

Diseño de bases de datos Relacionales

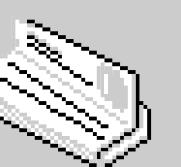
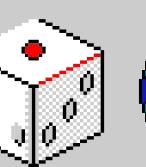
Bases de Datos I CCOMP4-1



Integrantes



Baca Flores Gabriel
Briceño Quiroz Anthony
Calle Rodriguez Valeria
Dueñas Flores Lucia
Vilca Montesinos Rodrigo
Vizcarra Vargas Dayana



11:11PM

Introducción:

Cap. 7: Relational Data Base Design

Sección 7.1

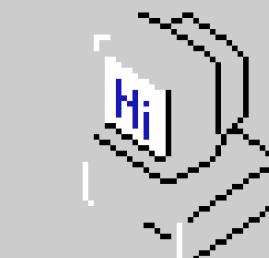
Sección 7.2

Sección 7.3

Sección 7.4

Sección 7.5

Sección 7.6



Topic 1

En general, el objetivo del diseño de base de datos relacionales es generar un conjunto de relaciones que nos permita almacenar información sin redundancias innecesarias y, al mismo tiempo nos permita recuperar información fácilmente. Este se consigue diseñando esquemas que tengan una forma normal adecuada.

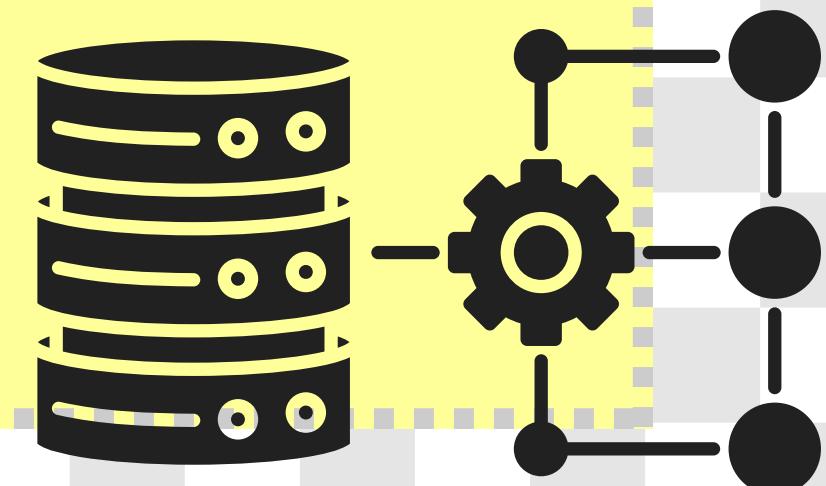
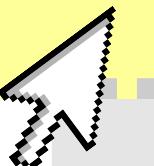
A presentation slide interface with a blue header bar. The main content area contains the explanatory text. On the right side, there is a vertical scroll bar with up and down arrows. At the bottom, there is a navigation bar with left and right arrows, and a large rectangular button labeled "Start" with a cursor icon pointing towards it.



21

Características de los buenos diseños relacionales:

[Back to Agenda Page](#)



7.1. Características de los buenos diseños relacionales:

En el capítulo 6, nos enseñaron el diseño entidad - relación, este; un paso fundamental en la creación de base de datos. Descubrimos que este enfoque nos permite generar esquemas de relaciones directamente a partir del diseño E-R, lo cual es crucial para el desarrollo de una base de datos bien estructurada. Sin embargo, es importante destacar que la calidad de estos esquemas depende en gran medida de la calidad del diseño E-R original.

Es esencial reconocer estos desafíos y consideraciones al diseñar nuestras relaciones, con el fin de evitar **redundancias innecesarias**, garantizar la coherencia y flexibilidad de nuestra base de datos a lo largo del tiempo. Este enfoque meticuloso y reflexivo nos permitirá construir una base de datos sólida y adaptable, capaz de satisfacer las necesidades presentes y futuras de nuestra base de datos.

in_dep (ID, name, salary, dept_name, building, budget)

classroom(building, room_number, capacity)

department(dept_name, building, budget)

course(course_id, title, dept_name, credits)

instructor(ID, name, dept_name, salary)

section(course_id, sec_id, semester, year, building, room_number, time_slot_id)

teaches(ID, course_id, sec_id, semester, year)

student(ID, name, dept_name, tot_cred)

takes(ID, course_id, sec_id, semester, year, grade)

advisor(s_ID, i_ID)

time_slot(time_slot_id, day, start_time, end_time)

prereq(course_id, prereq_id)

ID	name	salary	dept_name	building	budget
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000



7.1.1 Descomposición:

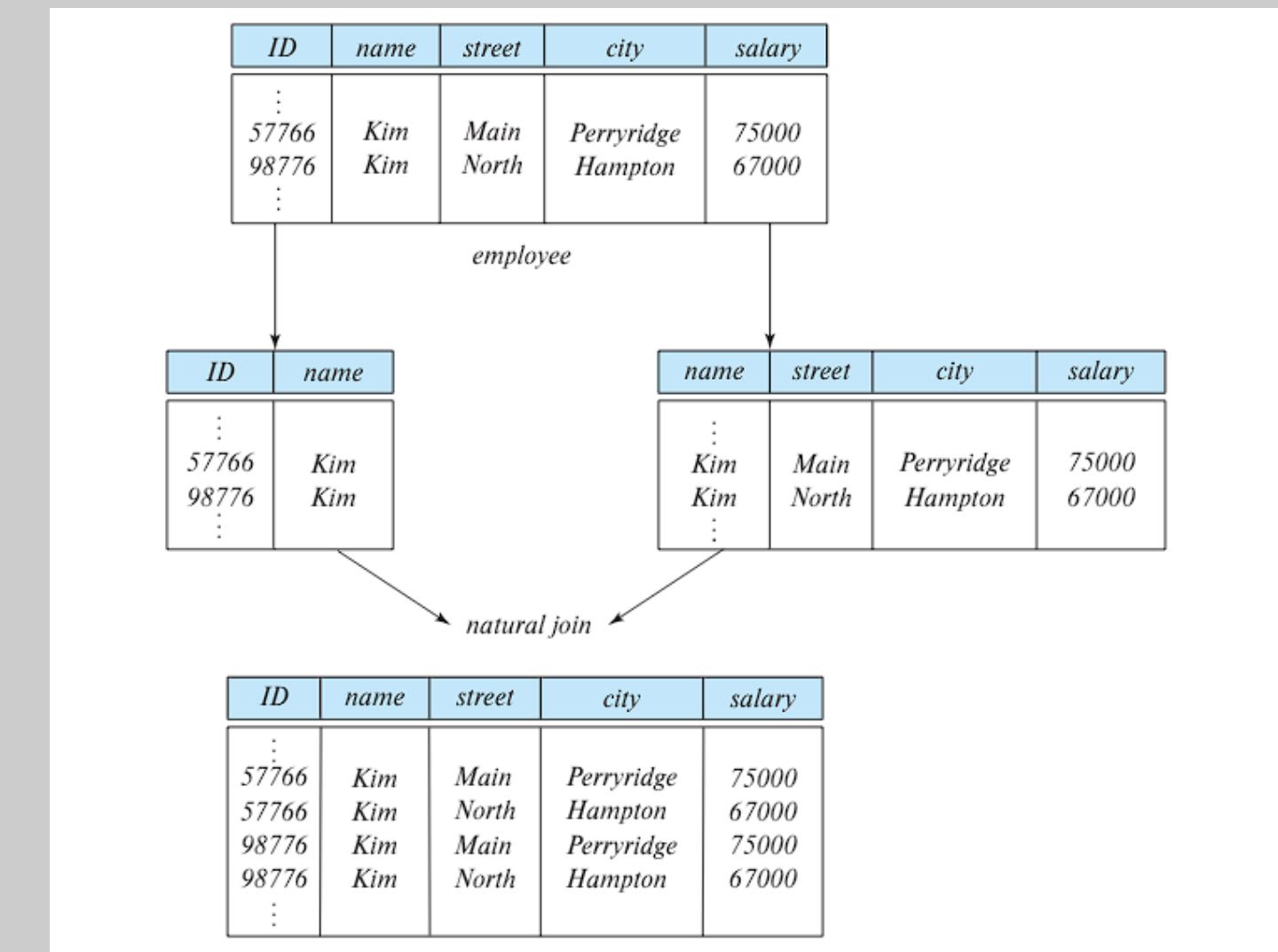
La única manera de evitar el problema de repetición de información en el esquema `in_dep` es descomponerlo en dos esquemas (en este caso, el esquema del instructor y el del departamento). Más adelante en este capítulo presentaremos algoritmos para decidir qué esquemas son apropiados y cuáles no. En general, un esquema que presenta repetición de información puede tener que descomponerse en varios esquemas más pequeños. No todas las descomposiciones de esquemas son útiles. Consideremos un caso extremo en el que todos los esquemas constan de un solo atributo. No podrían expresarse relaciones interesantes de ningún tipo.

employee (ID, name, street, city, salary)

employee1 (ID, name)

employee2 (name, street, city, salary)

(57766, Kim, Main, Perryridge, 75000)
(98776, Kim, North, Hampton, 67000)



Tipo de descomposiciones:

- Descomposiciones con pérdida
- Descomposiciones sin pérdida



7.1.2 Descomposición sin pérdidas:

Sea R un esquema de relación y permita que R1 y R2 formen una descomposición de R, es decir, considerando R, R1 y R2 como conjuntos de atributos, $R = R_1 \cup R_2$. Decimos que la descomposición es sin pérdidas si no hay pérdida de información al reemplazar R con dos esquemas de relación R1 y R2. La pérdida de información ocurre si es posible tener una instancia de una relación r(R) que incluya información que no se puede representar si en lugar de la instancia de r(R) debemos usar instancias de r1(R1) y r2(R2)

Más concretamente, decimos que la descomposición no tiene pérdidas si, para todas las instancias legales de la base de datos, la relación r contiene el mismo conjunto de tuplas que el resultado de la siguiente consulta SQL

```
select *\nfrom (select R1 from r)\n      natural join\n      (select R2 from r)
```

Álgebra relacional

$$\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) = r$$

$$r \subseteq \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$$

Puede parecer contraintuitivo que tengamos más tuplas pero menos información, pero así es. La versión descompuesta es incapaz de representar la ausencia de conexión entre un nombre y una dirección o un salario, y la ausencia de conexión sí es información.

En otras palabras, si proyectamos r sobre R1 y R2, y calculamos la unión natural de los resultados de la proyección, obtenemos exactamente r.

El resultado de employee1 natural join employee2 es un superconjunto de la relación original empleado, pero la descomposición tiene pérdidas, ya que el resultado de la unión ha perdido información sobre qué identificadores de empleado corresponden a qué direcciones y salarios.



