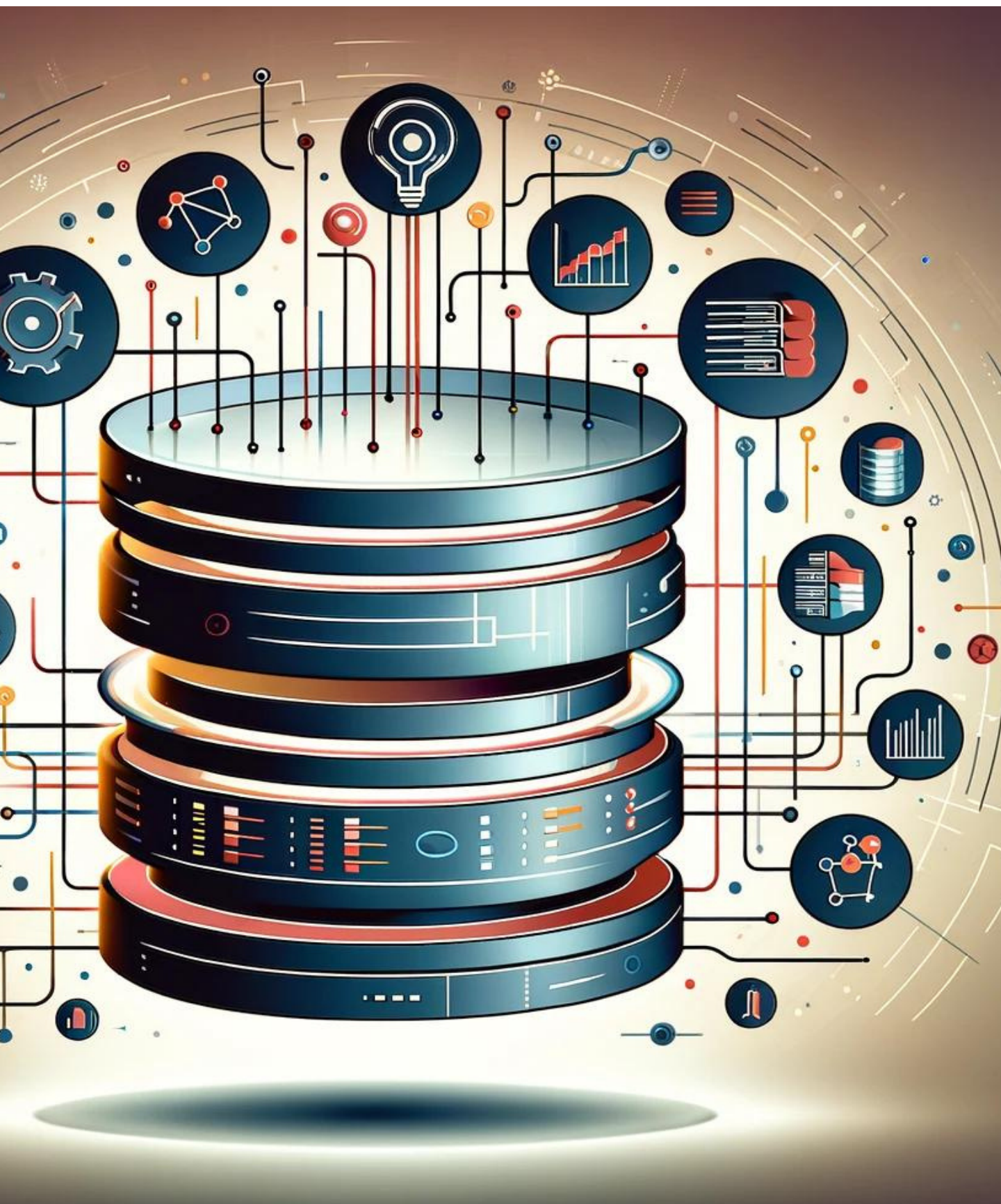




Bases de Datos 1



Alejandra Lliteras
Prof. Titular



Federico Orlando
Prof. Adjunto



TEMAS GENERALES

Bases de Datos 1

1

Modelo de
Datos

2

Teoría de
Diseño de
Bases de
Datos

3

Álgebra
Relacional

4

DBMS
Relacional
MySQL

5

Visualización
de Datos

TEMAS Y SUBTEMAS

Vimos...

Teoría de Diseño de Bases de Datos Relacionales

Conceptos Generales:

- Anomalía
- Dependencia Funcional
- Dependencia Funcional Trivial

TEMAS Y SUBTEMAS

Hoy veremos...

Teoría de Diseño de Bases de Datos Relacionales

Conceptos Generales:

- Clave de una relación
- Clave candidata
- Superclave
- Axiomas de Armstrong
- Clausura de un conjunto de atributos

Normalización:

- Descomposición
- Pérdida de Información
- Pérdida de Dependencias Funcionales
- Formas Normales
 - Boyce y Codd

Los atributos $\{A_1, A_2, \dots, A_n\}$ son la clave de una relación R si cumplen:

- $\{A_1, A_2, \dots, A_n\}$ determinan funcionalmente a todos los restantes atributos de la relación R

Clave de
una
relación

Los atributos $\{A_1, A_2, \dots, A_n\}$ son la clave de una relación R si cumplen:

- $\{A_1, A_2, \dots, A_n\}$ determinan funcionalmente a todos los restantes atributos de la relación R
- No existe un subconjunto de $\{A_1, A_2, \dots, A_n\}$ que determine funcionalmente a todos los restantes atributos de la relación R
 - Esto implica que una clave es un conjunto **minimal**-

