

# **Diseño de Bases Relacionales**

## **Normalización**



# Normalización

- ✍ Las bases de datos mal diseñadas tienen problemas de:
  - ✍ Almacenamiento redundante (varias copias de la misma información)
  - ✍ Perdidas no deseadas de información al modificar tuplas
  - ✍ La base entra en un estado no consistente al borrar una tupla
  - ✍ Imposibilidad de almacenar cierta información
    - ✍ registro(estudiante\_id, estudiante\_nombre, curso\_id, curso\_nombre)
- ✍ En este capítulo veremos una serie de requerimientos que hemos de hacer a las relaciones para que los dos problemas señalados arriba no se produzcan

**El resumen de este capítulo es: Todos los atributos de una relación deben ser atómicos y depender única y totalmente de la clave primaria.**

**La Clave Primaria, toda la clave primaria y nada más que la clave primaria**

# Primera Forma Normal

- ✍ Un atributo es atómico si sus elementos se pueden considerar como unidades indivisibles:
  - ✍ Ejemplos de dominios no atómicos:
    - ✍ Nombre (atributo compuesto)
    - ✍ Teléfonos (Atributos multivaluados)
- ✍ Un esquema R satisface la primera forma normal si los dominios de todos los atributos son atómicos
- ✍ Los valores no atómicos complican el almacenamiento y pueden provocar redundancia:
  - ✍ Por ejemplo, cuentas bancarias almacenadas con sus propietarios

# Primera Forma Normal (Contd.)

- ✍ Respetar la atomicidad y no intentéis soluciones “inteligentes”
  - ✍ Por ejemplo una cadena de caracteres debe tener un significado indivisible.
  - ✍ Supongamos que en la base de datos de la universidad a los profesores se les asigna identificadores del tipo: LAN013
  - ✍ Donde los tres primeros caracteres se refieren al departamento en el que se encuentran.
  - ✍ Hacer esto es una idea muy mala puesto que la información se codifica a nivel de programa y no de base de datos.

# Segunda Forma Normal

- ✍ Cada atributo que no sea una clave primaria debe depender únicamente de esa (de toda la clave primaria)
- ✍ Para normalizar divide la tabla
- ✍ Por ejemplo
  - ✍ registro(estudiante\_id, estudiante\_nombre, curso\_id, curso\_nombre)
  - ✍ Satisface los requerimiento de 1FN con clave primaria (estudiante\_id, curso\_id)
  - ✍ nombre\_estudiante depende de estudiante\_id pero no de la pareja (estudiante\_id, curso\_id)
  - ✍ Divídase en tablas tres tablas
    - ✍ estudiante(estudiante\_id, estudiante\_nombre)
    - ✍ asignatura(curso\_id, curso\_nombre)
    - ✍ registro(estudiante\_id, asignatura\_id)

Ahora podemos tener datos de estudiantes que no estan matriculados en ninguna asignatura, asignaturas que no tienen ningún estudiante y no necesitamos repetir los datos de estudiante/asignatura para cada registro

# Segunda Forma Normal (cont)

- ✍ ASEGURAROS DE NO PERDER INFORMACION AL PARTIR LA RELACION EN VARIAS TABLAS

# Tercera Forma Normal

- ✍ Definición de dependencia transitiva: Un atributo depende transitivamente de la clave primaria si depende de otro atributo que a su vez depende de la clave
- ✍ La tercera forma normal elimina estas dependencias
- ✍ Por ejemplo
  - ✍ pedido(pedido\_id, fecha, cliente\_id, cliente\_nombre)
  - ✍ satisface 1FN y 2FN con clave primaria pedido\_id
  - ✍ pero cliente\_nombre cambia si cambia cliente\_id
  - ✍ Así que debemos dividir la tabla en:
    - ✍ pedido(pedido\_id, fecha, cliente\_id)
    - ✍ cliente(cliente\_id, cliente\_nombre)

# Resumen

El proceso de normalización consiste:

1. Comprueba que cada tabla tiene un número fijo de columnas y las variables son sencillas (atómicas)
2. Identifica la clave primaria
3. Comprueba que todos los atributos (menos la clave primaria) depende de TODA la clave no de PARTE de ella.
4. Si existe dependencia parcial rompe la relación en varias subrelaciones.
5. Comprueba que todos los atributos dependen de la clave y no de otros atributos (dependencias transitivas)
6. Si existe dependencias no relacionadas con la clave primaria subdivide las tablas

Y ya teneis vuestra base de datos en 3FN



# Normalización

## La normalización es:

- ✍ Fácil de entender a nivel concetual
- ✍ Todo lo que hay que hacer es romper las relaciones hasta que cada realación resultante sea coherente en si misma
- ✍ Tras normalizar aseguras:
  - ✍ Que no se pierda información cuando se borran campos
  - ✍ Que no se almacena información duplicada
- ✍ El desarrollo matemático de la teoría de Normalización es bastante complejo así que no profundizaremos en el.
- ✍ Existen 4,5 y 6 formas normales pero con que vuestras bases de datos satisfaga la 3FN es suficiente

# En Resumen

- ✍ 1NF significa que tus relaciones tienen un número de atributos fijos y atómicos
- ✍ 2NF significa que los atributos dependen de toda la clave primaria y no de parte de ella
- ✍ 3NF significa que los atributos dependen directamente de la clave primaria (y no indirectamente a través de otro atributo)

Estos tres puntos es todo lo que os hace falta recordar

## Ejemplo: relación no normalizada

### ORDER

Customer No: 001964                      Order Number: 00012345  
Name: Mark Campbell                      Order Date: 14-Feb-2002  
Address: 1 The House  
          Leytonstone  
          E11 9ZZ

Product Number	Product Description	Unit Price	Order Quantity	Line Total
T5060	Hook	5.00	5	25.00
PT42	Bolt	2.50	10	20.50
QZE48	Spanner	20.00	1	20.00

Order Total: 65.50

ORDER (order no, order date, cust no, cust name, cust add,  
          (prod-no, prod-desc, unit-price, ord-qty, line-total)\*, order total

# Primera Forma Normal (1NF)

**Definición:** Una relación está (o satisface) la 1NF, sii, todos sus atributos son atómicos.

**Crea una nueva relacion con los grupos que se repiten**

ORDER (order no, order date, cust no, cust name, cust add,  
(*prod-no, prod-desc, unit-price, ord-qty, line-total*)\*, order total

## Pasos para convertir una relacion en 1NF:

- ✍ Crea una nueva relación con el grupo que se repite
- ✍ Añade a esta nueva relación la clave primaria de la relación que originalmente la contenía
- ✍ Darle un nombre a la nueva entidad
- ✍ Determina la clave primaria de la nueva entidad
- ✍ Repetir hasta que no queden más atributos no atómicos

## Ejemplo - 1NF

ORDER (order-no, order-date, cust-no, cust-name, cust-add,  
(*prod-no, prod-desc, unit-price, ord-qty, line-total*)\*, order-total)

1. Crea una nueva relación con el grupo que se repite

(*renombra la entidad original añadiendo un 1*)


ORDER-1 (order-no, order-date, cust-no, cust-name, cust-add, order-total)

(*prod-no, prod-desc, unit-price, ord-qty, line-total*)

2. Añade a esta nueva relación la clave primaria de la relación que originalmente la contenía.

ORDER-1 (order-no, order-date, cust-no, cust-name, cust-add, order-total)

(*order-no*, prod-no, prod-desc, unit-price, ord-qty, line-total)



3. Darle un nombre a la nueva entidad (*que acabe en 1 para indicar 1NF*)

ORDER-LINE-1 (order-no, prod-no, prod-desc, unit-price, ord-qty, line-total)

4. Determina la clave primaria de la nueva entidad

ORDER-LINE-1 (order-no, prod-no, prod-desc, unit-price, ord-qty, line-total)

## Segunda Forma Normal (2NF)

**Definition:** Una relación está (o satisface) la 2NF, sii, satisface la 1NF y todo atributo que no forma parte de la clave primaria depende TOTALMENTE de la clave.