7.1.3 Teoría de la normalización:

El método para diseñar una base de datos relacional consiste en utilizar un proceso comúnmente conocido como normalización. El objetivo es generar un conjunto de esquemas de relaciones que nos permita almacenar información sin redundancias innecesarias, pero que también nos permita recuperar información fácilmente. El planteamiento es el siguiente:

- Decidir si un esquema de relación dado está en «buena forma». Existen varias formas formas (llamadas formas normales), que veremos en la sección 7.3.
- Si un esquema de relación dado no está en «buena forma», entonces lo descomponemos en una serie de esquemas de relación más pequeños, cada uno de los cuales está en «buena forma». en varios esquemas de relación más pequeños, cada uno de los cuales tiene una forma normal adecuada. adecuada. La descomposición debe ser sin pérdidas.

Lo que se busca es que no se produzca problemas en dicha base de datos:

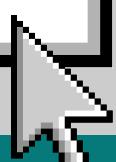
- **Redundancia:** Se llama así a los datos que se repiten continua e innecesariamente por las tablas de las bases de datos.
- **Ambigüedades:** Datos que no clarifican suficientemente el registro al que representan.

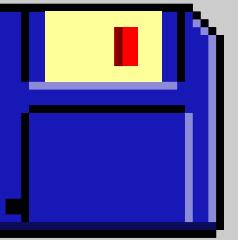
Primera forma:

En la forma normal propia al esquema relacional, de **uso obligatorio**.

- Todos los atributos llave están definidos.
- No hay grupos repetidos en la tabla. Cada fila/columna contiene un solo valor, no un conjunto de ellos.
- Todos los atributos son dependientes de la llave primaria.
- Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son simples e indivisibles.
- No debe existir variación en el número de columnas.
- Los campos no clave deben identificarse por la clave (dependencia funcional).
- Debe existir una independencia del orden tanto de las filas como de las columnas; es decir, si los datos cambian de orden no deben cambiar sus significados.

Start





La mayoría de tablas que existen o que se hacen, cumplen con esta forma, sin embargo hay ciertas tablas las cuales no satisfacen esta forma; he aquí unas de ellas:

- Una tabla que carece de una clave primaria. Esta tabla podría acomodar filas duplicadas, en violación de la condición de filas duplicadas.
- Una vista cuya definición exige que los resultados sean retornados en un orden particular, de modo que el orden de la fila sea un aspecto intrínseco y significativo de la vista. Las tuplas en relaciones verdaderas no están ordenadas una con respecto de la otra.
- Una tabla con por lo menos un atributo que pueda ser nulo. Un atributo que pueda ser nulo estaría en violación de la condición 4, que requiere a cada campo contener exactamente un valor de su dominio de columna. Sin embargo, debe observarse que este aspecto de la condición 4 es controvertido. Muchos autores consideran que una tabla está en 1FN si ninguna clave candidata puede contener valores nulos, pero se aceptan éstos para atributos (campos) que no sean clave, según el modelo original de Codd sobre el modelo relacional, el cual hizo disposición explícita para los nulos.

ALUMNO		
rut	nombre	curso
1-9	Pedro	Algoritmos y Estructuras de datos
2-7	Juan	Bases de Datos Algoritmos y Estructuras de datos
3-5	Diego	
4-4	Maria	Bases de Datos

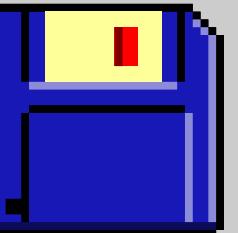
1FN : Ninguna tupla puede tener atributos multi valueds

ALUMNO		
rut	nombre	curso
1-9	Pedro	Algoritmos y Estructuras de datos
2-7	Juan	Bases de Datos
2-7	Juan	Algoritmos y Estructuras de datos Bases de Datos
3-5	Diego	
4-4	Maria	Bases de Datos

Cliente			
ID Cliente	Nombre	Apellido	Teléfono
123	Rachel	Ingram	555-861-2025
456	James	Wright	555-403-1659 555-776-4100
789	Cesar	Dure	555-808-9633

Solución ?

Cliente					
ID Cliente	Nombre	Apellido	Teléfono 1	Teléfono 2	Teléfono 3
123	Rachel	Ingram	555-861-2025		
456	James	Wright	555-403-1659	555-776-4100	
789	Cesar	Dure	555-808-9633		



Segunda Forma

Cliente			Teléfono del cliente	
ID Cliente	Nombre	Apellido	ID Cliente	Teléfono
123	Rachel	Ingram	123	555-861-2025
456	James	Wright	456	555-403-1659
789	Cesar	Dure	456	555-776-4100
			789	555-808-9633

ALUMNOS MATRICULADOS

rut	nombre	apellido	cod_curso	descripcion
1-9	Pedro	Pérez	AE600	Algoritmos y Estructuras de datos
2-7	Juan	Jara	BD253	Bases de Datos
2-7	Juan	Jara	AE600	Algoritmos y Estructuras de datos
3-5	Diego	Díaz	BD253	Bases de Datos
4-4	Maria	Martinez	BD253	Bases de Datos

ALUMNO		
rut	nombre	apellido
1-9	Pedro	Pérez
2-7	Juan	Jara
3-5	Diego	Díaz
4-4	Maria	Martinez

CURSO	
cod_curso	descripcion
AE600	Algoritmos y Estructuras de datos
BD253	Bases de Datos

MATRICULA	
rut	cod_curso
1-9	AE600
2-7	BD253
2-7	AE600
3-5	BD253
4-4	BD253

Entonces, la solución pasa por la creación de otra entidad/tabla para el almacenamiento de teléfonos.

Problema de Redundancia

matricula	nombre	dirección	telefono	curso	código	carrera
100	juan	ongolmo 340, concepción	78872890	base de datos	bd1	ingeniería civil informatica
100	juan	ongolmo 340, concepción	78872890	estadística	e2	ingeniería civil informatica
100	juan	ongolmo 340, concepción	78872890	analitica	a5	ingeniería civil informatica
200	ana	san martín 840, santiago	78342367	estadística	e2	ingeniería comercial
200	ana	san martín 840, santiago	78342367	analitica	a5	ingeniería comercial

Primera Forma en Redundancia: Separamos las tablas estableciendo la relación entre ella por matricula, que referencia al alumno (clave primaria). Por tanto la nueva tabla tiene como clave primaria “codigo” y posee un campo matricula que actúa como clave foránea a matricula en alumno.

La relación alumno.matricula <- curso.matricula establece la vinculación entre los registros de cada tabla.

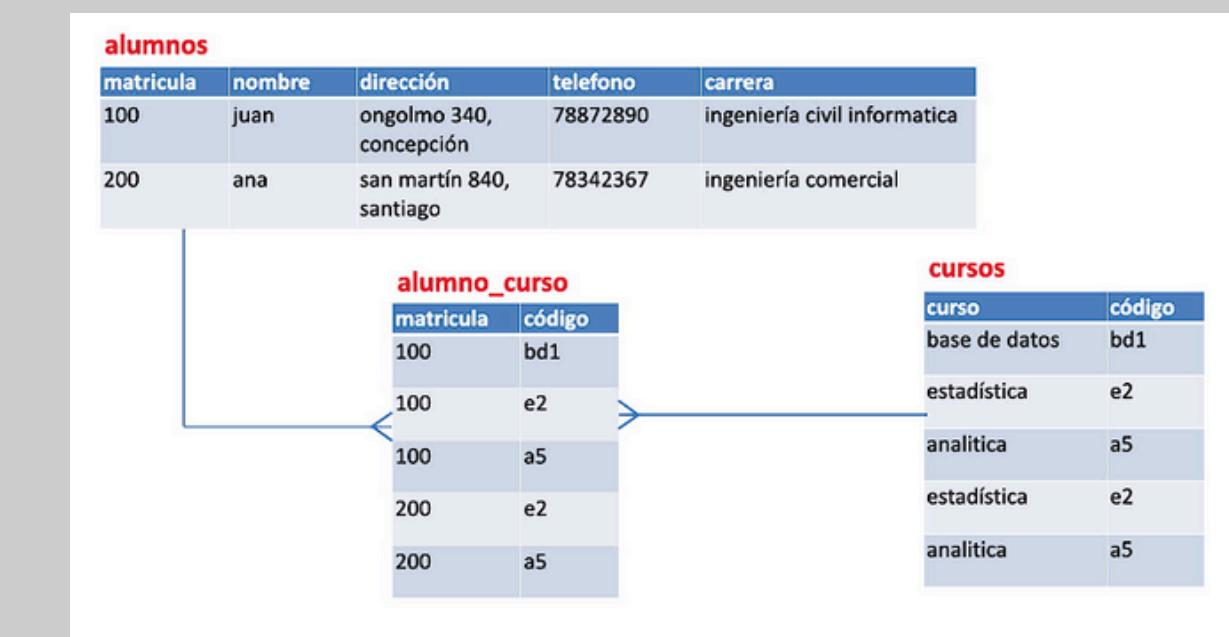
matricula	nombre	dirección	telefono	carrera
100	juan	ongolmo 340, concepción	78872890	ingeniería civil informatica
200	ana	san martín 840, santiago	78342367	ingeniería comercial

matricula	curso	código		
100	base de datos	bd1		
100	estadística	e2		
100	analítica	a5		
200	estadística	e2		
200	analítica	a5		