Clave de una relación

Teoría de Diseño de B B D D Relacionales

- Clave de una relación

 Ejemplo

—PERSONA(dni, nombre, edad, fechaNac)

df1: dni->nombre,edad,fechaNac

- **Clave: {dni}**

En caso de existir dos o más conjuntos de atributos $\{A_1, A_2, \dots, A_n\}$, $\{B_1, B_2, \dots, B_k\}$, ... $\{N_1, N_2, \dots, N_m\}$ en una relación R tales que

$\{A_1, A_2, \dots, A_n\}$ determinan funcionalmente a todos los restantes atributos de la relación R

$\{B_1, B_2, \dots, B_k\}$, ...y $\{N_1, N_2, \dots, N_m\}$ también por sí mismos determinan al resto de los atributos de R

No existe un subconjunto de $\{A_1, A_2, \dots, A_n\}$ o $\{B_1, B_2, \dots, B_k\}$, ...o $\{N_1, N_2, \dots, N_m\}$ que determine funcionalmente a todos los atributos de R

Entonces $\{A_1, A_2, \dots, A_n\}$, $\{B_1, B_2, \dots, B_k\}$, ... $\{N_1, N_2, \dots, N_m\}$ son CLAVES CANDIDATAS para la relación R

Clave candidata de una relación

Teoría de Diseño de BDD Relacionales

• Clave de una relación /Clave candidata

 Ejemplo:

—Dada la relación: PERSONA(dni, nombre, edad, fechaNacimiento, nroLegajo)

Donde

- *Una persona posee un único número de legajo asignado*
- *Un número de legajo pertenece a una sola persona*

— Se pueden enunciar las siguientes dfs

df1) dni -> nombre, edad, fechaNac, nroLegajo

df2) nroLegajo -> nombre, edad, fechaNac, dni

¿Clave o claves candidatas?

Clave candidata 1 (cc1): {dni}

Clave candidata 2 (cc2): {nroLegajo}

Teoría de Diseño de BDD Relacionales

- **Clave de una relación /Clave candidata**

 Ejemplo:

–Dada la relación: PERSONA(dni, nombre, edad, fechaNacimiento, nroLegajo, carrera)

» Donde

- *Una persona puede cursar diversas carreras*
- *Nombre indica como se llama la persona*
- *Una persona posee un único número de legajo asignado para cada carrera que cursa*
- *Un número de legajo pertenece a una sola persona de una carrera*

df1) dni -> nombre, edad, fechaNac

df2) nroLegajo, carrera -> dni

df3) dni, carrera -> nroLegajo

¿Clave o claves candidatas?

Clave candidata 1 (cc1): {nroLegajo, carrera }

Clave candidata 2 (cc2): {dni, carrera }

Los atributos $\{A_1, A_2, \dots, A_n\}$ son la superclave de una relación R si cumplen:

$\{A_1, A_2, \dots, A_n\}$ determinan funcionalmente a todos los restantes atributos de la relación R

Notar que:

- Una clave está contenida en una superclave
- Una superclave no necesariamente es minimal (como lo es la clave por la segunda condición de su definición)

Super Clave de una relación

Teoría de Diseño de B B D D Relacionales

- Superclave de una relación



–PERSONA(dni, nombre, edad, fechaNacimiento)

df1: dni->nombre,edad,fechaNac

- **superclave: {dni, nombre}**



Ejercicio Práctico

Actividades

- Determinar la o las claves candidatas
- Hallar las dependencias funcionales

Teoría de Diseño de B B D D Relacionales

● Ejercicio A

Dada la relación

VENTAS(codCliente, nombre, codVenta, monto)

—Donde:

- Un cliente realiza muchas compras
 - Una compra es realizada por un solo cliente
 - El monto representa el valor total de la compra realizada por un cliente
- Determinar las dependencias funcionales (dfs) y clave o clave candidatas válidas en VENTAS



...trabajando...

Teoría de Diseño de B B D D Relacionales

VENTAS(codCliente, nombre, codVenta, monto)

– Donde:

- Un cliente realiza muchas compras
- Una compra es realizada por un solo cliente
- El monto representa el valor total de la compra realizada por un cliente

– Determinar las dependencias funcionales (dfs) válidas en VENTAS

df1) codCliente -> nombre

df2) codVenta -> codCliente, monto

Clave candidata (codVenta)

Teoría de Diseño de B B D D Relacionales

● Ejercicio B

Dada la relación

VENTAS1(codCliente, nombreCliente, codVenta, monto)

—Donde:

- Un cliente realiza muchas compras
- Una compra puede ser realizada por más de un cliente
- El monto representa el valor total de la compra realizada por cada cliente

—Determinar las dependencias funcionales (dfs) y clave o claves candidatas válidas en VENTAS1



...trabajando...

Teoría de Diseño de BDD Relacionales

Ejercicio

Dada la relación

VENTAS1(codCliente, nombre, codVenta, monto)

– Donde:

- Un cliente realiza muchas compras
- Una compra puede ser realizada por más de un cliente
- El monto representa el valor total de la compra realizada por cada cliente

– Determinar las dependencias funcionales (dfs) válidas en VENTAS1

df1) codCliente -> nombre

df2) codVenta, codCliente -> monto

Clave candidata (codVenta, codCliente)

Teoría de Diseño de BBDD Relacionales

● Ejercicio C

Dada la relación

PERSONAEMPLEADA(dni, nombre, domicilio, depto, flngDepto, codEmpDepto, jefe)

Donde

- *En cada departamento hay un jefe para todos los empleados. Un mismo jefe puede estar asignado a más de un departamento*
- *Una persona puede trabajar en más de un departamento y en cada uno de ellos puede tener un código de empleado diferente*
- *El código de empleado no se repite para un mismo departamento, puede repetirse en diferentes departamentos*
- *Domicilio indica el lugar en el que vive una persona. Mas de una persona pueden vivir en el mismo domicilio*
- *Nombre indica la forma en la que se llama una persona. Notar que diferentes personas pueden llamarse igual*
- *flngDepto es la fecha de ingreso a un departamento*

Determinar las dependencias funcionales y clave o claves candidatas válidas en **PERSONAEMPLEADA**



...trabajando...

