

# 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

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

# Redundancia en los datos

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(gid, nombre, rating, horas, valorHora)**

# Redundancia en los datos

## Guías

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

Gasto de espacio

## Guías

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

Anomalías de inserción

## Guías

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

Anomalías de actualización

## Guías

gid	nombre	score	horas	valorHora
1	Juan	8	40	17000
2	Johanna	8	30	18000
3	Cristian	5	30	15000
4	Pedro	8	32	18000

Si cambia el valor hora para el score 8

# Problemas con la redundancia

Anomalías de actualización

## Guías

gid	nombre	score	horas	valorHora
1	Juan	8	40	17000
2	Johanna	8	30	18000
3	Cristian	5	30	15000
4	Pedro	8	32	18000

Si cambia el valor hora para el score 8

Datos inconsistentes!!

# Problemas con la redundancia

Anomalías de actualización

## Guías

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 cambia el valor hora para el score 8

Cambiar en todas las tuplas

# Problemas con la redundancia

Anomalías de eliminación

## Guías

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

## Guías

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

# Solución

¿Qué pasó aquí?

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

Esto se llama una **dependencia funcional**

# Restricciones de Integridad

- Los datos deben satisfacer restricciones de integridad
- Estas restricciones son importantes en la modelación
- ¿Cómo nos pueden ayudar a especificar lo que queremos en cada relación?

# 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

# Dependencia Funcional

Ejemplo

Personas(rut, nombre)

- rut → nombre

# Dependencia Funcional

## Ejemplo

Personas(rut, nombre)

- rut → nombre

Películas(pid, título, año, director)

- título, año → director

# Dependencia Funcional

## Ejemplo

Personas(rut, nombre)

- rut → nombre

Películas(pid, título, año, director)

- título, año → director

GeolInfo(cid, nombre\_ciudad, región, num\_habitantes, intendente, coordenadas)

- nombre\_ciudad, región → num\_habitantes, intendente, coordenadas\_geográficas

# Dependencia Funcional

## Ejemplo

Personas(rut, nombre)

- rut → nombre

Lado izquierdo no es necesariamente una llave

Películas(pid, título, año, director)

- título, año → director

GeolInfo(cid, nombre\_ciudad, región, num\_habitantes, intendente, coordenadas)

- nombre\_ciudad, región → num\_habitantes, intendente, coordenadas\_geográficas

# Dependencia Funcional

Definición

$X \rightarrow Y$  es válida en una relación  $R$  ssi para toda tupla  $t_1, t_2 \in R$  se tiene:

$$\pi_X(t_1) = \pi_X(t_2) \quad \text{implica} \quad \pi_Y(t_1) = \pi_Y(t_2)$$

# Dependencia Funcional

Ejemplo

¿Qué dependencias agregaría?

- Persona(rut, nombre, apellido\_p, apellido\_m)
- Festival(nombre, año, ciudad)
- Entrada(rut, nombre\_festival, año\_festival, ciudad\_festival, categoría, precio)

# Dependencia Funcional

## Ejemplo

¿Qué dependencias agregaría?

- Persona(rut, nombre, apellido\_p, apellido\_m)  
rut → nombre, apellido\_p, apellido\_m
- Festival(nombre, año, ciudad)  
nombre → ciudad
- Entrada(rut, nombre\_festival, año\_festival, ciudad\_festival, categoría, precio)  
nombre\_festival, año\_festival, ciudad\_festival, categoría → precio

# Dependencia Funcional

## Ejemplo

Las llaves:

- Persona(rut, nombre, apellido\_p, apellido\_m)
- Festival(nombre, año, ciudad)
- Entrada(rut, nombre\_festival, año\_festival,  
ciudad\_festival, categoría, precio)

# Dependencia Funcional

## Ejemplo

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

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

# Dependencia Funcional

## Ejemplo

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

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

¿Cuál va a ser la llave?

# Buenas y malas dependencias

DJE	Dept	Jefe	Empleado	ES	Empleado	Salario
	D1	Pérez	Ureta		Ureta	600
	D1	Pérez	Assad		Assad	800
	D2	Correa	Vargas		Vargas	800
	D3	Pérez	Gómez		...	...
	D4	Pérez	Camus		...	...
	...	...	...			

- **DJE:** Depto → Jefe
- **ES:** Empleado → Salario (Empleado es llave)

# Buenas y malas dependencias

Anomalía de inserción

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

No podemos almacenarlo en **DJE**

# Buenas y malas dependencias

Anomalía de eliminación

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

Redundancia

Tenemos dos tuplas indicando que Pérez es jefe de D1

# 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 que departamento (ej. Pérez con D1)

# Buenas y malas dependencias

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

¡Este tipo de situaciones queremos evitar!

# Anomalías

## Ejemplo

Tabla de personas, que pueden tener más de un teléfono

NRTC	nombre	run	teléfono	ciudad
	Fran	12.256.279-0	98456258	Santiago
	Fran	12.256.279-0	88845621	Santiago
	José	15.963.279-2	97584263	Curicó
	Andy	17.145.203-1	87775021	Temuco
	...	...	...	

- run → nombre, ciudad (pero no run → teléfono)

# Anomalías

## Ejemplo

Tabla de personas, que pueden tener más de un teléfono

- Redundancia?
- Anomalía de actualización?