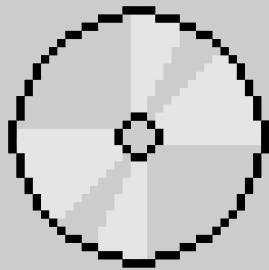
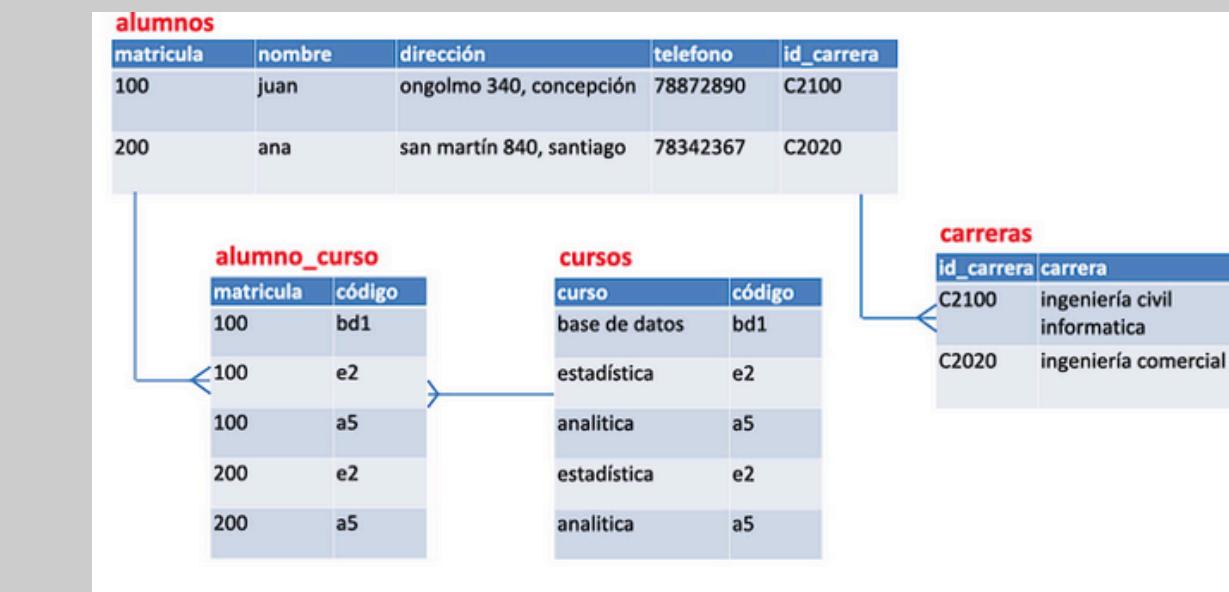
matricula	nombre	dirección	telefono	carrera
100	juan	ongolmo 340, concepción	78872890	ingeniería civil informatica
200	ana	san martín 840, santiago	78342367	ingeniería comercial

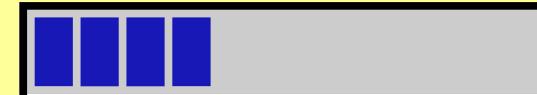
matricula	curso	código		
100	base de datos	bd1		
100	estadística	e2		
100	analítica	a5		
200	estadística	e2		
200	analítica	a5		

Segunda Forma en Redundancia. Cuando se estable una relación muchos a muchos (N:M) se genera una relación como tabla adicional. En este caso alumno_curso.



Tercera Forma en Redundancia. Eliminar aquellos campos que no dependan de la clave. Carrera no depende directamente de la clave primaria en alumnos, por tanto debe sacarse de la tabla alumnos.

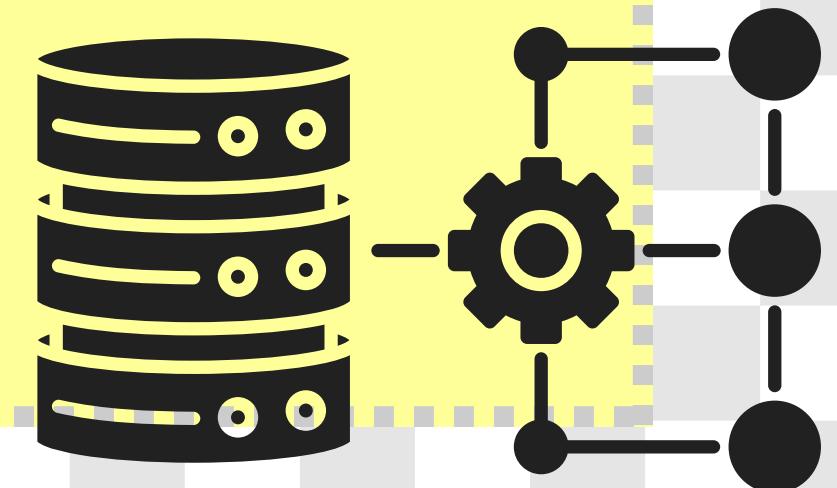
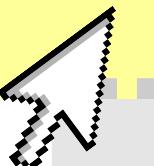




72

Descomposición mediante dependencias funcionales

[Back to Agenda Page](#)



7.2. Descomposición mediante dependencias funcionales

Una base de datos modela un conjunto de entidades y relaciones en el mundo real. Generalmente hay una variedad de restricciones (reglas) sobre los datos en el mundo real. Por ejemplo, algunas de las restricciones que se espera que se cumplan en una base de datos universitaria son:

- Los estudiantes e instructores se identifican de forma única por su identificación.
- Cada estudiante e instructor tiene un solo nombre.
- Cada instructor y estudiante está (principalmente) asociado con un solo departamento.²
- Cada departamento tiene un solo valor para su presupuesto y un solo edificio asociado.

Un caso de una relación que satisface todas esas restricciones del mundo real se denomina caso legal de la relación; una instancia legal de una base de datos es aquella en la que todas las instancias de relación son instancias legales.

Un esquema de relación es un conjunto de atributos, pero no todos los conjuntos de atributos son esquemas.

Cuando utilizamos una letra griega minúscula, nos referimos a un conjunto de atributos que puede o no ser un esquema. Se utiliza una letra romana cuando queremos indicar que el conjunto de atributos es definitivamente un esquema.



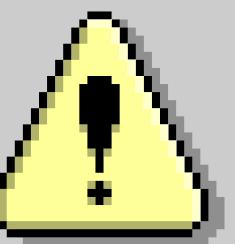
7.2.1 Convenciones de anotación:

Al hablar de algoritmos para el diseño de bases de datos relacionales, tendremos que hablar de relaciones arbitrarias y su esquema, en lugar de hablar sólo de ejemplos. Recordando nuestra introducción al modelo relacional en el capítulo 2, resumimos aquí nuestra notación.

- utilizamos letras griegas para los conjuntos de atributos
- Utilizamos una letra mayúscula romana para referirnos a un esquema de relación
- Utilizamos la notación $r(R)$ para mostrar que el esquema R corresponde a la relación r .
- Cuando un conjunto de atributos es una superclave, podemos denotarlo por K . Una superclave pertenece a un esquema de relación específico, por lo que utilizamos la terminología « K es una superclave para R ».
- Utilizamos un nombre en minúsculas para las relaciones. En nuestros ejemplos, estos nombres pretenden que sean realistas (por ejemplo, instructor), mientras que en nuestras definiciones y algoritmos, utilizamos letras simples, como r .
- Así pues, la notación $r(R)$ se refiere a la relación r con el esquema R . Cuando escribimos $r(R)$, nos referimos tanto a la relación como a su esquema.
- Una relación tiene un valor determinado en un momento dado y utilizamos el término «instancia de r ». Cuando está claro que se trata de una instancia, podemos utilizar simplemente el nombre de la relación (por ejemplo, r).

Para simplificar, supondremos que los nombres de los atributos sólo tienen un significado dentro del esquema de la base de datos.

esquema de base de datos.



7.2.2 Claves y dependencias funcionales:

Algunos de los tipos de restricciones más utilizados en el mundo real pueden representarse formalmente como claves (superclaves, claves candidatas y claves primarias) o como dependencias funcionales, que definimos a continuación:

Volvemos a formular esta definición del siguiente modo:

Dado $r(R)$, un subconjunto $K \rightarrow R$ es una superclave de $r(R)$ si, en cualquier instancia legal de $r(R)$, para todos los pares t_1 y t_2 de tuplas en la instancia de r si $t_1 \neq t_2$, entonces $t_1[K] \neq t_2[K]$. Es decir, no hay dos tuplas en cualquier instancia legal de la relación $r(R)$ que puedan tener el mismo valor en el conjunto de atributos K . Si no hay dos tuplas en r que tengan el mismo valor en K , entonces un valor K identifica de forma única una tupla en r .

Mientras que una superclave es un conjunto de atributos que identifica de forma única una tupla entera, una nos permite expresar restricciones que identifican únicamente los valores de ciertos atributos.

Consideremos un esquema de relación $r(R)$, y dejemos que $\alpha \subseteq R$ y $\beta \subseteq R$.

- Dada una instancia de $r(R)$, decimos que la instancia satisface la dependencia funcional $\alpha \rightarrow \beta$ si para todos los pares de tuplas t_1 y t_2 de la instancia tales que $t_1[\alpha] = t_2[\alpha]$, se da también el caso de que $t_1[\beta] = t_2[\beta]$.
- Decimos que la dependencia funcional $\alpha \rightarrow \beta$ se cumple en el esquema $r(R)$ si, cada instancia legal instancia de $r(R)$ satisface la dependencia funcional.

ID, dept_name → name, salary, building, budget

Utilizando la notación de dependencia funcional, decimos que K es una superclave para $r(R)$ si la dependencia funcional $K \rightarrow R$ se mantiene en $r(R)$. En otras palabras, K es una superclave si para cada instancia legal de $r(R)$, para cada par de tuplas t_1 y t_2 de la instancia, siempre que $t_1[K] = t_2[K]$, también se da el caso de que $t_1[R] = t_2[R]$ (es decir, $t_1 = t_2$). Las dependencias funcionales nos permiten expresar restricciones que no podemos expresar con superclaves.

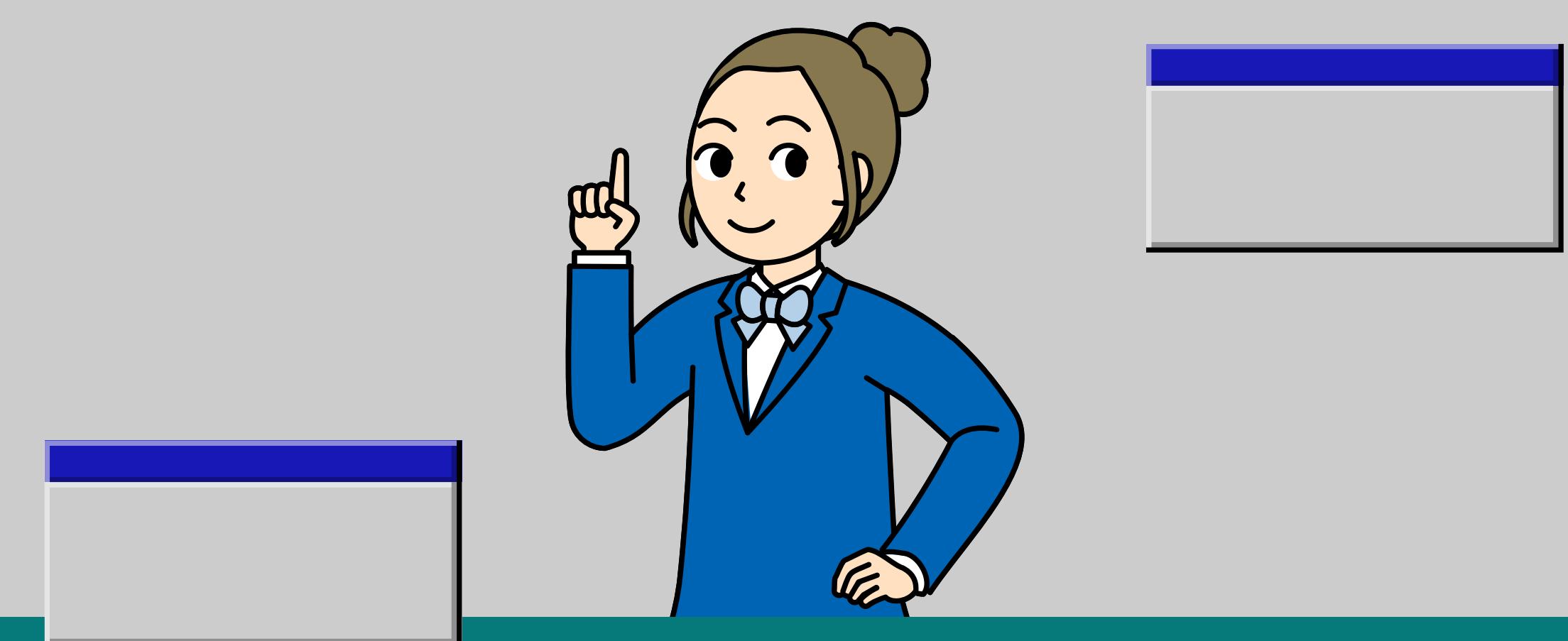
Utilizaremos las dependencias funcionales de dos maneras:

- Para probar instancias de relaciones para ver si satisfacen un conjunto dado F de dependencias funcionales.
- Para especificar restricciones sobre el conjunto de relaciones legales. Así pues, sólo nos ocuparemos de las instancias de relaciones que satisfacen un conjunto dado de dependencias funcionales. Si deseamos limitarnos a las relaciones del esquema $r(R)$ que satisfacen un conjunto F de dependencias funcionales, diremos que F se cumple en $r(R)$

<i>building</i>	<i>room_number</i>	<i>capacity</i>
Packard	101	500
Painter	514	10
Taylor	3128	70
Watson	100	30
Watson	120	50

Relación en el aula.

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_3	c_2	d_3
a_3	b_3	c_2	d_4



7.2.3 Descomposición sin pérdidas y dependencias funcionales:

Podemos usar dependencias funcionales para mostrar cuándo ciertas descomposiciones no tienen pérdidas. Sean R, R1, R2 y F como se indica arriba. R1 y R2 forman una descomposición sin pérdidas de R si en al menos una de las siguientes dependencias funcionales está en F+:

- $R_1 \cap R_2 \rightarrow R_1$
- $R_1 \cap R_2 \rightarrow R_2$

En otras palabras, si $R_1 \cap R_2$ forma una superclave para R1 o R2, la descomposición de R es una descomposición sin pérdidas. Podemos usar el cierre de atributos para probar eficientemente las superclaves, como hemos visto anteriormente.

Supongamos que descomponemos un esquema de relación r(R) en r1(R1) y r2(R2), donde $R_1 \cap R_2 \rightarrow R_1$. Entonces se deben imponer las siguientes restricciones SQL al descompuesto esquema para garantizar que su contenido sea coherente con el esquema original.

- $R_1 \cap R_2$ es la clave principal de r1. Esta restricción impone la dependencia funcional.
- $R_1 \cap R_2$ es una clave externa de r2 que hace referencia a r1. Esta restricción asegura que cada tupla en r2 tenga una tupla coincidente en r1, sin que no aparecería en la unión natural de r1 y r2.

Si r1 o r2 se descomponen aún más, siempre que la descomposición garantice que todos los atributos en $R_1 \cap R_2$ están en una relación, la restricción de clave primaria o externa en r1 o r2 sería heredado por esa relación.

in_dep (ID, name, salary, dept_name, building, budget)

Solución



*instructor (ID, name, dept_name, salary)
department (dept_name, building, budget)*

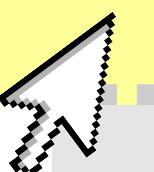


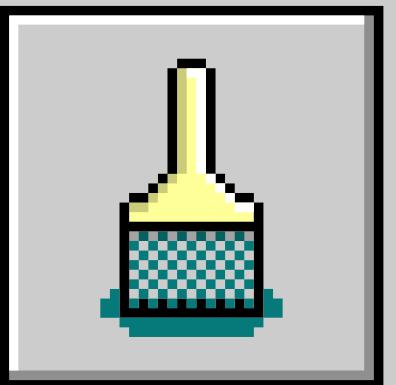
7.3

Formas Normales



[Back to Agenda Page](#)



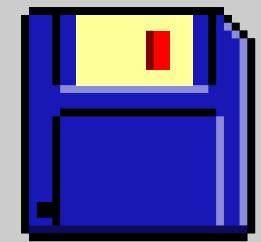


Forma Normal

Normalización de datos, mientras mas alta sea la forma normal, existiran menos inconsistencias y anomalias



7.3.1 Forma Normal de Boyce - Codd (BCNF)



¿Como funciona?

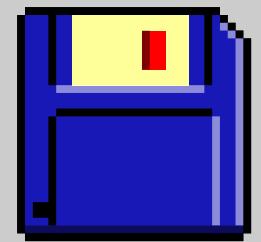
Nos ayuda a eliminar la redundancia en las bases de datos, asegurando que las dependencias funcionales se manejen de forma adecuada. Simplificando, cada atributo no clave, debe dar un hecho único sobre una clave, solo esta.

$\alpha \rightarrow \beta$ es trivial ($\beta \subseteq \alpha$)

X → Trivial

Todo lo de β esta en α

Dependencia funcional, no dice nada, atributo dará sus valores
ID_Estudiante Nombre Nombre ID_Estudiante Nombre y Apellido
 Apellido Nombre Nombre
 Nombre Nombre Nombre



7.3.1 Forma Normal de Boyce - Codd (BCNF)

Ejemplo

$\alpha \rightarrow \beta$ is a trivial functional dependency (i.e., $\beta \subseteq \alpha$).

$\alpha \rightarrow \beta$ se considera trivial si todos los atributos en β están incluidos

Ingredientes para hacer receta

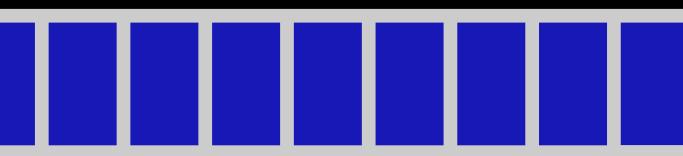
Yo quiero hacer un pastel :)

- Harina, huevos, mas harina

“Redundando”, harina :



Atributo que ya lo incluye, dando la obvio (trivial)

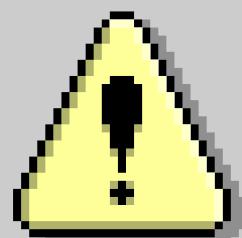


7.3.1 Forma Normal de Boyce - Codd (BCNF)

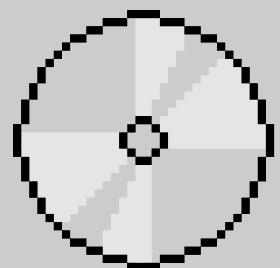
Ejemplo

α is a superkey for schema R .

α es una superclave para el esquema R.



Superclave (Conjunto de uno o mas atributos) mezclar de forma única
No tiene que ser la PK



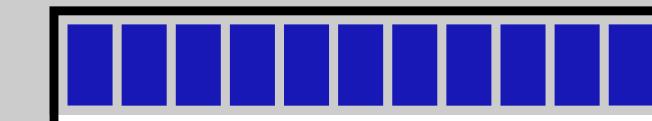
Podremos tener atributos que combinados, pueden ser identificados de forma
única cada fila en una tabla

7.3.1 Forma Normal de Boyce - Codd (BCNF)

Ejemplo



221-10-521805



Código	Fecha Ingreso	Etapa	Codigo Carrera	Combinación Nombre Apellidos	Repetición si Existiera	Año Nacimiento
221-10-631805	22	1	10	5218	0	5
221-10-631815	22	1	10	5218	1	5

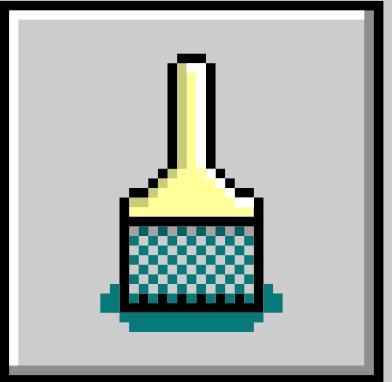
05 02 18

Freddy Cardenas de la Roca

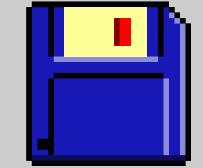
Freddy Cardenas
de la Roca

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

7.3.1 Forma Normal de Boyce - Codd (BCNF)



Profesores(ID_Profesor, Nombre, Departamento y
Presupuesto_Departamento)



Departamento \rightarrow Presupuesto Departamento (Cada Dep, tiene un
presupuesto)

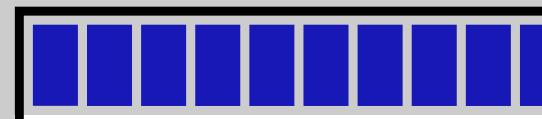


Departamento no es la unica clave



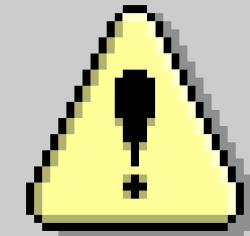
Departamento $\rightarrow\rightarrow$ Mas Profesores

Profesores(ID_Profesor, Nombre, Departamento y
Presupuesto_Departamento)

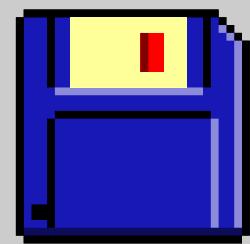


No esta en un BCNF

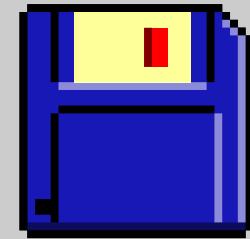
7.3.1 Forma Normal de Boyce - Codd (BCNF)



Profesores (ID_Profesor, Nombre, Departamento y Presupuesto_Departamento)



Departamentos (Departamento, Presupuesto_Departamento)



Profesores (ID_Profesor, Nombre, Departamento)



7.3.2 Tercera Forma Normal (3NF)

¿Como funciona?

En la 3NF, se busca eliminar las dependencias transitivas para evitar redundancias. No debe existir atributos en una tabla que dependa de otros atributos que no sean la PK.



- $\alpha \rightarrow \beta$ is a trivial functional dependency.
- α is a superkey for R .
- Each attribute A in $\beta - \alpha$ is contained in a candidate key for R .



Cada atributo A en $\beta - \alpha$ está contenido en una clave candidata para R?

7.3.2 Tercera Forma Normal (3NF)

¿Como funciona?