Teoría de Diseño de BBDD Relacionales

PERSONAEMPLEADA(dni, nombre, domicilio, codDepto, flngDepto, codEmpDepto, jefe)

Donde

- *En cada departamento hay un jefe para todos los empleados. Un mismo jefe puede estar asignado a más de un departamento*
- *Una persona puede trabajar en más de un departamento y en cada uno de ellos puede tener un código de empleado diferente*
- *El código de empleado no se repite para un mismo departamento, puede repetirse en diferentes departamentos*
- *Domicilio indica el lugar en el que vive una persona. Mas de una persona pueden vivir en el mismo domicilio*
- *Nombre indica la forma en la que se llama una persona. Notar que diferentes personas pueden llamarse igual*
- flngDepto es la fecha de ingreso a un departamento

Determinar las dependencias funcionales válidas en **PERSONAEMPLEADA**

df1) dni -> nombre, domicilio

df2) codDepto -> jefe

df3) codDepto, codEmpDepto -> dni, flngDepto

df4) codDepto, dni -> codEmpDepto, flngDepto

Teoría de Diseño de BDD Relacionales

Ejercicio

Dada la relación

PERSONAEMPLEADA(dni, nombre, domicilio, codDepto, flngDepto, codEmpDepto, jefe)

Determinar las dependencias funcionales válidas en **PERSONAEMPLEADA**

df1) dni -> nombre, domicilio

df2) codDepto -> jefe

df3) codDepto, codEmpDepto -> dni, flngDepto

df4) codDepto, dni -> codEmpDepto, flngDepto

Claves candidatas

Cc1: (codDepto, codEmpDepto)

Cc2: (codDepto, dni)

Teoría de Diseño de B B D D Relacionales

Ejercicio D

Dada la relación

PERSONAEMPLEADA1(dni, nombre, domicilio, depto, flngDepto, codEmpDepto, jefe)

Donde

- *Cada persona en un departamento tiene asignado a un jefe. El mismo jefe puede estar asignado a diferentes personas de un departamento o de diversos departamentos*
- *Una persona puede trabajar en más de un departamento y en cada uno de ellos puede tener un código de empleado diferente*
- *El código de empleado no se repite para un mismo departamento, puede repetirse en diferentes departamentos*
- *Domicilio indica el lugar en el que vive una persona. Mas de una persona pueden vivir en el mismo domicilio*
- *Nombre indica la forma en la que se llama una persona. Notar que diferentes personas pueden llamarse igual*
- *flngDepto es la fecha de ingreso a un departamento*

Determinar las dependencias funcionales y clave o clave candidatas válidas en **PERSONAEMPLEADA1**



...trabajando...

Teoría de Diseño de BBDD Relacionales

PERSONAEMPLEADA1(dni, nombre, domicilio, depto, flngDepto, codEmpDepto, jefe)

Donde

- *Cada persona en un departamento tiene asignado a un jefe. El mismo jefe puede estar asignado a diferente personas de un departamento o de diversos departamentos*
- *Una persona puede trabajar en mas de un departamento y en cada uno de ellos puede tener un código de empleado diferente*
- *El código de empleado no se repite para un mismo departamento, puede repetirse en diferentes departamentos*
- *Domicilio indica el lugar en el que vive una persona. Mas de una persona pueden vivir en el mismo domicilio*
- *Nombre indica la forma en la que se llama una persona. Notar que diferentes personas pueden llamarse igual*
- *flngDepto es la fecha de ingreso a un departamento*

Determinar las dependencias funcionales válidas en **PERSONAEMPLEADA1**

df1) dni -> nombre, domicilio

df2) codDepto, codEmpDepto -> dni, jefe, flngDepto

df3) codDepto, dni -> codEmpDepto, jefe, flngDepto

Teoría de Diseño de B B D D Relacionales

Ejercicio

Dada la relación

PERSONAEMPLEADA1(dni, nombre, domicilio, depto, flngDepto, codEmpDepto, jefe)

Dependencias funcionales:

df1) dni -> nombre, domicilio

df2) codDepto, codEmpDepto -> dni, jefe, flngDepto

df3) codDepto, dni -> codEmpDepto, jefe, flngDepto

Claves candidatas:

Cc1: (codDepto, codEmpDepto)

Cc2: (codDepto, dni)

Teoría de Diseño de BDD Relacionales

● Ejercicio E

–EMPRESA(idGerente, nombreGerente, idEmpleado, nombreEmpleado, idEmpleadoMaestranza)

Donde:

- El id de gerente es único
- El id del empleado es único
- Cada empledo responde a un único gerente
- El idEmpleadoMaestranza representa el identificador de cada uno de los empleados de maestranza de la empresa, de los cuales se saben son muchos.



...trabajando...

Teoría de Diseño de BDD Relacionales

- Ejemplo
 - EMPRESA(idGerente, nombreGerente, idEmpleado, nombreEmpleado, idEmpleadoMaestranza)
 - Donde:
 - El id de gerente es único
 - El id del empleado es único
 - Cada empleado responde a un único gerente
 - El idEmpleadoMaestranza representa el identificador de cada uno de los empleados de maestranza de la empresa, de los cuales se saben son muchos.

