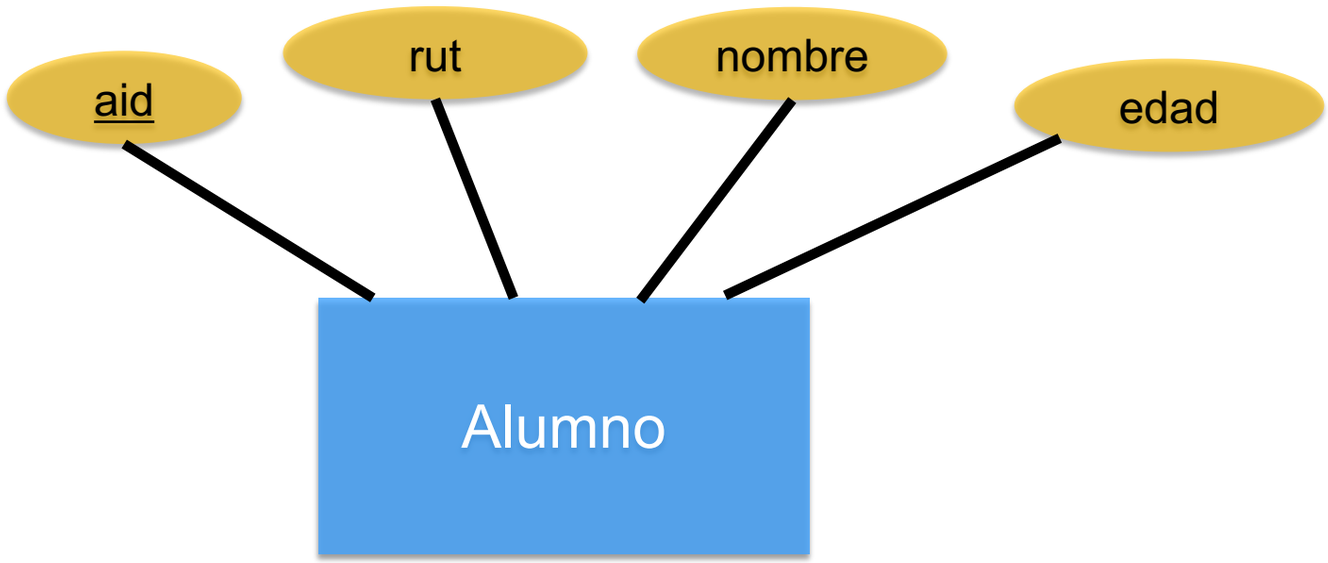
Región(rid, nombre, descripción)