# Anomalías

Anomalía de inserción - actualización

Cuando introducimos o modificamos datos en una tabla y no reflejamos la inserción (o modificación) en las otras tablas

# Anomalías

Anomalía de eliminación

Cuando se eliminan un conjunto de valores y perdemos más datos de los que se querían borrar

# Anomalías

Redundancia

Cuando se almacena un dato más de una vez

# 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

Ejercicio: demostrar la observación

# Dependencias

Observación 2

Si  $Z \subseteq Y$ , y tenemos la dependencia:

- $X \rightarrow Y$

Podemos deducir que:

- $X \rightarrow Z$

Con  $X, Y, Z$  conjuntos de atributos

Ejercicio: demostrar la observación

# Dependencias

Observación 3

Llamamos dependencia trivial a la dependencia:

$$X \rightarrow Y$$

Si se tiene que  $Y \subseteq X$

Ejercicio: demostrar la observación

# Dependencias

Observación 4

Si tenemos que:

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

Podemos decir que:

$$X \rightarrow Y, Z$$

Ejercicio: demostrar la observación

# Dependencias

Observación 5

Si tenemos que:

$$X \rightarrow Y$$

Podemos decir que:

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

Para cada  $Z$

Ejercicio: demostrar la observación

# Dependencias

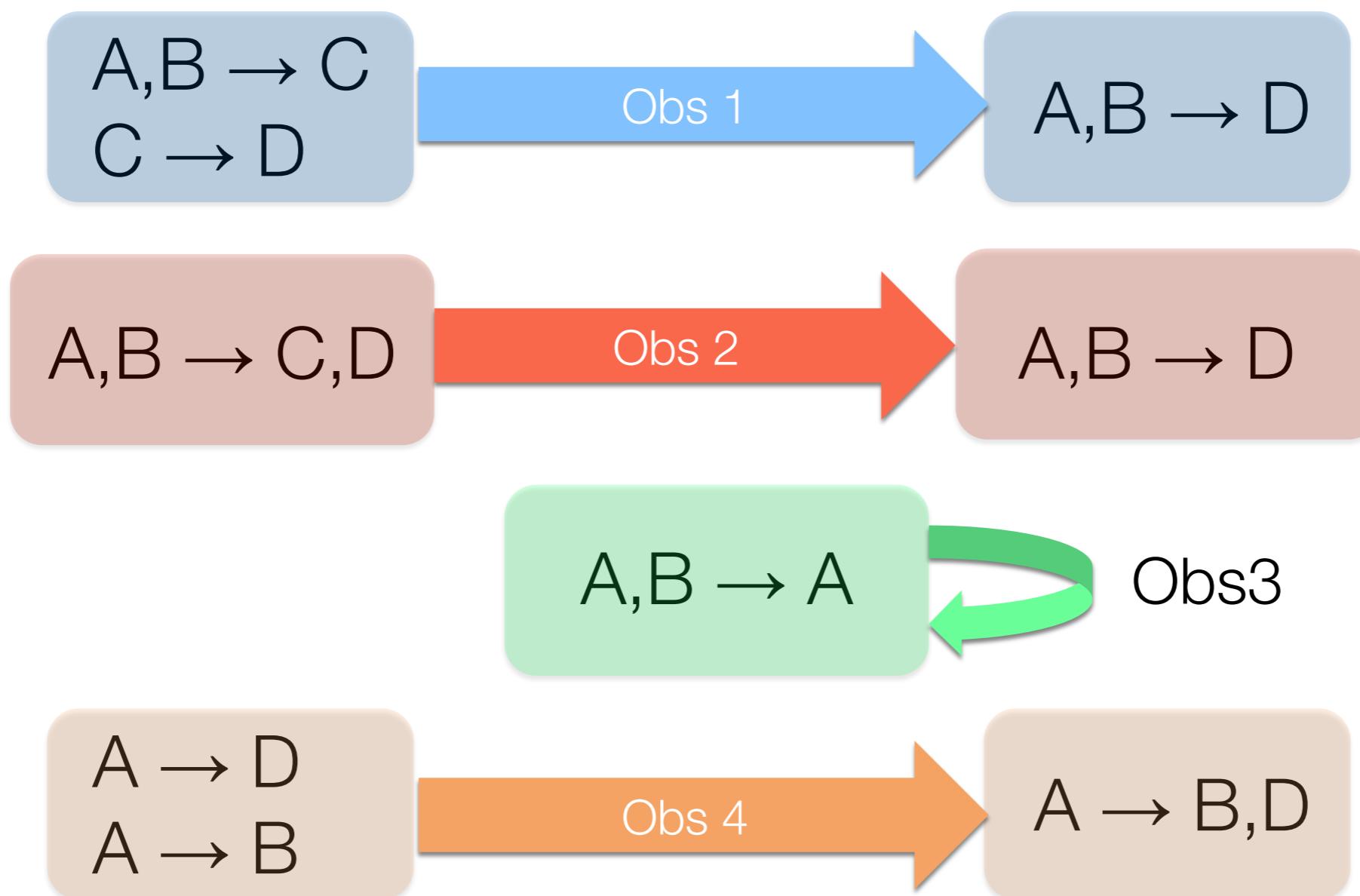
Observación 6

Si tenemos  $X \rightarrow Y$ , 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**

# Dependencias

Observaciones

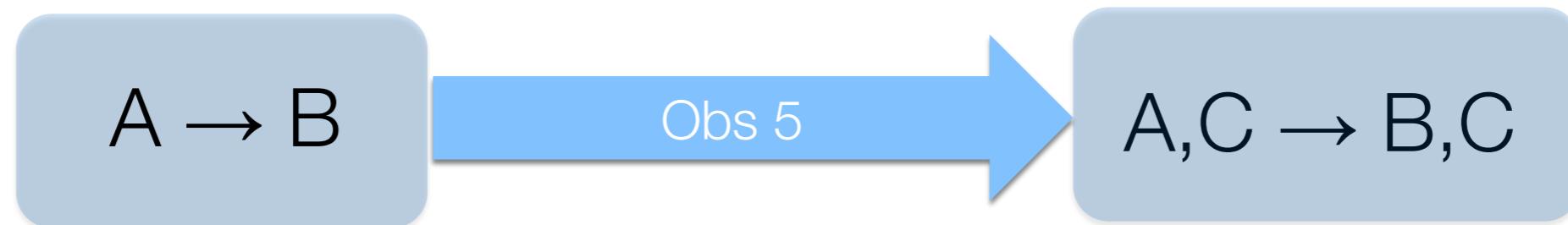
$R(A,B,C,D,E)$



# Dependencias

Observaciones

$R(A,B,C,D,E)$



Observación 6: A,B es llave candidata

# Dependencias

## Ejemplo

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

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

Podemos inferir además  $a \rightarrow b, c$  (por lo tanto, **a** es llave)