Dependencias funcionales válidas en EMPRESA:

df1) idGerente -> nombreGerente

df2) idEmpleado -> nombreEmpleado, idGerente

Teoría de Diseño de BDD Relacionales

- Ejemplo

- EMPRESA(idGerente, nombreGerente, idEmpleado, nombreEmpleado, idEmpleadoMaestranza)

- Donde:

- El id de gerente es único
 - El id del empleado es único
 - Cada empleo responde a un único gerente
 - El idEmpleadoMaestranza representa el identificador de cada uno de los empleados de maestranza de la empresa, de los cuales se saben son muchos.

Dependencias funcionales válidas en EMPRESA:

df1) idGerente -> nombreGerente

df2) idEmpleado -> nombreEmpleado, idGerente

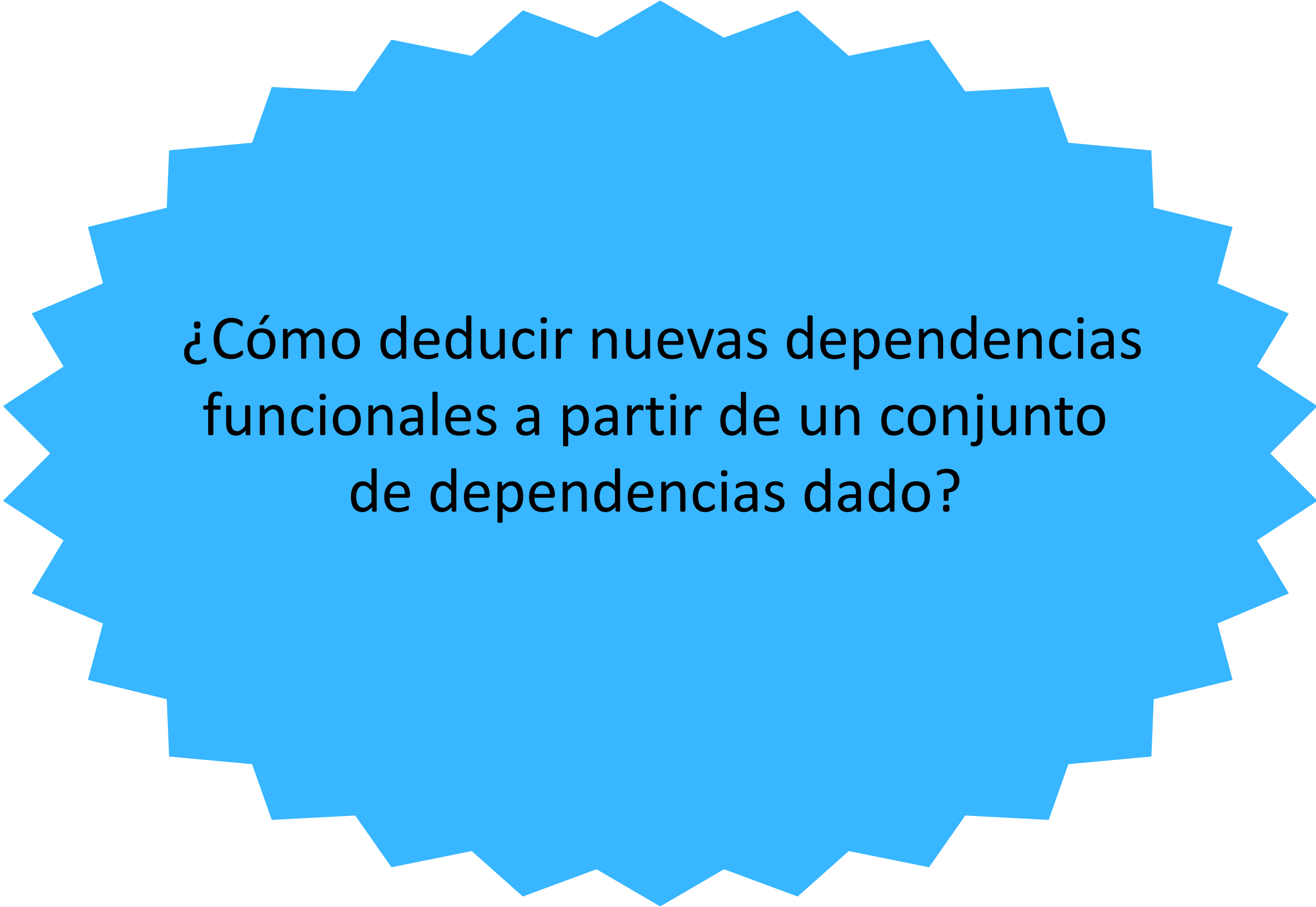
Claves Candidatas en EMPRESA

cc1: (idEmpleado, idEmpleadoMaestranza)

Teoría de diseño de BDD Relacionales

Axiomas de Armstrong



A blue starburst shape with a jagged, multi-pointed border, centered on a white background. Inside the starburst, the text is written in a black, sans-serif font.

¿Cómo deducir nuevas dependencias
funcionales a partir de un conjunto
de dependencias dado?