BCNF

- Vid → rid

- vid es una llave

Si agregamos llaves artificiales

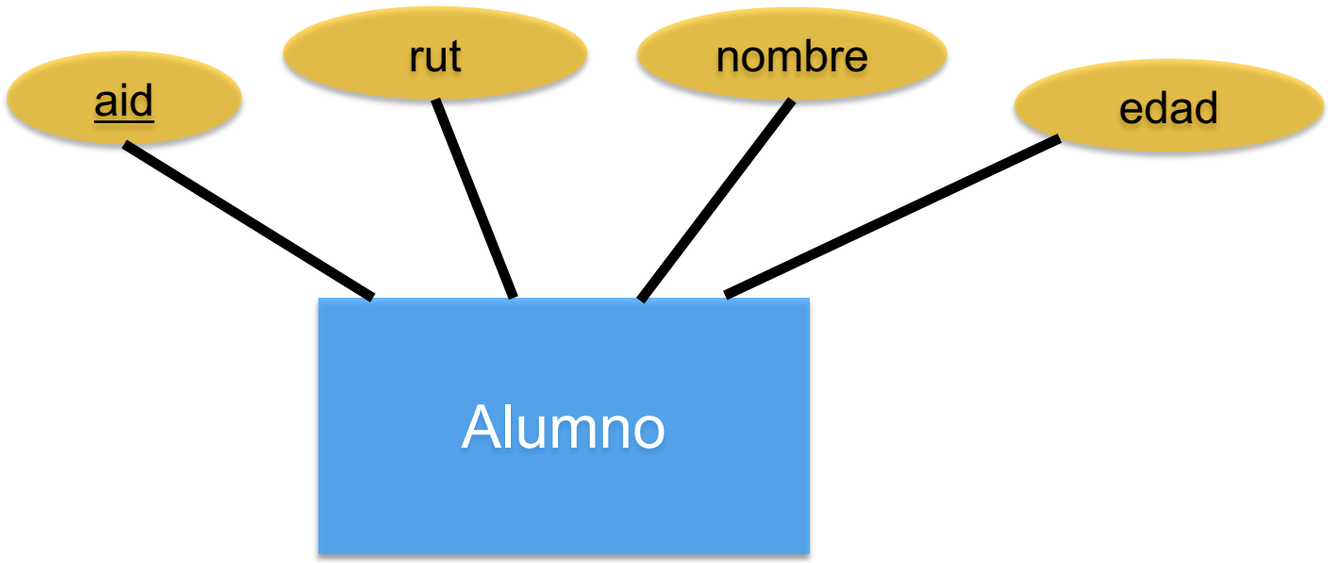


Alumnos(<u>aid</u>, rut, nombre, edad)

- aid → rut, nombre, edad
- rut → nombre, edad (pero no aid)

NOT BCNF

Si agregamos llaves artificiales

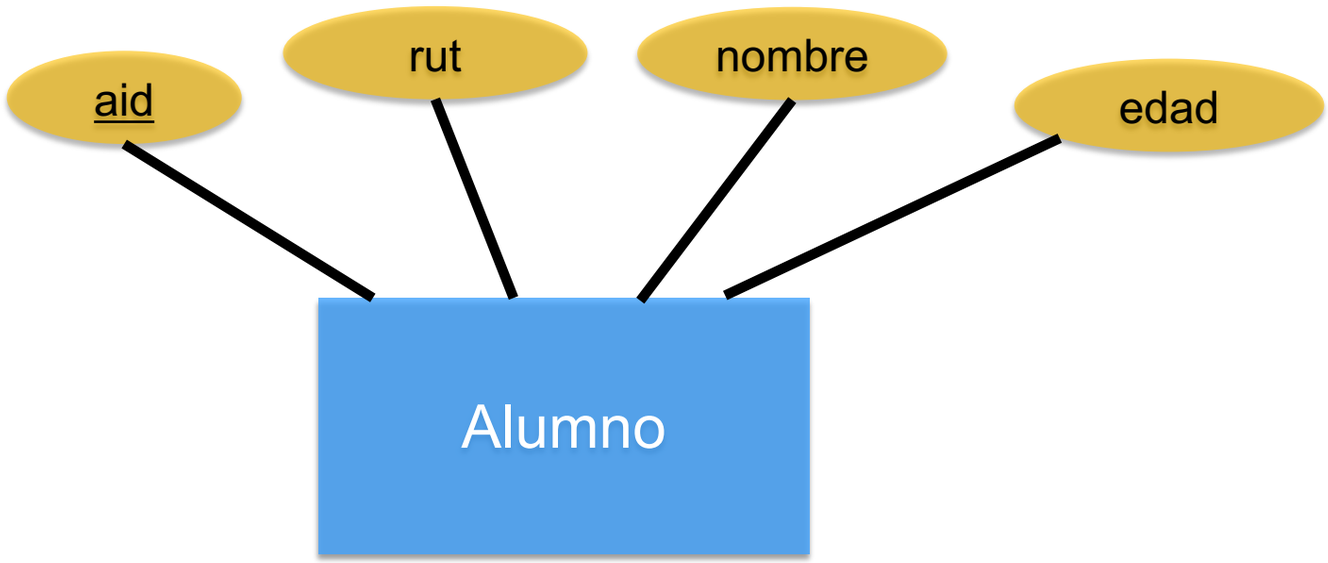


Alumnos(<u>aid</u>, rut, nombre, edad)

- aid → rut, nombre, edad
- rut → aid, nombre, edad

BCNF

Si agregamos llaves artificiales



Alumnos(<u>aid</u>, rut, nombre, edad)

En realidad, aquí sabemos qué las llaves son:

- aid
- rut

→ **BCNF**

BCNF vs 3NF

Nuestra descomposición siempre va a ser sin pérdida de información, sin embargo, puede ocurrir lo siguiente:

R(a,b,c,d,e)

- DF1: a,b,c → d,e
- DF2: d,e → b

llaves:

- a,b,c
- d,e,a,c

¡Hay una violación de BCNF: d,e → b!

¡Pero el esquema ya está en 3NF!

Recapitulación

- Partimos desde tablas posiblemente mal diseñadas que generan anomalías
- Agregamos dependencias funcionales
- Intentamos descomponer en BCNF
- Si tenemos problemas con las dependencias utilizar 3NF