**Elimina dependencias funcionales**

### **Pasos para convertir una relacion 1NF a 2NF:**

- ✍ Elimina los atributos que dependen parcialmente de la clave primaria y crea con ellos una nueva relación.
- ✍ Añade a esta relación una copia del atributo/s del cual dependen (será la clave primaria de la nueva relación)
- ✍ Nombra a la nueva entidad (*añade un 2 para indicar 2NF*)
- ✍ Renombra a la entidad original (*añade un 2 para indicar 2NF*)

## Ejemplo - 1NF < 2NF

ORDER-LINE-1 (order-no, prod-no, prod-desc, unit-price, ord-qty, line-total)

1. Elimina los atributos que dependen parcialmente de la clave primaria y crea con ellos una nueva relación.

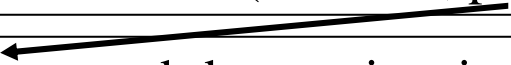
ORDER-LINE-1 (order-no, prod-no, ord-qty, line-total)

(prod-desc, unit-price)

2. Añade a esta relación una copia del atributo/s del cual dependen (será la clave primaria de la nueva relación)

ORDER-LINE-1 (order-no, prod-no, ord-qty, line-total)

(prod-no, prod-desc, unit-price)



3. Nombra a la nueva entidad (*añade un 2 para indicar 2NF*)

PRODUCT-2 (prod-no, prod-desc, unit-price)

4. Renombra a la entidad original (*añade un 2 para indicar 2NF*)

ORDER-LINE-2 (order-no, prod-no, ord-qty, line-total)

## Tercera Forma Normal (3NF)

**Definition:** Una relación está (o satisface) la 3NF si está en 2NF y todos los atributos que no son clave primaria dependen no-transitivamente de la clave primaria.

### Elimina dependencias transitivas

#### Pasos para convertir 2NF en 3NF:

- ✍ Elimina los atributos que presentan dependencias transitivas y crea una nueva relación con ellos
- ✍ Añade a esta nueva relación una copia de los atributos con los que están relacionados (son determinantes) los atributos eliminados. Estos atributos serán la clave primaria de la nueva relación.
- ✍ Nombra a la nueva entidad (*añade un 3 para indicar 3NF*)
- ✍ Renombra a la entidad original (*añade un 3 para indicar 3NF*)



## ejemplo - 2NF a 3NF

ORDER-2 (order-no, order-date, cust-no, cust-name, cust-add, order-total)

1. Elimina los atributos que presentan dependencias transitivas y crea una nueva relación con ellos

ORDER-2 (order-no, order-date, cust-no, order-total)

(cust-name, cust-add )

2. Añade a esta nueva relación una copia de los atributos con los que están relacionados (son determinantes) los atributos eliminados. Estos atributos serán la clave primaria de la nueva relación.

ORDER-2 (order-no, order-date, cust-no, order-total)

(cust-no, cust-name, cust-add )



3. Nombra a la nueva entidad (*añade un 3 para indicar 3NF*)

CUSTOMER-3 (cust-no, cust-name, cust-add )

4. Renombra a la entidad original (*añade un 3 para indicar 3NF*)

ORDER-3 (order-no, order-date, cust-no, order-total)