Axiomas de Armstrong

- Permiten inferir nuevas dependencias funcionales dado un conjunto base que resultó evidente
- Aplicándolos halla un conjunto completo y seguro donde todas las dependencias funcionales halladas son correctas
- Al generar todas las dependencias funcionales algunas son triviales

Axiomas de Armstrong

- Axiomas Básicos
 - Reflexión
 - Aumento
 - Transitividad
- Axiomas que se deducen a partir de los básicos
 - Unión
 - Descomposición
 - Pseudotransitividad

Teoría de Diseño de B B D D Relacionales

- Axiomas de Armstrong

Reflexión

X es un conjunto de atributos

y

$Y \subseteq X$ entonces $X \rightarrow Y$

Sabemos que $a \rightarrow a$, luego se puede decir que $a, b \rightarrow a$

Demostración:

Si $Y \subseteq X$ y existen dos tuplas diferentes de R tales que $t1[x]=t2[x]$ por definición de dependencia funcional $t1[y]=t2[y]$

Teoría de Diseño de B B D D Relacionales

- **Axiomas de Armstrong**



Aumento

Si $X \rightarrow Y$;
Z es un conjunto de atributos,
entonces
 $Z, X \rightarrow Z, Y$

Demostración:

Asumamos que $X \rightarrow Y$ vale pero $X, Z \rightarrow Y, Z$ no vale

Si $X \rightarrow Y$ entonces cada vez que

1) $t1[x] = t2[x]$ implica

2) $t1[y] = t2[y]$

Por otro lado, cada vez que

3) $t1[x, z] = t2[x, z]$ implica

4) $t1[y, z] \neq t2[y, z]$

De 1) y 3) se deduce $t1[z] = t2[z]$

De 2) y 4) se deduce que $t1[y, z] = t2[y, z]$

Teoría de Diseño de B BDD Relacionales

- **Axiomas de Armstrong**



Transitividad

Si $X \rightarrow Y$;
 $Y \rightarrow Z$,
entonces $X \rightarrow Z$

Demostración:

1) $X \rightarrow Y$

2) $Y \rightarrow Z$

$t1[x]=t2[x]$ implica por 1)

$t1[y]=t2[y]$ implica por 2)

$t1[z]=t2[z]$ entonces

$X \rightarrow Z$

Teoría de Diseño de BDD Relacionales

- **Axiomas de Armstrong**



Unión

Si $X \rightarrow Y$;
 $X \rightarrow Z$,
entonces $X \rightarrow Y, Z$

Demostración:

1) $X \rightarrow Y$

2) $X \rightarrow Z$

Si $X \rightarrow Y$, por aumentación vale que $X \rightarrow XY$

Si $X \rightarrow Z$, por aumentación vale que $X, Y \rightarrow Y, Z$

Luego por transitividad, $X \rightarrow Y, Z$

Teoría de Diseño de B B D D Relacionales

- **Axiomas de Armstrong**



Descomposición

Si $X \rightarrow Y, Z$

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

Demostración:

$X \rightarrow Y, Z$

por reflexividad vale que $Y, Z \rightarrow Y$

Luego, por transitividad $X \rightarrow Y$

Por reflexividad también vale que $Y, Z \rightarrow Z$

Luego por transitividad, también vale que $X \rightarrow Z$

Teoría de Diseño de B B D D Relacionales

- **Axiomas de Armstrong**

Pseudotransitividad

Si $X \rightarrow Y$; $Y, Z \rightarrow W$ entonces $X, Z \rightarrow W$

Demostración:

$X \rightarrow Y$

por aumento vale que $X, Z \rightarrow Y, Z$

Por otro lado, se sabe que $Y, Z \rightarrow W$

Luego por transitividad, vale que $X, Z \rightarrow W$

Teoría de diseño de BDD Relacionales

Clausura de un conjunto de atributos



¿Cómo se puede comprobar que un conjunto de atributos $\{A_1, A_2, \dots, A_n\}$ de una relación R permite recuperar al resto de los atributos de dicha relación?

- sea **F** un conjunto de dependencias funcionales sobre un esquema R y,
- sea **X** un subconjunto de R.

La clausura del conjunto de atributos X respecto de F, se denota X^+ y es el conjunto de atributos **A** tal que la dependencia $X \rightarrow A$ puede deducirse a partir de F, por los axiomas de Armstrong

Es decir, X^+ son todos los atributos determinados por X en R

Clausura de un conjunto de atributos

$$X^+$$

```
Result := X
While (hay cambios en result)
do
  For (cada dependencia
    funcional  $Y \rightarrow Z$  en  $F$ ) do
    if ( $Y \subseteq \text{result}$ ) then
      result := result  $\cup$  Z
```

sea F un conjunto de dependencias funcionales sobre un esquema R y,
sea X un subconjunto de R .

Clausura de
un conjunto
de atributos

X^+

Algoritmo para
hallar X^+

A blue starburst shape with a jagged, multi-pointed border, centered on a white background. Inside the starburst, there is black text.

Algoritmo para hallar X^+

¿Cómo funciona?

Teoría de Diseño de BDD Relacionales

- Ejemplo

Dada la relación: PERSONA(dni, nombre, edad, fechaNacimiento, nroLegajo, carrera)

Ejemplo ya visto

Donde

- *Una persona puede cursar diversas carreras*
- *Nombre indica como se llama la persona*
- *Una persona posee un único número de legajo asignado para cada carrera que cursa*
- *Un número de legajo pertenece a una sola persona de una carrera*

df1) dni -> nombre, edad, fechaNac

df2) nroLegajo, carrera -> dni

df3) dni, carrera -> nroLegajo

Clave candidata 1 (cc1): {nroLegajo, carrera }

Clave candidata 2 (cc2): {dni, carrera }

Teoría de Diseño de BDD Relacionales

- Ejemplo

- Dada la relación: PERSONA(dni, nombre, edad, fechaNacimiento, nroLegajo, carrera)

- df1) dni \rightarrow nombre, edad, fechaNac

- df2) nroLegajo, carrera \rightarrow dni

- df3) dni, carrera \rightarrow nroLegajo

Clave candidata 1 (cc1): {nroLegajo, carrera }

Clave candidata 2 (cc2): {dni, carrera }

Por ejemplo, podríamos preguntarnos: ¿Es cierto que a partir de los atributos de cc1, puedo recuperar los atributos restantes de PERSONA?