# Dependencias

Ejemplo

Podemos inferir además  $a \rightarrow b, c$  (por lo tanto, **a** es llave)

**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  $\pi_b(t_1) = \pi_b(t_2)$

# Dependencias

Ejemplo

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:** usar esta idea de demostración para los ejercicios planteados en cada observación

# 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
- título → 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 → dirección, teléfono
$R_2$	cine, pelicula, horario, precio	cine, pelicula, horario → precio
$R_3$	pelicula, director	pelicula → director

# Descomposición

Ejemplo: buen diseño

Es un buen diseño porque:

- No hay anomalías, cada dependencia funcional define una llave
- No perdemos dependencias funcionales, pues todas están restringidas a sus respectivas tablas
- No perdemos información:

$$R_1 = \pi_{cine, direccion, telefono}(MAL)$$

$$R_2 = \pi_{cine, pelicula, horario, precio}(MAL)$$

$$R_3 = \pi_{pelicula, director}(MAL)$$

$$MAL = R_1 \bowtie R_2 \bowtie R_3$$

# 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

# Boyce-Codd Normal Form (BCNF)

Las tablas pueden tener más de una llave

Generalmente nos enfocamos en las llaves minimales o candidatas

**X** es minimal si no existe una llave **X'** tal que  $\mathbf{X}' \subseteq \mathbf{X}$

# Boyce-Codd Normal Form (BCNF)

Las tablas pueden tener más de una llave

Generalmente nos enfocamos en las llaves minimales o candidatas

**X** es minimal si no existe **X'** tal que  $X' \subseteq X$

**La definición vale para cualquier llaves minimales y no minimales**

# 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  $X \rightarrow Y$  y descomponer en 2 tablas con esquemas:

$$R1(X,Y) \quad \text{y} \quad R2(A - Y)$$



# BCNF

## Algoritmo

- INPUT: - el esquema de la tabla R(A)  
- dependencias funcionales estandarizadas

1. Tomar una dependencia funcional  $X \rightarrow Y$  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
...	...	...	...	...	...	...
...	...	...	...	...	...	...

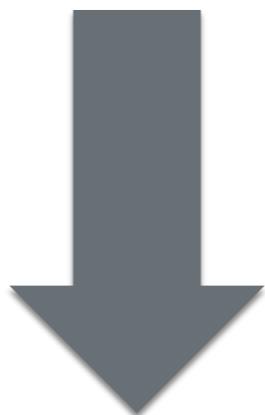
2. Repetir con las tablas que aun no están en FNBC

# BCNF

Algoritmo

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

$a,b \rightarrow c,e$  rompe BCNF



**R1(a,b,c,e)**

**R2(a,b,d,f)**

# 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:

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

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

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

# BCNF

## Ejemplo

Descomposición 1:

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

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



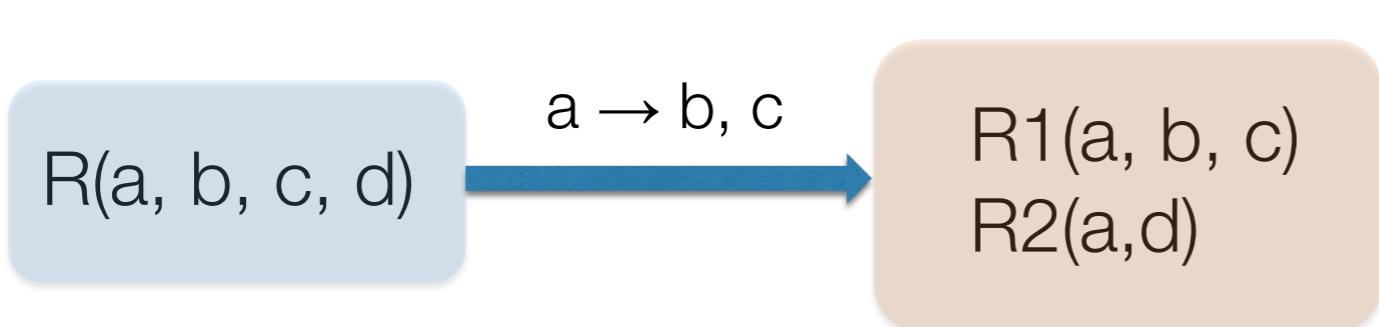
# BCNF

## Ejemplo

Descomposición 1:

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

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



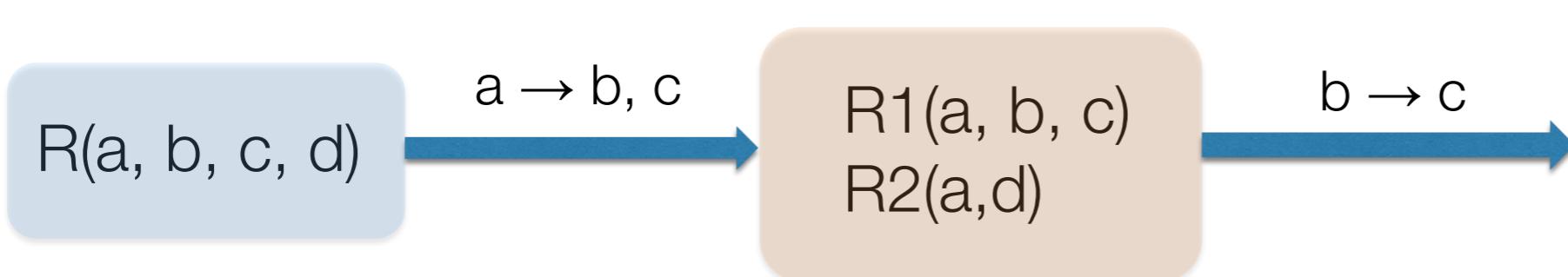
# BCNF

## Ejemplo

Descomposición 1:

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

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



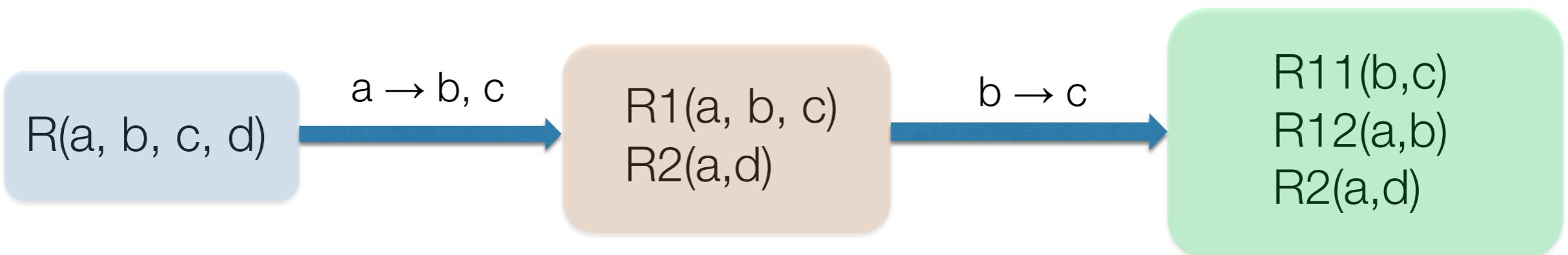
# BCNF

## Ejemplo

Descomposición 1:

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

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



# BCNF

## Ejemplo

Descomposición 2:

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

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



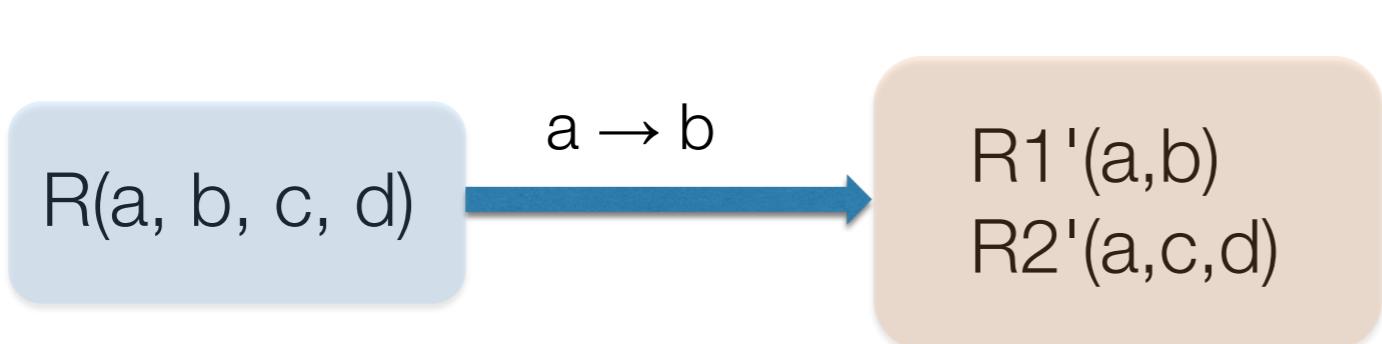
# BCNF

## Ejemplo

Descomposición 2:

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

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



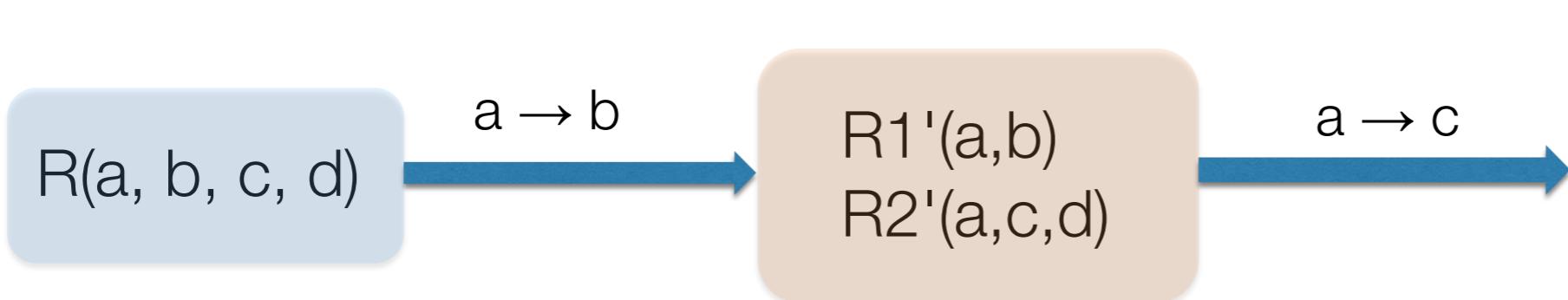
# BCNF

## Ejemplo

Descomposición 2:

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

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



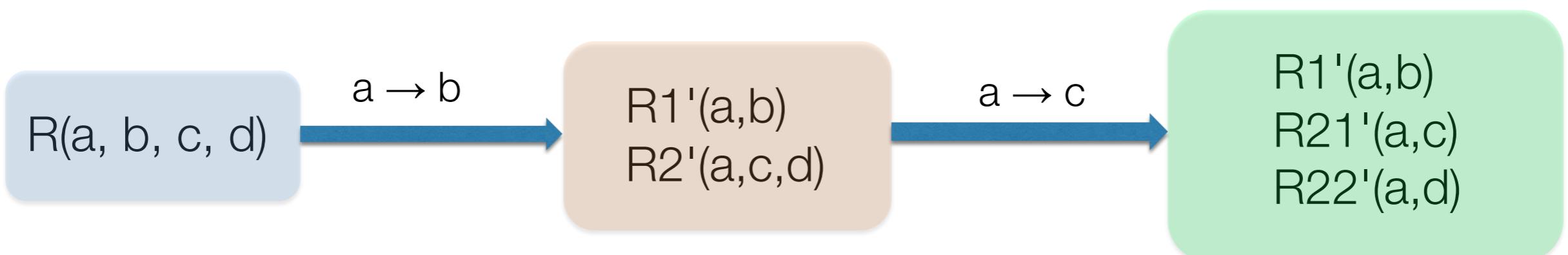
# BCNF

## Ejemplo

Descomposición 2:

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

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



# 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	precio	categoría
Canon T3	fotografía	300	fotografía
Nokia 5000	fotografía	400	fotografía
Galaxy IV	celular	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**(A, B, C) y dependencia funcional  $A \rightarrow B$ , para A, B, C conjuntos de atributos disjuntos, se tiene que la descomposición en **R1**(A, B) y **R2**(A, C) con  $A \rightarrow B$  es **sin pérdida de información**

# Problemas con BCNF

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

UCP(unidad, compañía, producto)

- DF1: unidad → compañía
- DF2: compañía, producto → unidad

Hay una violación de BCNF (unidad → compañía), ya que de DF2 se sabe que (compañía, producto) es llave candidata, pero (unidad) no.

# Problemas con BCNF

- DF1: unidad → compañía
- DF2: compañía, producto → unidad

Pero al descomponer primero por DF1:

UC(unidad, compañía)

UP(unidad, producto)

Para la primera relación aplica la dependencia DF1  
(unidad → compañía), pero para la segunda no aplica  
ninguna

Se perdió la DF1!

# Problemas con BCNF

- DF1: unidad → compañía
- DF2: compañía, producto → unidad

Si descomponemos primero con DF2:

UCP(compañía, producto, unidad)

CP(compañía, producto)

Para la primera relación aplica la dependencia DF2  
(compañía, producto → unidad), pero para la segunda no  
aplica ninguna.

Se perdió la DF2!

# 3NF

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

**Z** es llave minimal si no existe llave **Z'** tal que  $Z' \subseteq Z$

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

# 3NF

## Ejemplo

Curso(sala, profesor, módulo)

- sala → profesor
- profesor, módulo → sala

Al llevarla a BCNF:

Curso1(sala, profesor)

- sala → profesor

Curso2(sala, módulo)

- Sin dependencias!

# 3NF

## Ejemplo

Curso(sala, profesor, módulo)

- sala → profesor
- profesor, módulo → sala

Pero esta relación está en 3NF: (profesor, módulo) es llave minimal, por lo que profesor es parte de una llave

Permitimos redundancia porque en este caso no existe descomposición en BCNF que preserve las dependencias

# 3NF

Algoritmo

INPUT:

- el esquema de la tabla  $R(A)$
- dependencias funcionales en un formato estandarizado

1. Para cada df  $X \rightarrow Y$  crear una tabla con esquema  $X \cup Y$
2. Si al final, los esquemas resultantes  $R_1, \dots, R_n$  no contienen una llave del original, agregar una.

# 3NF

## Ejemplo

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

<u>Nombre Comuna</u>	<u>Región</u>	Tasa x m2	RUT	Nombre	Apellido	<u>Rol Lote</u>	M2	Avaluo
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  $X \cup Y$

<u>Nombre Comuna</u>	<u>Región</u>	<u>Tasa x m2</u>
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  $X \cup Y$

<u>Nombre Comuna</u>	<u>Región</u>	<u>Tasa x m2</u>
A	I	2
A	I	2
B	III	1,1
B	X	1,1
C	V	0,5
	...	...

<u>RUT</u>	<u>Nombre</u>	<u>Apellido</u>
111111	Claudio	Gonzalez
222222	Maria	Zapata
333333	Carlos	Fernandez
...	...	...

RUT → Nombre, Apellido

NombreComuna, Region → Tasa

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  $X \cup Y$

<u>Nombre Comuna</u>	<u>Región</u>	<u>Tasa x m2</u>
A	I	2
A	I	2
B	III	1,1
B	X	1,1
C	V	0,5
...	...	...

<u>RUT</u>	<u>Nombre</u>	<u>Apellido</u>
111111	Claudio	Gonzalez
222222	Maria	Zapata
333333	Carlos	Fernandez
...	...	...

RUT → Nombre, Apellido

<u>Tasa x m2</u>	<u>M2</u>	<u>Avaluo</u>
34	455	960
35	570	1040
27	895	1790
...	...	...

NombreComuna, Region → Tasa

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  $X \cup Y$

<u>Nombre</u> <u>Comuna</u>	<u>Región</u>	<u>Tasa x m2</u>
A	I	2
A	I	2
B	III	1,1
B	X	1,1
C	V	0,5
...	...	...

<u>RUT</u>	<u>Nombre</u>	<u>Apellido</u>
111111	Claudio	Gonzalez
222222	Maria	Zapata
333333	Carlos	Fernandez
...	...	...

RUT → Nombre, Apellido

<u>Tasa x m2</u>	<u>M2</u>	<u>Avaluo</u>
34	455	960
35	570	1040
27	895	1790
...	...	...

NombreComuna, Region → Tasa

Tasa, m2 → Avalúo

<u>Nombre</u> <u>Comuna</u>	<u>Región</u>	<u>RUT</u>	<u>Rol</u> <u>Lote</u>	<u>M2</u>
A	I	111111	34	455
...	...	...	...	...

NombreComuna, Region, Rol → RUT, M2



{NombreComuna, Region, Rol }

# 3NF

## Ejemplo

2. Si al final, los esquemas resultantes R<sub>1</sub>, ..., R<sub>n</sub> no contienen una llave del original, agregar una.