Cada atributo A en $\beta - \alpha$ está contenido en una clave candidata para R

Ejemplo

3NF



ID_Emppleado (clave primaria)

Nombre

Apellido

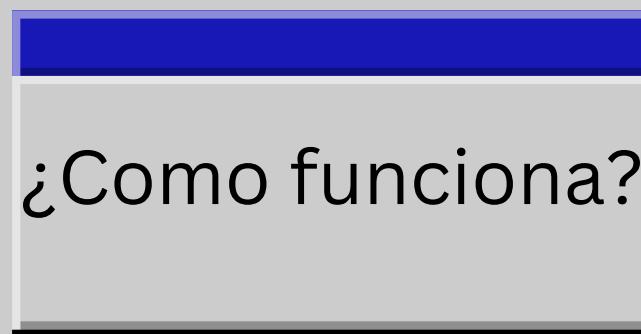
ID_Departamento

Nombre_Departamento

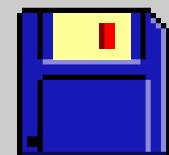
ID_Emppleado, Nombre, Apellido, y
ID_Departamento (FK)

ID_Departamento Nombre_Departamento

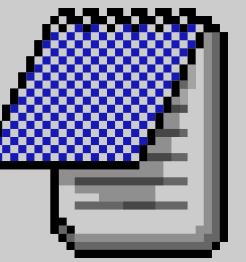
7.3.2 Tercera Forma Normal (3NF)



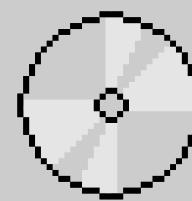
Empleados con ID_Emppleado, Nombre, y ID_Departamento.



Departamentos con ID_Departamento, Nombre_Departamento, y
Ubicación_Departamento.



7.3.3 Comparaciones entre BCNF y 3NF

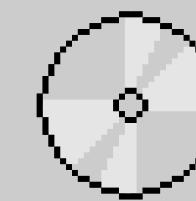
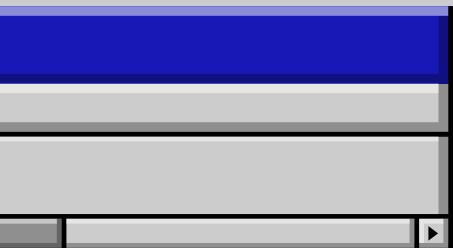


BCNF (Forma Normal de Boyce-Codd)

Es más estricta que 3NF.

Requiere que para cada dependencia funcional no trivial

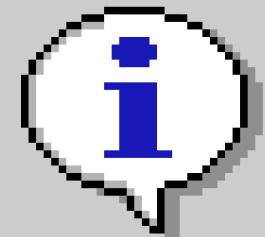
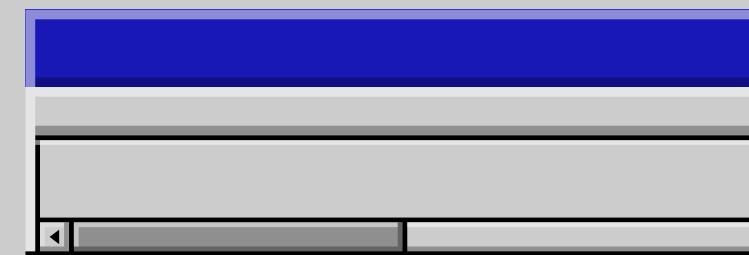
α debe ser una superclave, lo que significa que α debe poder identificar de forma única cada fila en una tabla.



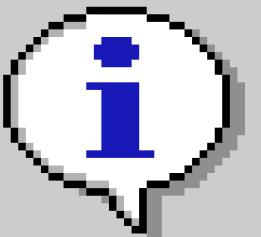
3NF (Tercera Forma Normal):

Es menos estricta que BCNF.

Permite dependencias funcionales no triviales donde α no es una superclave si cada atributo en β que no es parte de α es parte de una clave candidata.



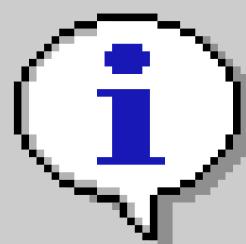
Registro (ID_Estudiante, Nombre, Apellido, ID_Departamento, Nombre_Departamento, Presupuesto_Departamento)



7.3.3 Comparaciones entre BCNF y 3NF



Registro (ID_Estudiante, Nombre, Apellido, ID_Departamento, Nombre_Departamento, Presupuesto_Departamento)



BCNF (Forma Normal de Boyce-Codd)

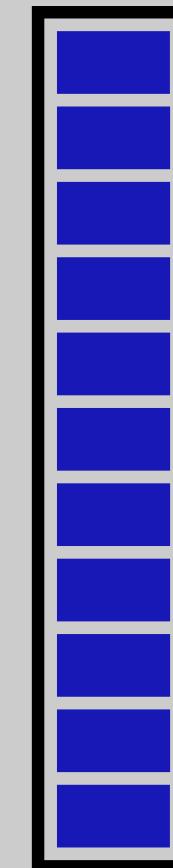
Problema en BCNF:

Nombre_Departamento y Presupuesto_Departamento dependen de ID_Departamento, pero ID_Departamento no es una superclave de la tabla completa.



Departamento (ID_Departamento, Nombre_Departamento, Presupuesto_Departamento)

Superclave ID_Departamento



3NF (Tercera Forma Normal):

Problema en 3NF:

Presupuesto_Departamento es una dependencia transitiva a través de ID_Departamento y no es parte de ninguna clave candidata en la tabla Registro.



Registro (**ID_Estudiante**, Nombre, Apellido, ID_Departamento(FK))

Departamentos (**ID_Departamento**, Nombre_Departamento, Presupuesto_Departamento)



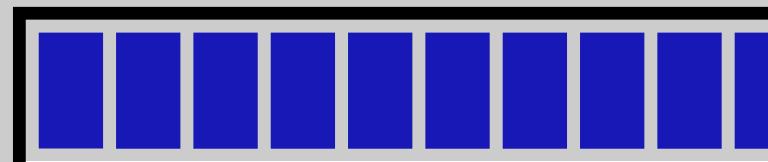
7.3.4 Formas Normales Superiores



¿Como funciona?



Las Formas Normales Superiores, como la Cuarta Forma Normal (4NF) y la Quinta Forma Normal (5NF), se ocupan de problemas más complejos como las dependencias multivaluadas, las dependencias de unión y otros tipos de redundancia que BCNF y 3NF no resuelven.



7.3.4 Formas Normales Superiores

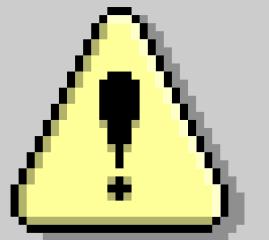


(ID, child name)

(ID, phone number)

Multivaluados

Instructor con 2 Hijos, y con 2 numeros distintos



(ID, child name, phone number)

(99999, David, 512-555-1234)

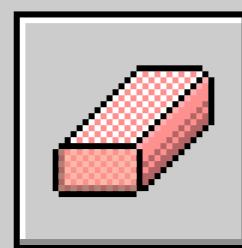
(99999, David, 512-555-4321)

(99999, William, 512-555-1234)

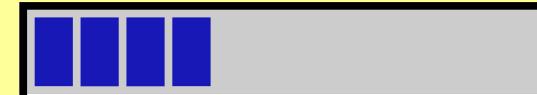
(99999, William, 512-555-4321)



Si no repitiéramos los números de teléfono y guardáramos solo la primera y la última tupla, registraríamos los nombres de los hijos y sus teléfonos, pero diríamos que David tiene el número 512-555-1234 y William el 512-555-4321, y eso es incorrecto

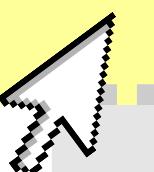


Deberíamos tener tablas para Hijos y N° de Telefono, asociadas por el ID de Instructor, pero con las formas normales no es suficiente



7.4 Teoría de la dependencia funcional

[Back to Agenda Page](#)



7.4.1 Clausura de un conjunto de dependencias funcionales



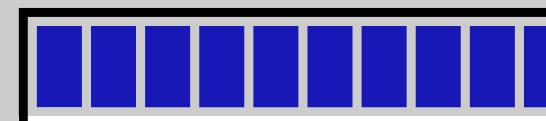
Terminologías

F

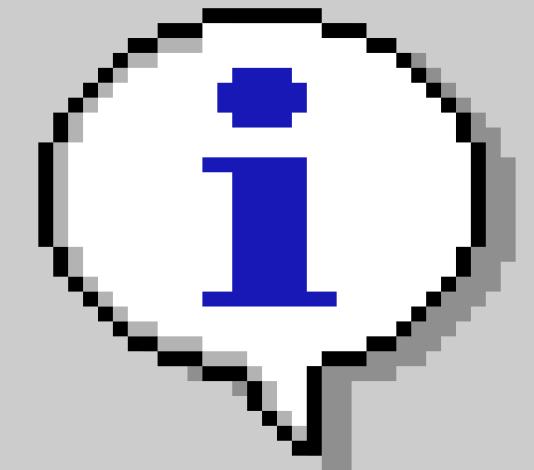
conjunto de DF's especificadas en un esquema de relación R.

F^+

F + DF's lógicamente inferidas



Clausura



**¿Existen dependencias
funcionales lógicamente
inferidas?**

Axiomas o Reglas de Inferencia



$F \models X \rightarrow Y$

Indica que la dependencia funcional $X \rightarrow Y$ se infiere del conjunto de dependencias funcionales F.

A
r
m
st
ro
n
g

- (RI1) **reflexiva:** Si $X \supseteq Y$, entonces $X \rightarrow Y$
- (RI2) **de aumento:** $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
- (RI3) **transitiva:** $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$
- (RI4) **descomposición:** $\{X \rightarrow YZ\} \models X \rightarrow Y$
- (RI5) **unión:** $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$
- (RI6) **pseudotransitiva:** $\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$

} Se pueden derivar a partir de estas tres.

Siendo W, X, Y, Z conjuntos de atributos.