- **Para ello podemos ejecutar el algoritmo de X^+ instanciándolo con la información de PERSONA**



Ejemplo ya visto

Teoría de Diseño de BBDD Relacionales

Result:= X

While (hay cambios en result) do

For (cada dependencia funcional $Y \rightarrow Z$ en F) do

if ($Y \subseteq \text{result}$) then

result := result \cup Z

PERSONA(dni, nombre, edad, fechaNacimiento, nroLegajo,
carrera)

Hallar (nroLegajo, carrera)⁺

Result= (nroLegajo, carrera)

Paso 1) Tomamos la dep. fun: dni -> nombre, edad, fechaNac, {dni} **no está incluido en result, no agrego nada a result =>** (nroLegajo, carrera)

Paso 2) Tomamos la dep. fun. nroLegajo, carrera -> dni , {nroLegajo, carrera} **está incluido en result, agrego {dni} a result =>** (nroLegajo, carrera, dni)

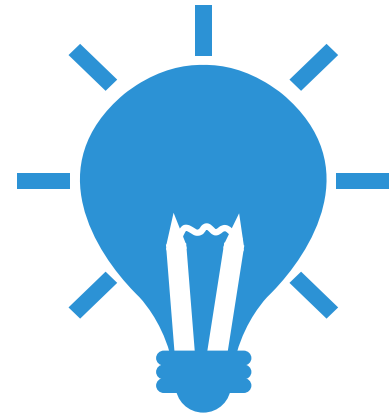
Paso 3) Tomamos la dep. fun. dni, carrera -> nroLegajo, {dni, carrera} **está incluido en result, agrego {nroLegajo} a result =>** (nroLegajo, carrera, dni)

Como ya recorrí todas las dependencias funcionales y result cambió vuelvo a iterar

Paso 1) Tomamos la dep. fun. dni -> nombre, edad, fechaNac, {dni} **está incluido en result, agrego {nombre, edad, fechaNac} a result =>** (nroLegajo, carrera, dni , nombre, edad, fechaNac)

RECUPERE A TODOS LOS ATRIBUTOS DE PERSONA!

Teoría de Diseño de B B D D Relacionales



- De la misma manera podríamos haber probado con la otra clave candidata
- Recordar que este algoritmo **NO** asegura que el **conjunto de atributos de partida sea mínimo**

Teoría de Diseño de BDD Relacionales

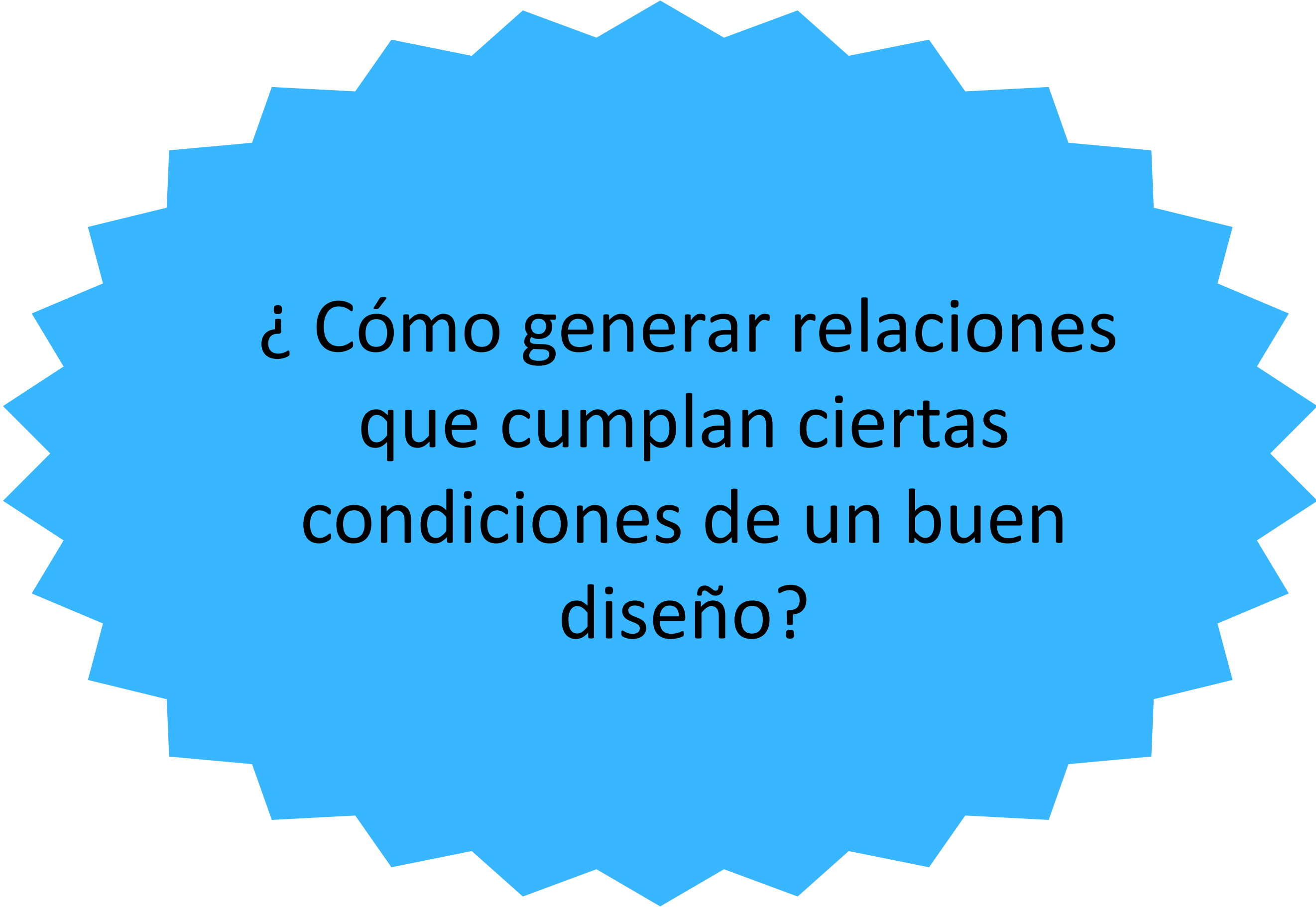
- Hasta ahora vimos, para una relación R
 - **Cómo hallar la o las claves candidatas**
 - Y como usar el algoritmo de la clausura de atributos para corroborar que a partir de un subconjunto de atributos de R se puede recuperar al resto de los atributos de la relación (aunque éste no asegura que dicho subconjunto sea mínimo)
 - **Cómo hallar dependencias funcionales**
 - Y su conjunto completo mediante los Axiomas de Armstrong
- **A continuación:**
 - Considerando las dependencias funcionales y las claves candidatas, veremos conceptos necesarios para realizar un **proceso que permite generar relaciones** que cumplan ciertas condiciones de un buen diseño (quitando anomalías) con el fin de **normalizar** esquemas