<u>Nombre Comuna</u>	<u>Región</u>	<u>Tasa x m<sup>2</sup></u>
A	I	2
A	I	2
B	III	1,1
B	X	1,1
C	V	0,5
...	...	...

<u>RUT</u>	<u>Nombre</u>	<u>Apellido</u>
111111	Claudio	Gonzalez
222222	Maria	Zapata
333333	Carlos	Fernandez
...	...	...

RUT → Nombre, Apellido

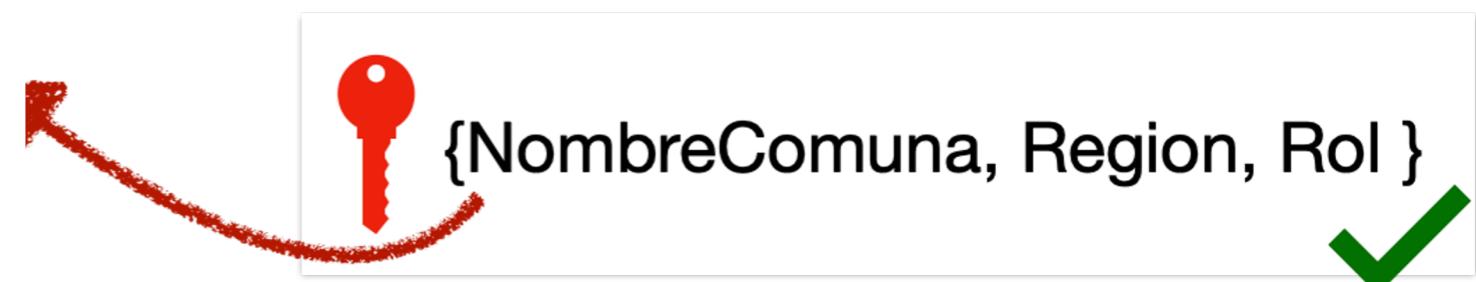
<u>Tasa x m<sup>2</sup></u>	<u>M2</u>	<u>Avaluo</u>
34	455	960
35	570	1040
27	895	1790
...	...	...

NombreComuna, Region → Tasa

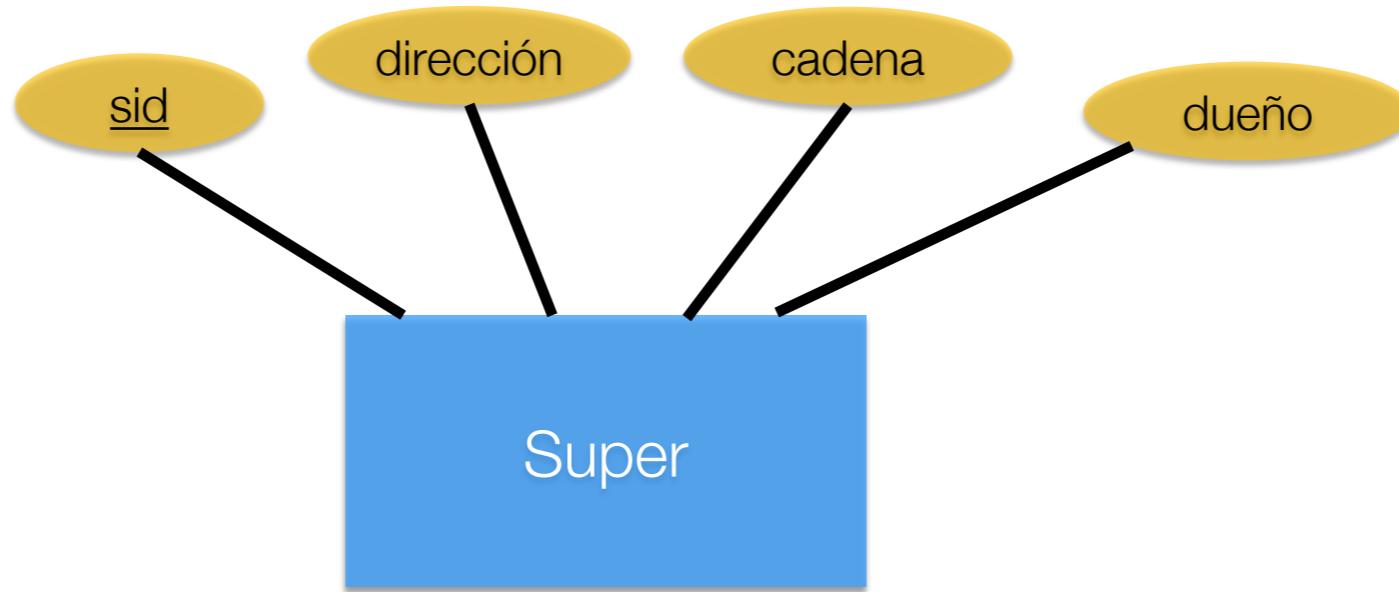
Tasa, m<sup>2</sup> → Avalúo

<u>Nombre Comuna</u>	<u>Región</u>	<u>RUT</u>	<u>Rol Lote</u>	<u>M2</u>
A	I	111111	34	455
...	...	...	...	...