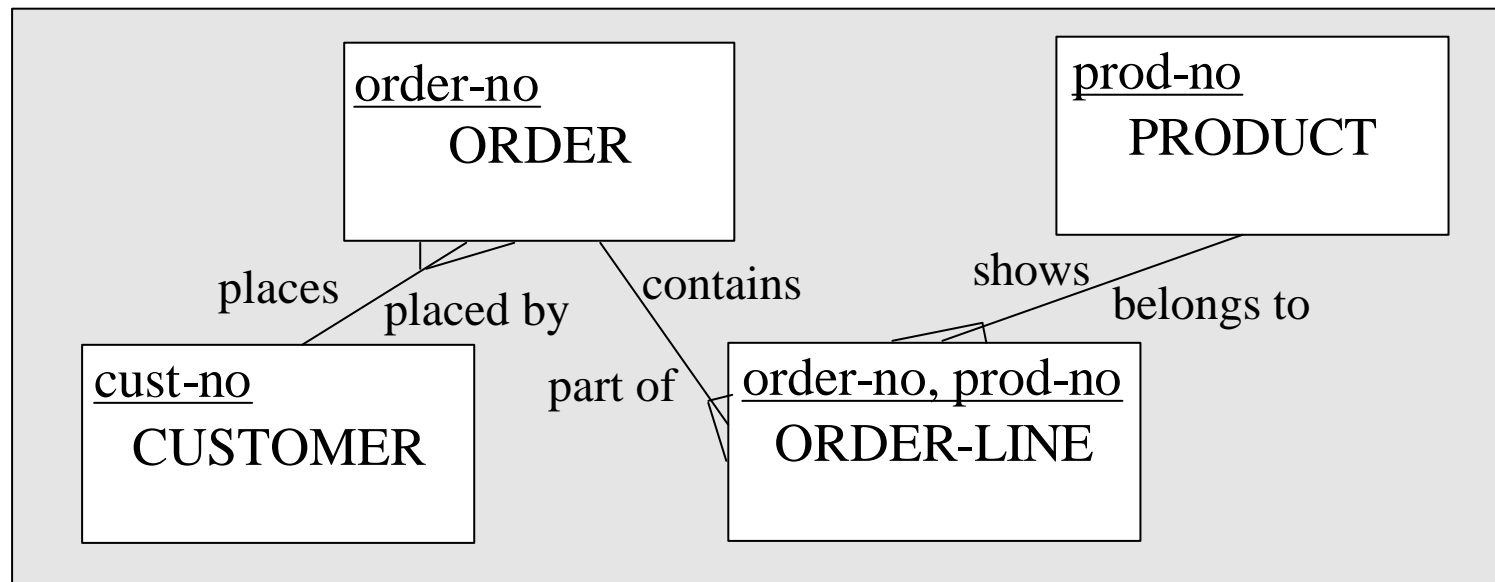
## Ejemplo - Relations en 3NF

ORDER-3 (order-no, order-date, cust-no, order-total)

CUSTOMER-3 (cust-no, cust-name, cust-add )

PRODUCT-2 (prod-no, prod-desc, unit-price)

ORDER-LINE-2 (order-no, prod-no, ord-qty, line-total)



# Chapuzas compatibles con la 3FN

✍ Empresa(empresa\_id, empresa\_nombre, beneficios\_2000, beneficios\_2001, beneficios\_2002);

# Normaliza: (1)

✍ vacacion(Lugar\_id, Lugar\_Nombre, cliente\_id, cliente\_Nombre, fecha)

✍ ¿Atributos atómicos?

✍ Sí, es 1FN

✍ ¿Cuál es la clave?

✍ vacacion(Lugar\_id, Lugar\_Nombre, cliente\_id, cliente\_Nombre, fecha)

✍ 2FN – ¿Todos los atributos (que no sean clave primaria) dependen de toda la clave?

✍ Lugar\_Nombre depende de Lugar\_id

✍ crea: Lugar\_2(Lugar\_id, Lugar\_Nombre)

✍ cliente\_Nombre depende de cliente\_id

✍ crea: cliente\_2 (cliente\_id, cliente\_Nombre)

✍ y nos queda: vacacion\_2 (Lugar\_id, cliente\_id, fecha)

✍ Ahora ya satisfacemos los requerimientos de la 2FN

✍ ¿Hay dependencia Transitivas?