Teoría de diseño de BDD Relacionales

Normalización de esquemas de relación.

Algunos conceptos.



A blue starburst shape with a jagged, multi-pointed border, centered on a white background. Inside the starburst, the text is written in a black, sans-serif font.

¿ Cómo generar relaciones
que cumplan ciertas
condiciones de un buen
diseño?

Teoría de Diseño de B B D D Relacionales

Descomposición o particionamiento de un esquema

- Es una forma aceptada de eliminar las anomalías de una relación
- Consiste en separar los atributos de una relación en dos nuevas relaciones (bajo ciertos criterios)
- Al particionar, no se debe perder
 - Información
 - Dependencias funcionales

Dado un esquema **R**

*donde vale una dependencia
funcional $X \rightarrow Y$*

R se descompone/particiona en

R1(X, Y)

R2(R-Y)

Descomposición
de un esquema

Al descomponer, no se debe perder:

- Información
- Dependencias funcionales
 - Validación simple
 - Validación formal mediante un algoritmo

Descomposición
de un esquema

Descomposición de un esquema

Pérdida de
Información

Si a un esquema **R**, se lo particiona en dos subesquemas **R1** y **R2**

entonces, se debe cumplir alguna de las siguientes condiciones:

- **$R1 \cap R2$** clave en el esquema **R1**
o bien,
- **$R1 \cap R2$** clave en el esquema **R2**

Al particionar, se debe verificar que cada una de las dependencias funcionales que valían en el esquema **R**, sigan valiendo en alguna de las particiones **R_i**.

Cuando se chequean las dependencias funcionales pueden ocurrir dos cosas:

- los atributos de la dependencia funcional original quedaron todos incluidos en alguna de las particiones generadas
=> Validación simple
- los atributos de la dependencia funcional original quedaron distribuidos en más de una partición
=> Validación formal mediante un algoritmo

Descomposición
de un esquema

Pérdida de
dependencias
funcionales

Al particionar, se debe verificar que cada una de las dependencias funcionales que valían en el esquema **R**, sigan valiendo en alguna de las particiones **R_i**.

Cuando se chequean las dependencias funcionales pueden ocurrir dos cosas:

- los atributos de la dependencia original quedaron totales en las particiones

=> 1/

- los atributos de la dependencia original no quedaron totales en las particiones



Estos conceptos serán retomados y ampliados más adelante en esta clase y en las venideras.

Verificación formal mediante un algoritmo

Descomposición
de un esquema

Pérdida de
dependencias
funcionales

Teoría de diseño de BDD Relacionales

Normalización de esquemas de relación.

Formas Normales



- Criterio para determinar grado de vulnerabilidad a inconsistencias y anomalías
- Existen diferentes formas normales
- Al lograr aplicar una mayor forma normal se logrará menor vulnerabilidad

Forma
Normal
(FN)

Teoría de Diseño de BDD Relacionales

Algunas Formas Normales

- Primera Forma Normal (1FN)
- Segunda Forma Normal (2FN)
- Tercera Forma Normal (3FN)
- Forma Normal de Boyce y Codd

Se determinan a partir
de las dependencias
funcionales

El esquema no debe
tener atributos
polivalentes o
compuestos

- Conocida por su acrónimo en inglés de BCNF
- Particionar llevando un esquema a esta FN, asegura que:
 - las anomalías dejan de estar (sólo puede quedar redundancia),
 - que no se pierda información y,
 - en algunos casos, asegura que no se pierdan dependencias funcionales

Forma
Normal
De
Boyce
y
Codd
(FNBC)
(BCNF)

Un esquema de relación está en BCNF sí, siempre que una dependencia funcional de la forma $X \rightarrow A$ es válida en R, entonces se cumple que:

- X es superclave de R
o bien
- $X \rightarrow A$ es una dependencia funcional trivial

Forma
Normal
De
Boyce
y
Codd
(FNBC)
(BCNF)

Teoría de Diseño de B B D D Relacionales

- **Hasta ahora vimos que**
 - Las formas normales se usan para sacar anomalías
 - A mayor forma normal, menos vulnerabilidad
- **A continuación**
 - Intentaremos llevar, en principio, un esquema a la Forma Normal de Boyce y Codd para normalizarlo

Para normalizar un esquema (en principio) y propiciar un buen diseño, vamos a valernos de:

- las dependencias funcionales del esquema
 - las claves candidatas
 - la definición de BCNF
- la descomposición o particionamiento del esquema