NombreComuna, Region, Rol → RUT, M2

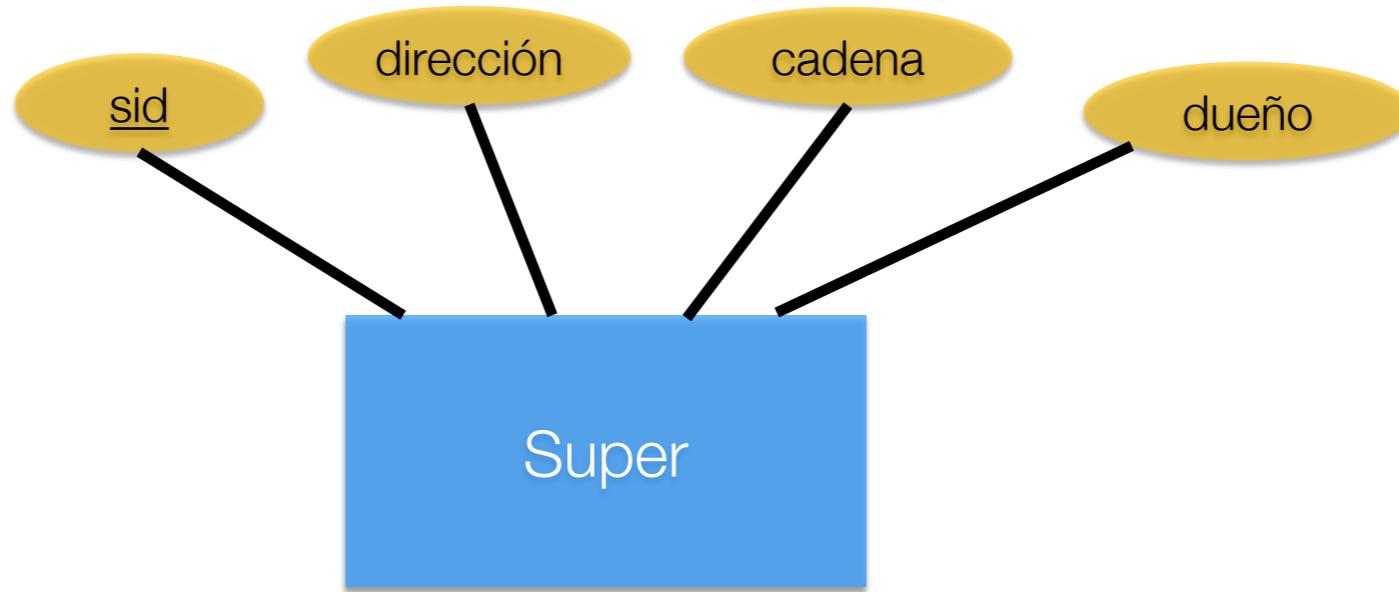


# Si nos “equivocamos” en E/R



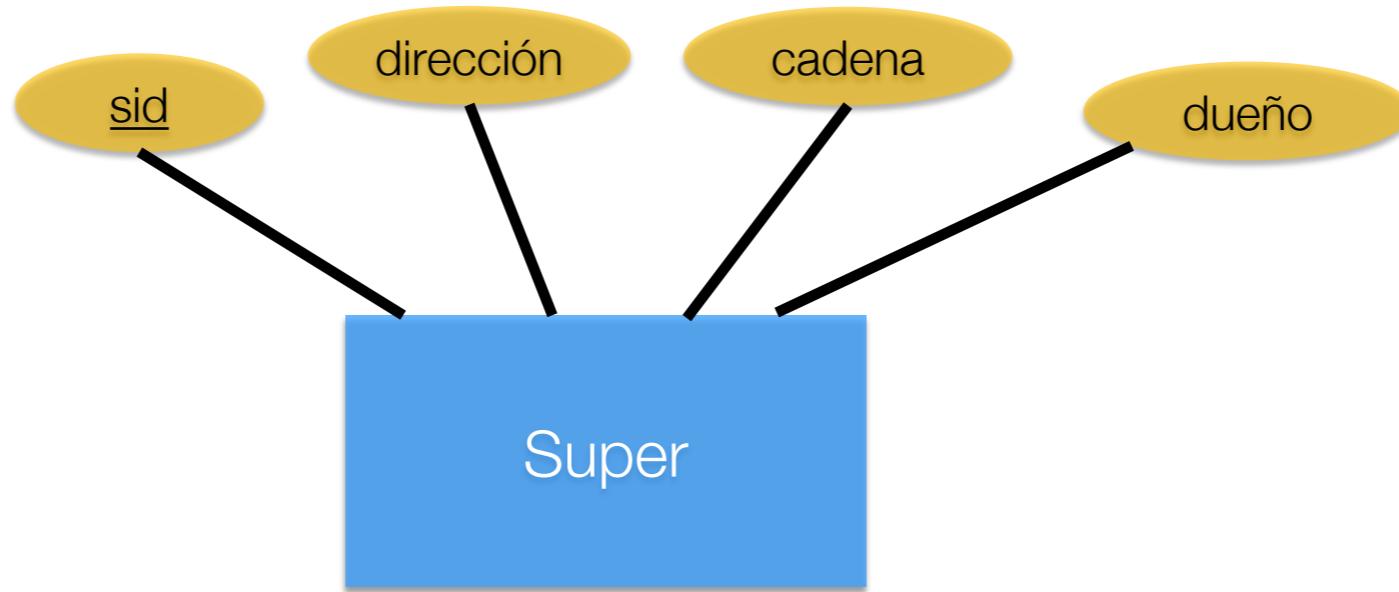
Super(sid, dirección, cadena, dueño)

# Si nos “equivocamos” en E/R



Super(sid, dirección, cadena, dueño)  
• cadena → dueño

# Si nos “equivocamos” en E/R



Super(sid, dirección, cadena, dueño)