✍ No hay dependencia transitivas así que satisface 3FN

# Normaliza: (2)

- ✍ reserva(habitacion\_id,fecha,cliente\_id,cliente\_Nombre)
- ✍ ¿Atributos atómicos?
  - ✍ Sí, es 1FN
- ✍ ¿Cuál es la clave?
  - ✍ reserva(habitacion\_id,fecha,cliente\_id,cliente\_Nombre)
- ✍ 2FN – ¿Todos los atributos (que no sean clave primaria) dependen de toda la clave?
  - ✍ cliente\_Nombre depende de cliente\_id
  - ✍ crear: cliente\_2 (cliente\_id, cliente\_Nombre)
  - ✍ reserva\_2 (habitacion\_id,fecha,cliente\_id,)
- ✍ Ahora ya satisfacemos los requerimientos de la 2FN
- ✍ ¿Hay dependencia Transitivas?
  - ✍ No hay dependencia transitivas así que satisface 3FN

# Normaliza: (3)

- ✍ asigna(empleado\_id, empleado\_Nombre, deptID, deptNombre)
- ✍ ¿Atributos atómicos?
  - ✍ Sí, es 1FN
- ✍ ¿Cuál es la clave?
  - ✍ asigna(empleado\_id, empleado\_Nombre, deptID, deptNombre)
- ✍ 2FN – ¿Todos los atributos (que no sean clave primaria) dependen de toda la clave
  - ✍ empleado\_Nombre depende solo de empleado\_id
  - ✍ crear: emp\_2 (empleado\_id, empleado\_Nombre)
  - ✍ deptNombre depende solo de deptID
  - ✍ crear: dept\_2 (deptID, deptNombre)
  - ✍ nos queda: asigna\_2 (empleado\_id, deptID)
- ✍ Ahora ya satisfacemos los requerimientos de la 2FN
- ✍ ¿Hay dependencia Transitivas?
  - ✍ No hay dependencia transitivas así que satisface 3FN

# Normaliza 4

- ✍ receta(receta\_ID,medicina,cantidad,cliente\_id,cliente\_Nombre)
- ✍ ¿Atributos atómicos?
  - ✍ Sí, es 1FN
- ✍ ¿Cuál es la clave? Asume una medicina por receta.
  - ✍ receta(receta\_ID,medicina,cantidad,cliente\_id,cliente\_Nombre)
- ✍ 2FN – ¿Todos los atributos (que no sean clave primaria) dependen de toda la clave
  - ✍ Si, es 2FN
- ✍ ¿Hay dependencia Transitivas?
  - ✍ cliente\_Nombre depende de cliente\_id
  - ✍ crear: cliente\_3 (cliente\_id, cliente\_Nombre)
  - ✍ nos queda: receta\_3 (receta\_ID,medicina,cantidad,cliente\_id)
- ✍ No hay más dependencias transitivas así que satisface 3FN

# Normaliza 5

- ✍ lineapedido(pedido\_ID, linea\_ID, producto\_ID, productoNombre, cantidad, cliente\_id, cliente\_Nombre)
- ✍ ¿Atributos atómicos?
  - ✍ Sí, es 1FN
- ✍ ¿Cuál es la clave?
  - ✍ lineapedido(pedido\_ID, linea\_ID, producto\_ID, productoNombre, cantidad, cliente\_id, cliente\_Nombre)
- ✍ 2FN – ¿Todos los atributos (que no sean clave primaria) dependen de toda la clave
  - ✍ cliente\_id y cliente\_Nombre dependen solo de pedido\_ID
  - ✍ crear: pedido\_2 (pedido\_ID, cliente\_id, cliente\_Nombre)
  - ✍ nos queda: lineapedido\_2 (pedido\_ID, linea\_ID, producto\_ID, productoNombre, cantidad)
- ✍ ¿Hay dependencia Transitivas?
  - ✍ cliente\_Nombre depende de cliente\_id
  - ✍ crear: cliente\_3(cliente\_id, cliente\_Nombre)
  - ✍ productoNombre depende de producto\_ID
  - ✍ crear: producto\_3 (producto\_ID, productoNombre)
  - ✍ nos queda: lineapedido\_3(pedido\_ID, linea\_ID, producto\_ID, cantidad)
  - ✍ y: pedido\_3 (pedido\_ID, cliente\_id)
- ✍ No hay más dependencias transitivas así que satisface 3FN



# Resumen

**El resumen de este capítulo es: Todos los atributos de una relación deben ser atómicos y depender única y totalmente de la clave primaria.**

**La Clave Primaria, toda la clave primaria y nada más que la clave primaria**