Dado un
esquema de
relación:

$$r(A, B, C, G, H, I)$$

Ejemplo

$$\begin{array}{l} A \rightarrow B \\ A \rightarrow C \\ CG \rightarrow H \\ CG \rightarrow I \\ B \rightarrow H \end{array}$$

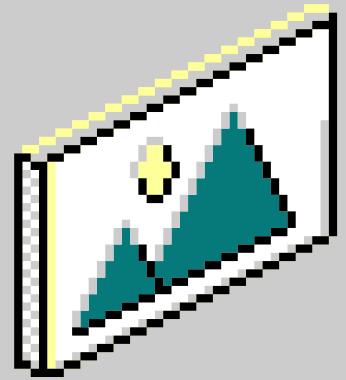
$$A \rightarrow H \quad \left. \begin{array}{l} \{A \rightarrow B, \\ B \rightarrow H\} \end{array} \right\}$$

Regla Transitiva

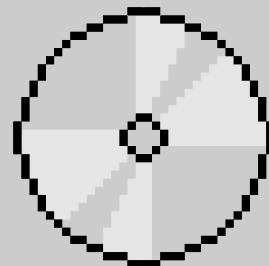
$$AG \rightarrow I \quad \left. \begin{array}{l} A \rightarrow C, \\ AG \rightarrow I \end{array} \right\}$$

Regla
Pseudotransitiva

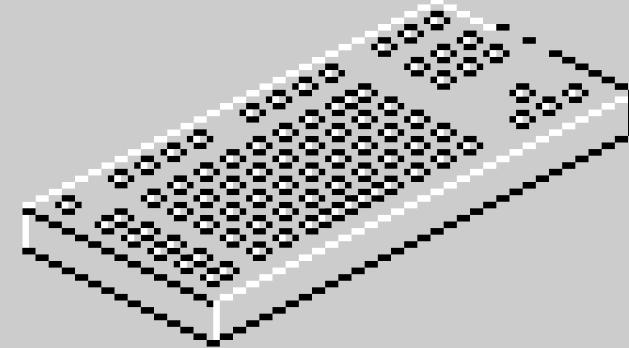
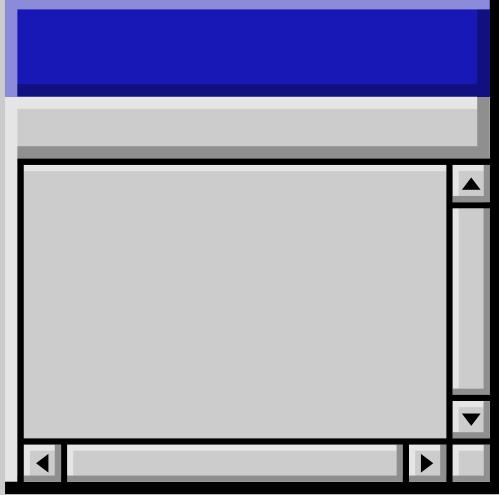
Algoritmo de clausura de Atributos



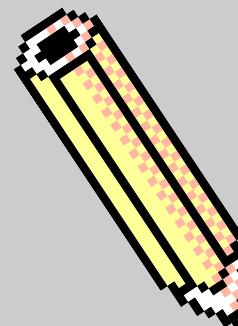
Procedimiento
para calcular
 F^+ de F



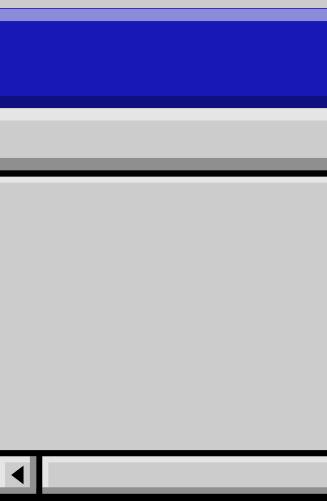
F^+ de F
repeat
 for each DF en $F^+(f)$
 aplicar la regla de aumento y reflexiva en F
 añadir las DF resultantes a F^+
 for each par de DF f_1 y f_2 en F^+
 If f_1 y f_2 se pueden combinar por transitividad
 añadir las DF resultantes a F^+
until F^+ no cambia por ninguna otra F

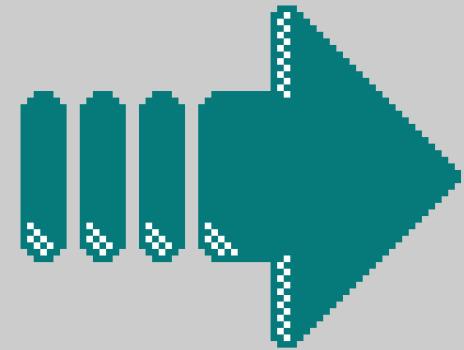


7.4.2 Clausura de conjuntos de atributos

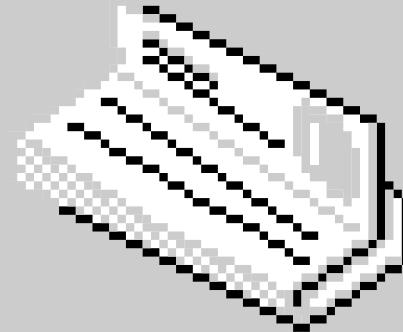


Un atributo B está **funcionalmente determinado** por a si $a \rightarrow B$. Para probar si un conjunto de atributos a es una superclave debemos crear un algoritmo eficiente para calcular el conjunto de atributos determinado funcionalmente por a .





Terminologías

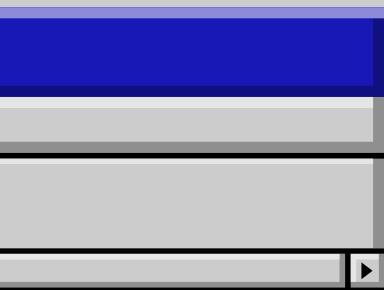


a

conjunto de atributos que aparece en la parte izquierda de alguna dependencia funcional de F.

a⁺

Conjunto de atributos **determinados funcionalmente** por a ,
basados en F .

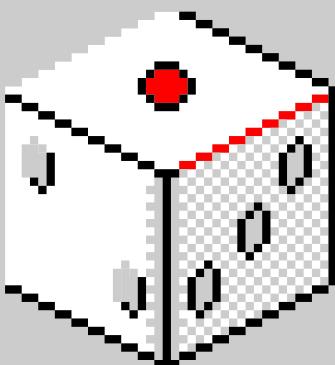


Algoritmo

β, γ : Conjunto de atributos

```
result :=  $\alpha$ 
repeat
    for each  $\beta \rightarrow \gamma$  in  $F$  do
        begin
            if
                 $\beta \subseteq result$ 
            then
                result := result  $\cup \gamma$ 
        end
    until result no cambie
```

Usos



Demostrar si α es una superclave.

Calculamos α^+ y verificamos si α^+ contiene todos los atributos en R

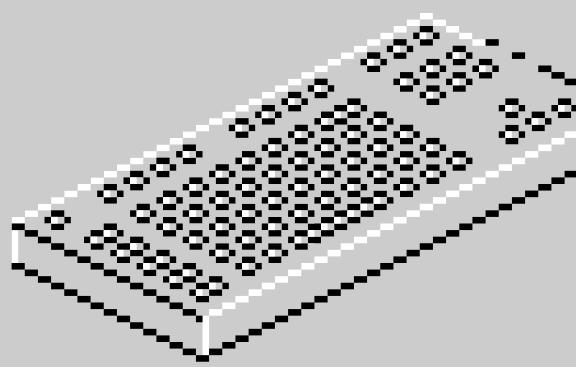
Podemos comprobar si una dependencia funcional $\alpha \rightarrow \beta$ se cumple.

(1) $\beta \subseteq \alpha$

Nos da una forma alternativa de calcular F^+ .

for $F^+ \gamma \subseteq R, S \subseteq \gamma^+$
 then $\gamma \rightarrow S$

Ejemplo



$$(AG)^+ \rightleftharpoons AG$$

result := AG

A → B en F

result := *ABG*

$$A \rightarrow C$$

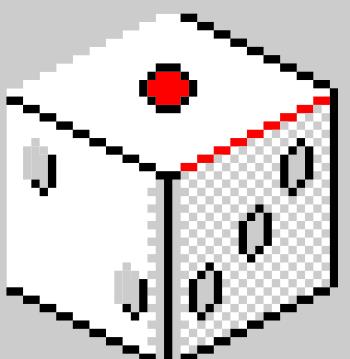
result := *ABCG*

$$CG \rightarrow H$$

result := ABCGH

$$CG \rightarrow I$$

result := ABCGHI



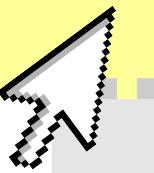
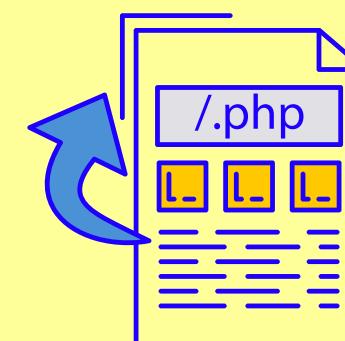
$$\begin{array}{l} A \rightarrow B \\ A \rightarrow C \\ CG \rightarrow H \\ CG \rightarrow I \\ B \rightarrow H \end{array}$$

[Back to Agenda Page](#)



7.4.3

[Back to Agenda Page](#)





7.4 Teoria de la dependencia funcional (Parte 2)

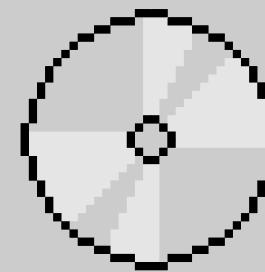
• Atributos Extraños

Imagina una tabla en una base de datos donde los atributos representan características de un objeto. Ahora, si uno de esos atributos puede ser deducido o calculado usando otros atributos de esa misma fila, se considera "extraño". Es como si esa información estuviera redundante o innecesaria.

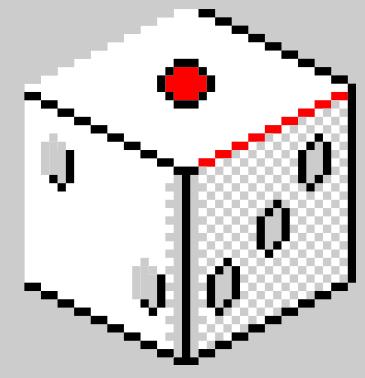
ID_empleado	Nombre	Fecha de nacimiento	Edad
1	Juan	1990-05-15	32
2	María	1985-10-20	36
3	Pedro	1988-03-10	33

Por ejemplo, si tienes un atributo "**Edad**" y otro "**Fecha de nacimiento**", la edad se puede calcular a partir de la fecha de nacimiento. Por lo tanto, "Edad" sería un atributo extraño, ya que se puede deducir de otro atributo presente en la misma fila (la fecha de nacimiento).





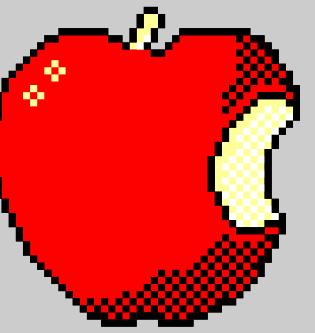
Definicion Formal:



Considere un conjunto F de dependencias funcionales y la dependencia funcional $\alpha \rightarrow \beta$ en F .