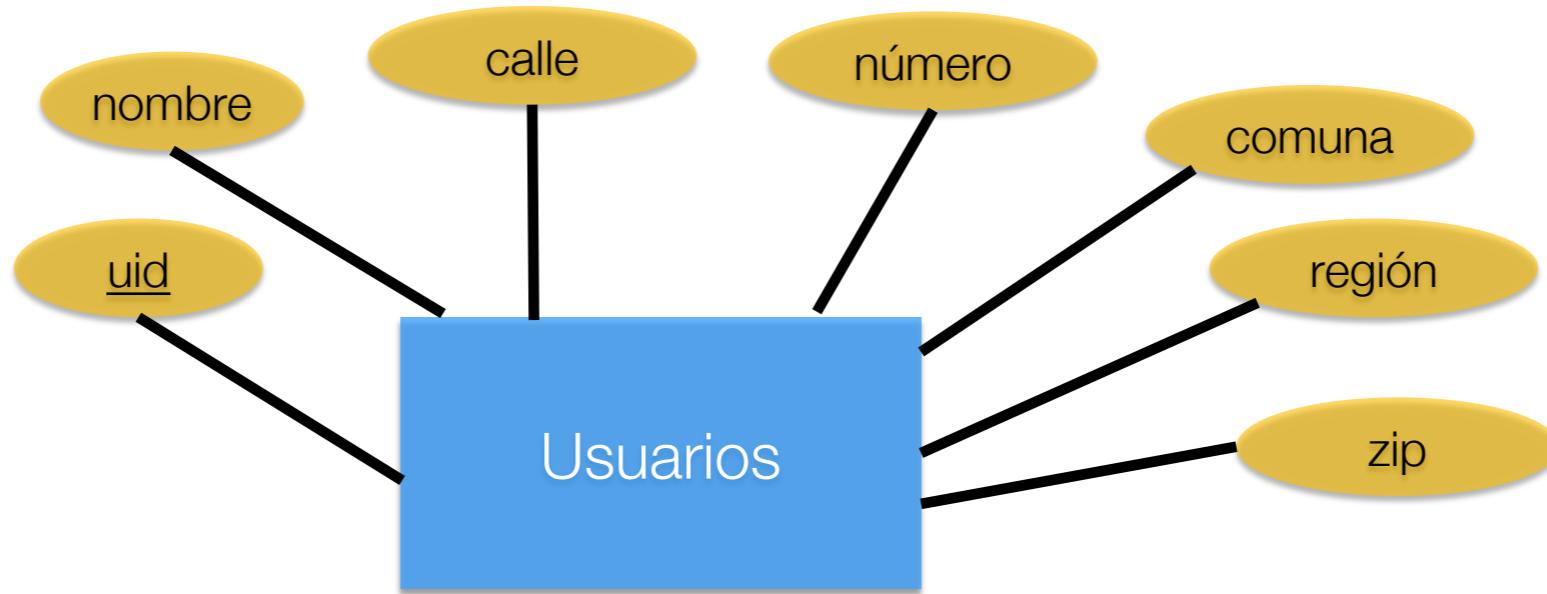
- cadena → dueño

Super(sid, dirección, cadena)

Dueños(cadena, dueño)

**BCNF**

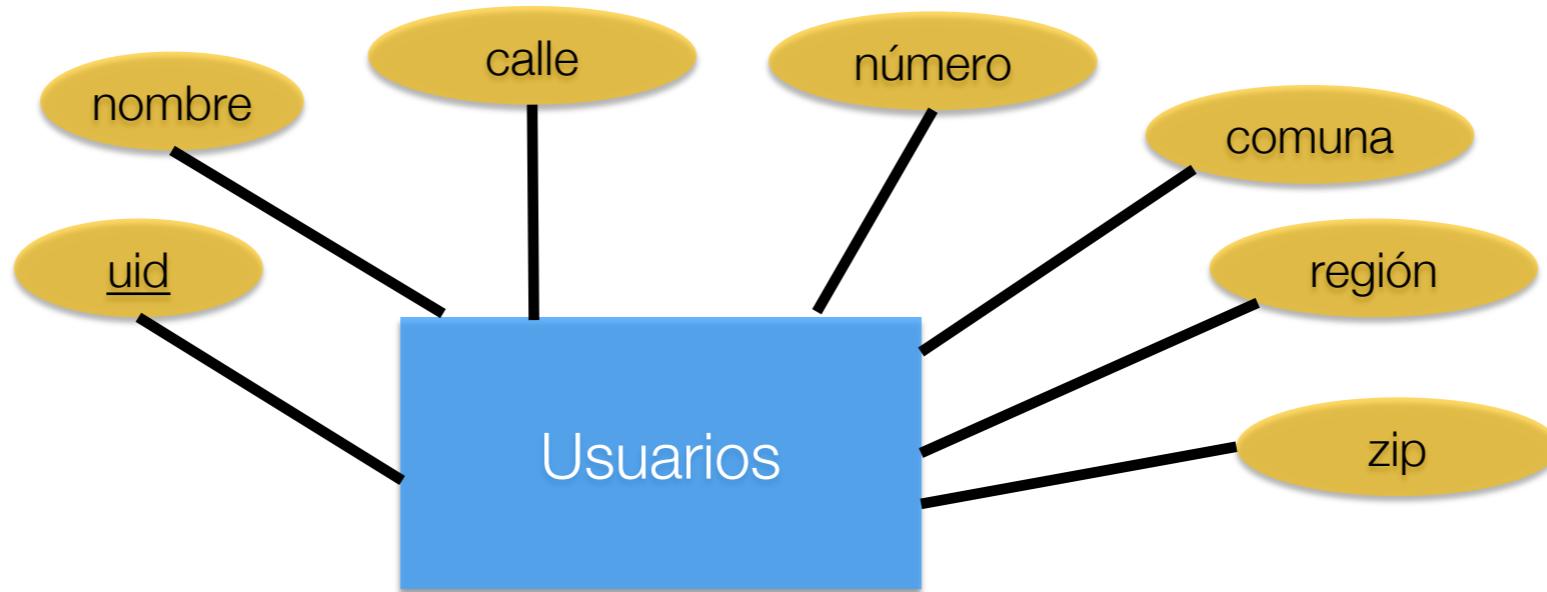
# Si nos “equivocamos” en E/R



Usuarios(uid, nombre, calle, número, comuna, región, zip)

- zip → calle, comuna, región

# Si nos “equivocamos” en E/R



Usuarios(uid, nombre, calle, número, comuna, región, zip)  
• zip → calle, comuna, región

Usuarios(uid, nombre, número, zip)  
CódigoPostal(zip, calle, comuna, región)

**BCNF**

# 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