Para normalizar un esquema (es decir, para propiciar un buen diseño, y

- las dependencias

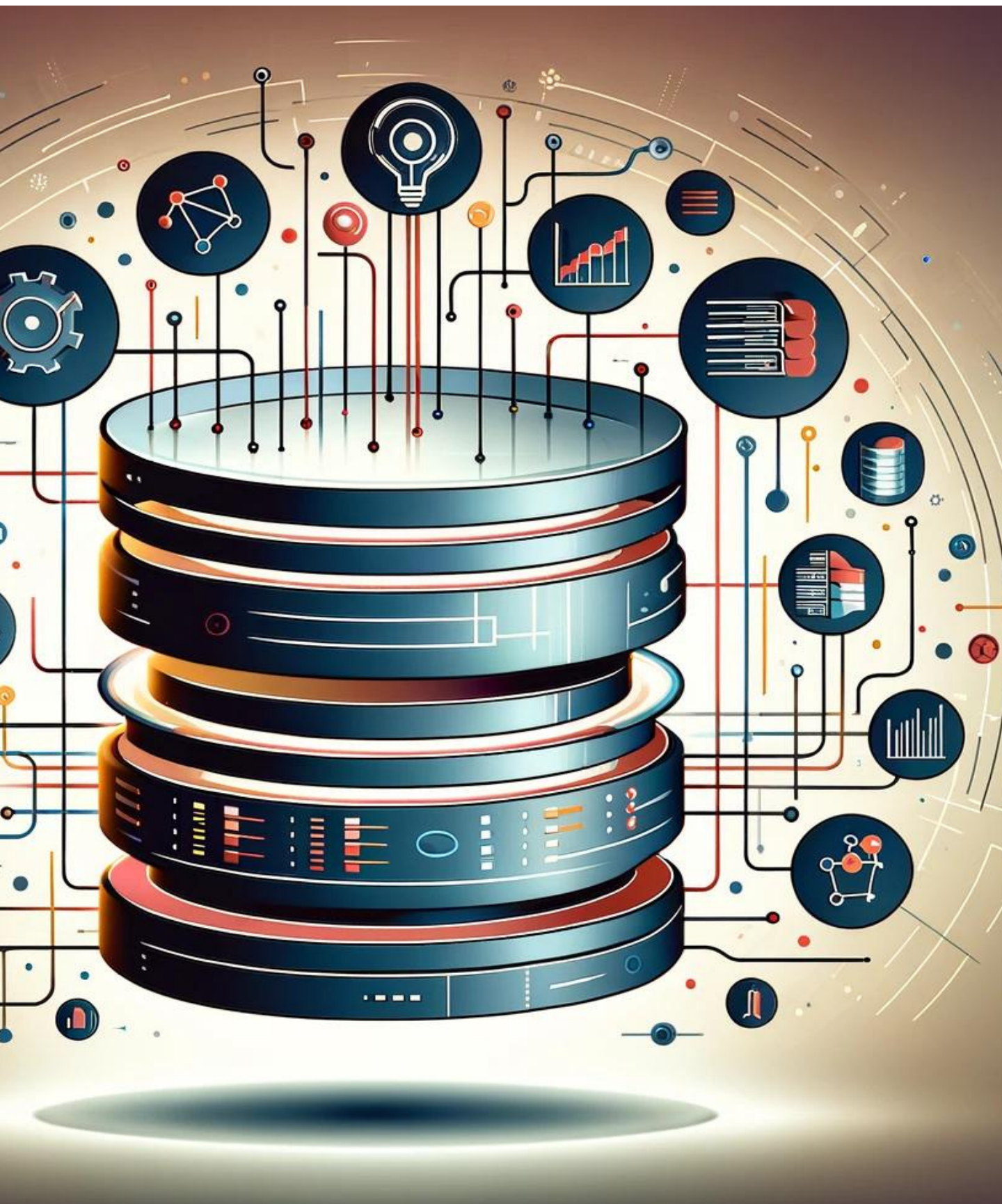
Próxima clase teórica

Importante: consolidar los conceptos vistos hasta esta clase relacionados al diseño para trabajar en la siguiente clase

BCNF

particionamiento del
esquema





Bibliografía de la clase

Bibliografía

- Date, C. J. (2019). *Database design and relational theory: normal forms and all that jazz*. Apress.
- Garcia-Molina, H. (2008). *Database systems: the complete book*. Pearson Education India.
- Ullman, J. D. (1988). Principles of database and knowledge-base systems.
- Albarak, M., Bahsoon, R., Ozkaya, I., & Nord, R. L. (2020). Managing Technical Debt in Database Normalization. *IEEE Transactions on Software Engineering*.
- Jadhav, R., Dhabe, P., Gandewar, S., Mirani, P., & Chugwani, R. (2020). A New Data Structure for Representation of Relational Databases for Application in the Normalization Process. In *Machine Learning and Information Processing* (pp. 305-316). Springer, Singapore.
- Ghawi, R. (2019, May). Interactive Decomposition of Relational Database Schemes Using Recommendations. In *International Conference: Beyond Databases, Architectures and Structures* (pp. 97-108). Springer, Cham.
- Stefanidis, C., & Koloniari, G. (2016, November). An interactive tool for teaching and learning database normalization. In *Proceedings of the 20th Pan-Hellenic Conference on Informatics* (pp. 1-4).
- Knowledge Base of Relational and NoSQL Database Management Systems https://db-engines.com/en/ranking_trend
- Akhtar, A. (2023). Popularity Ranking of Database Management Systems. *arXiv preprint arXiv:2301.00847*.



Importante!
Los slides usados en las clases teóricas de esta materia, no son material de estudio por sí solos en ningun caso.