- **Eliminación del lado izquierdo:** El atributo A es extraño en α si $A \in \alpha$ y F lógicamente implica $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$
- **Eliminación del lado derecho:** El atributo A es extraño en β si $A \in \beta$ y el conjunto de dependencias funcionales $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ implica lógicamente F





Así podemos probar si un atributo es extraño:

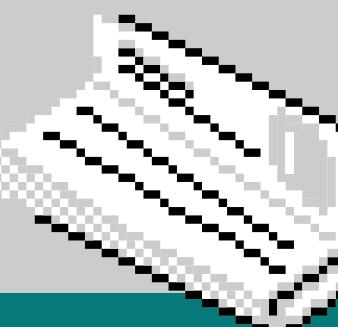
Sea R un esquema de relación y F el conjunto de datos de dependencias funcionales que se mantiene en R . Considere un atributo A es una dependencia de $\alpha \rightarrow \beta$.

- Si $A \in \beta$, para comprobar si A es extraño, consideramos el conjunto:

$$F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$$

Y comprobamos si $\alpha \rightarrow A$ se puede inferir en F' . Para hacerlo calculamos α^+ bajo F ; si α^+ incluye a A , entonces A es extraño en β .

- Si $A \in \alpha$, para verificar si A es extraño, sea $y = \alpha - \{A\}$, y verifique si $y \rightarrow \beta$ se puede inferir de F . Para hacerlo calcule y^+ bajo F ; si y^+ incluye todos los atributos en β , entonces A es extraño en α .





EJEMPLO: Supongamos que tenemos la siguiente dependencia funcional:

$$\begin{aligned}\alpha = \{A, B\} &\rightarrow \beta = \{C, D\} \\ \alpha &\rightarrow \beta\end{aligned}$$

Determinar si el atributo A es "extraño"

1. **Si $A \in \beta$:** Dado que A no está en β , no necesitamos realizar este paso.

2. **Si $A \in \alpha$:**

- Ahora vamos a calcular si A es "extraño" en α .

- **Paso 1:** Definimos $y = \alpha - \{A\} = \{B\}$.

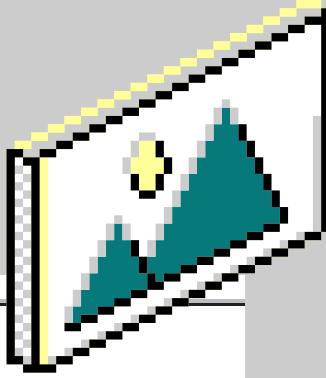
- **Paso 2:** Calculamos y^+ bajo F. Supongamos que tenemos las siguientes dependencias funcionales adicionales: $\{B \rightarrow C, C \rightarrow D\}$.

- Entonces, y^+ sería $\{B, C, D\}$.

- **Paso 3:** Verificamos si y^+ incluye todos los atributos en β , es decir, C y D. En este caso, y^+ sí incluye todos los atributos en β .

- Como y^+ incluye todos los atributos en β , concluimos que A es "extraño" en α .

∴ A es "extraño" en la dependencia funcional $\{A, B\} \rightarrow \{C, D\}$.



- **Cobertura canónica** (Todos los datos son capturados y almacenados correctamente en la BD)

Una cobertura canónica de FC para F es un conjunto de dependencias tales que F implica lógicamente todas las dependencias en FC, y FC implica todas las dependencias en F:

PROPIEDADES:

- Ninguna dependencia funcional en FC contiene atributos extraños
- Cada lado izquierdo de una dependencia funcional en FC es único. Es decir, no hay dos dependencias $\alpha_1 \rightarrow \beta_1$ y $\alpha_2 \rightarrow \beta_2$ en F_c tales que $\alpha_1 = \alpha_2$.

$F_c = F$

repetir

Utilice la regla de unión para reemplazar cualquier dependencia en F_c de la forma $\alpha_1 \rightarrow \beta_1$ y $\alpha_1 \rightarrow \beta_2$ con $\alpha_1 \rightarrow \beta_1 \beta_2$.

Encuentre una dependencia funcional $\alpha \rightarrow \beta$ en F_c con un atributo extraño ya sea en α o en β .

/* Nota: la prueba de atributos extraños se realiza utilizando F_c , no F */

Si se encuentra un atributo extraño, elimínelo de $\alpha \rightarrow \beta$ en F_c hasta (F_c no cambia)

Figura 7.9 Cobertura canónica informática.

EJEMPLO:

ID_empleado	Nombre	Fecha de nacimiento	Edad
1	Juan	1990-05-15	32
2	María	1985-10-20	36
3	Pedro	1988-03-10	33

Dependencias Funcionales:

- $ID_empleado \rightarrow Nombre$
- $ID_empleado \rightarrow Fecha\ de\ nacimiento$
- $Fecha\ de\ nacimiento \rightarrow Edad$

Cobertura Canónica sin Atributos Extraños:

- $ID_empleado \rightarrow Nombre$
- $ID_empleado \rightarrow Fecha\ de\ nacimiento$

EJEMPLO:

Consideremos ahora un ejemplo. Supongamos que tenemos el siguiente conjunto F de dependencias funcionales del esquema (A, B, C):

$$A \rightarrow BC$$

$$B \rightarrow C$$

$$A \rightarrow B$$

$$AB \rightarrow C$$

Calculemos una cobertura canónica para F.

- Hay dos dependencias funcionales con el mismo conjunto de atributos en el lado izquierdo de la flecha

$$A \rightarrow BC$$

$$A \rightarrow B$$

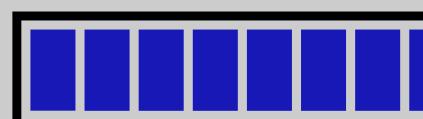
Combinamos estas dependencias funcionales en $A \rightarrow BC$.

- A es extraño en $AB \rightarrow C$ porque F implica lógicamente $(F - \{AB \rightarrow C\}) \cup \{B \rightarrow C\}$. Esta afirmación es cierta porque $B \rightarrow C$ ya está en nuestro conjunto de dependencias funcionales.
- C es extraño en $A \rightarrow BC$, ya que $A \rightarrow BC$ está lógicamente implicado por $A \rightarrow B$ y $B \rightarrow C$.

Así, nuestra portada canónica es:

$$A \rightarrow B$$

$$B \rightarrow C$$





7.4.4 Preservacion de la dependencia (Asegurar que las relaciones entre los datos se mantengan correctamente)

Sea F un conjunto de dependencias funcionales de un esquema R , y sea R_1, R_2, \dots, R_n una descomposición de R . La restricción de F a R_i es el conjunto F_i de todas las dependencias funcionales en F^+ que incluyen sólo atributos de R_i . Dado que todas las dependencias funcionales en una restricción involucran atributos de un solo esquema de relación, es posible probar la satisfacción de dicha dependencia verificando solo una relación.

Por ejemplo:

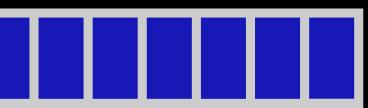
Supongamos que $F = \{A \rightarrow B, B \rightarrow C\}$, y tenemos una descomposición en AC y AB . La restricción de F a AC incluye $A \rightarrow C$, ya que $A \rightarrow C$ está en F^+ , aunque no esté en F .

```
calcular F+;
para cada esquema Ri en D
    comienza
        Fi := la restricción de F+ a Ri ;
    fin
```

```
F' :=  
para cada restricción Fi
    comienza
        F' = F'   Fi
```

```
finalizar el cálculo F'  
+; si (F'+ = F+) entonces devuelve (verdadero)
en caso contrario devuelve (falso);
```

Figura 7.10 Pruebas para la preservación de la dependencia.



Supongamos que tenemos dos tablas en una base de datos:

ID_empleado	Nombre_empleado	Departamento	Edad
1	Juan	Ventas	30
2	Maria	Marketing	35
3	Carlos	Ventas	28

Y tenemos el conjunto de dependencias funcionales

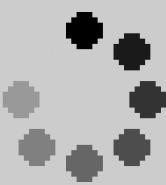
$$F = \{ID_empleado \rightarrow Edad, Departamento \rightarrow Nombre_Jefe\}.$$

Ahora, veamos cómo se aplica la restricción de F a la tabla de empleados:

1. Restricción de F a la tabla de empleados:

- En la tabla de empleados, los atributos relevantes son ID_empleado y Edad.
- Dado que $ID_empleado \rightarrow Edad$ es una dependencia que se puede derivar de F, pertenece a F^+ , el cierre de F.
- Por lo tanto, la restricción de F a la tabla de empleados incluye la dependencia $ID_empleado \rightarrow Edad$, aunque no esté explícitamente en F.

Departamento	Nombre_Jefe
Ventas	Ana
Marketing	Luis



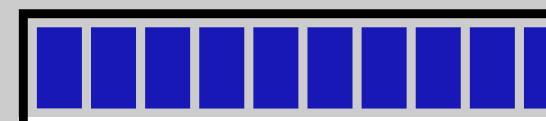
Métodos para probar la preservación de dependencias en una descomposición de un esquema relacional.

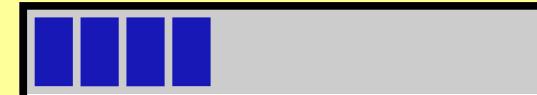
1. Prueba Directa en las Relaciones de la Descomposición:

- Se prueba cada dependencia funcional en el conjunto F en las relaciones de la descomposición.
- No siempre garantiza la preservación de todas las dependencias, pero es sencillo de aplicar.

1. Enfoque del Cierre de Atributos Modificado:

- No requiere calcular el cierre completo de F^+ .
- Se calcula el cierre de atributos de cada dependencia en F en cada subesquema de la descomposición.
- Si todas las dependencias se conservan en cada subesquema, entonces la descomposición preserva la dependencia.

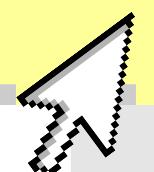
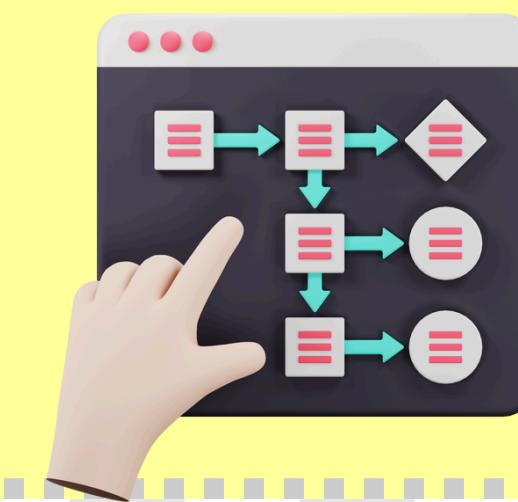


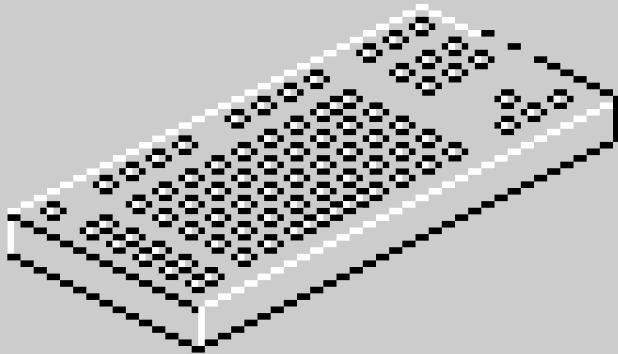
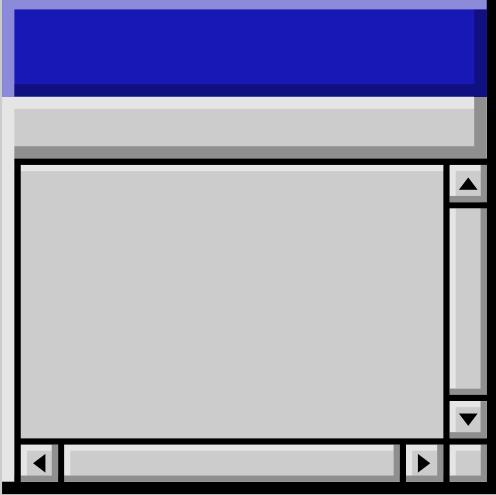


7.5

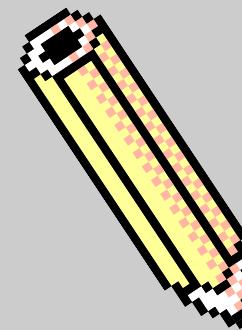
Algoritmos de descomposición mediante dependencias funcionales

[Back to Agenda Page](#)

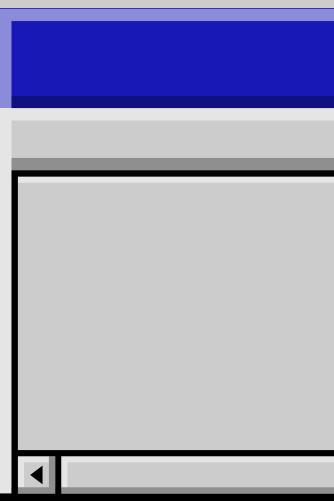




7.5.1 BCNF Decomposition



Se puede utilizar directamente para probar si una relación está en BCNF.
(Forma Normal de Boyce-Codd)



Testing for BCNF

- Para comprobar si existe una dependencia no trivial $\alpha \rightarrow \beta$ causa una violación de BCNF, calcular y verificar que incluye todos los atributos de R, es decir, es una superclave para R.
- Para comprobar si un esquema de relación R está en BCNF, basta con verificar solo las dependencias en el conjunto.

```
result := {R};  
done := false;  
while (not done) do  
    if (there is a schema  $R_i$  in result that is not in BCNF)  
        then begin  
            let  $\alpha \rightarrow \beta$  be a nontrivial functional dependency that holds  
            on  $R_i$  such that  $\alpha^+$  does not contain  $R_i$  and  $\alpha \cap \beta = \emptyset$ ;  
            result := (result -  $R_i$ )  $\cup$  ( $R_i - \beta$ )  $\cup$  (  $\alpha, \beta$  );  
        end  
    else done := true;
```

- Para cada subconjunto α de atributos en R_i , comprueba que α^+ no incluye ningún atributo de $R_i - \alpha$, o incluye todos los atributos de R_i .

$$\alpha \rightarrow (\alpha^+ - \alpha) \cap R_i.$$

BCNF Decomposition Algorithm

Ahora podemos establecer un método general para descomponer un esquema de relación de modo que satisfaga BCNF.

Por ejemplo:

Supongamos que tenemos un diseño de base de datos que utiliza el class-relación, cuyo esquema es el que se muestra a continuación.

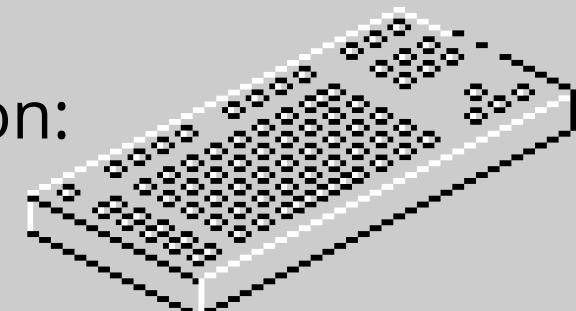
*class (course_id, title, dept_name, credits, sec_id, semester, year, building,
room_number, capacity, time_slot_id)*

- El conjunto de dependencias funcionales que debemos mantener en este esquema son:

*course_id → title, dept_name, credits
building, room_number → capacity
course_id, sec_id, semester, year → building, room_number, time_slot_id*

- La dependencia funcional:

course_id → title, dept_name, credits



Por ejemplo

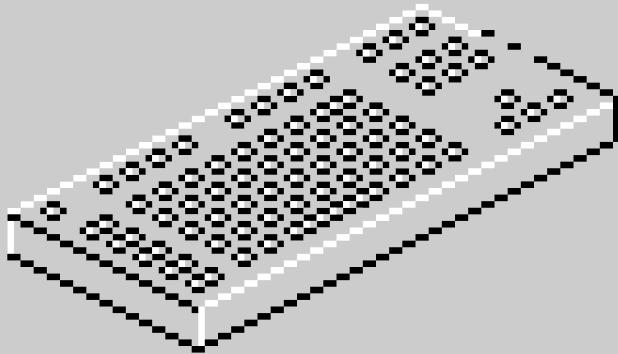
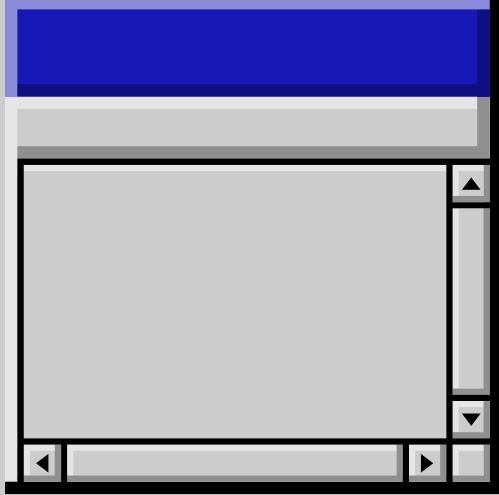
- Pero course_id no es una superclave. Reemplazamos class con dos relaciones con los siguientes esquemas:

course (course_id, title, dept_name, credits)
*class-1 (course_id, sec_id, semester, year, building, room_number
capacity, time_slot_id)*

- Una clave candidata para class-1 es {course id, sec id, semester, year}. La dependencia funcional:
 $building, room_number \rightarrow capacity$
- En class-1, pero {building, room number} no es una superclave para class-1. reemplazamos class-1 dos relaciones con los siguientes esquemas:

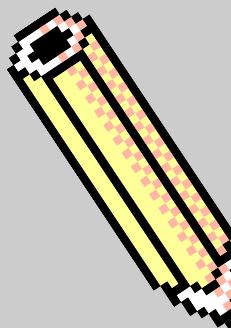
classroom (building, room_number, capacity)
*section (course_id, sec_id, semester, year,
building, room_number, time_slot_id)*

- Estos dos esquemas están en BCNF.



7.5.2 3NF Decomposition

(Tercera Forma Normal)



Tenga en cuenta que el algoritmo considera el conjunto de esquemas. $R_j, j=1, 2, \dots, i$; inicialmente $i=0$, y en este caso el conjunto está vacío.

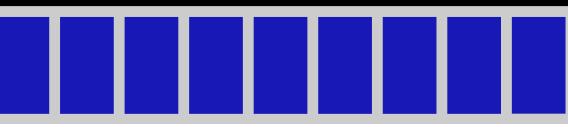
dept_advisor (s_ID, i_ID, dept_name)

- Está en 3NF aunque no esté en BCNF. El algoritmo utiliza las siguientes dependencias funcionales en F:

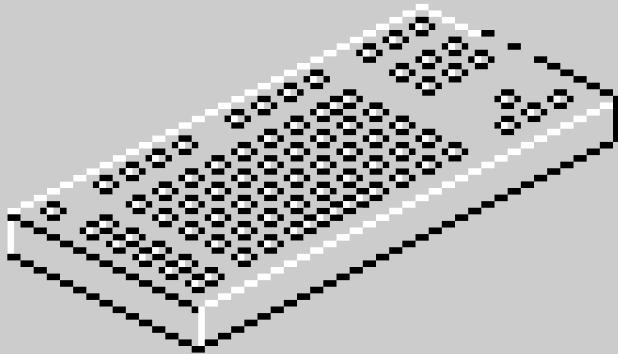
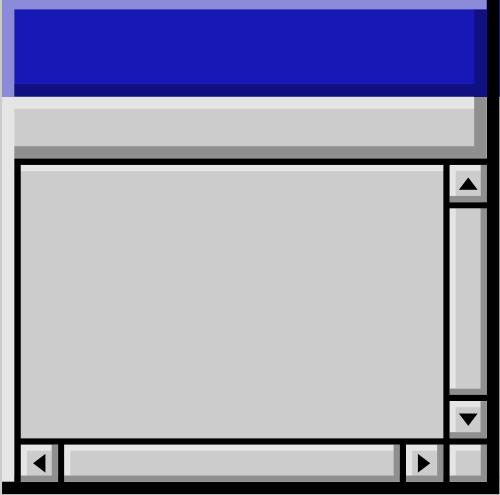
$f_1: i_ID \rightarrow dept_name$
 $f_2: s_ID, dept_name \rightarrow i_ID$

- No hay atributos extraños en ninguna de las dependencias funcionales en F, entonces FC contiene F1 y F2. El algoritmo luego genera como R1 el esquema, $(i_ID, dept_name)$, y como R2 el esquema $(s_ID, dept_name, i_ID)$. Luego el algoritmo encuentra que R2 contiene una clave candidata, por lo que no se crea ningún esquema de relación adicional.

El algoritmo 3NF puede producir resultados que cumplen tanto con 3NF como con BCNF. Esto sugiere un enfoque para generar un diseño en BCNF: primero utilizar el algoritmo 3NF y luego, para los esquemas que no cumplen con BCNF, aplicar el algoritmo BCNF. Si la descomposición resultante no preserva la dependencia, se debe regresar al diseño 3NF.

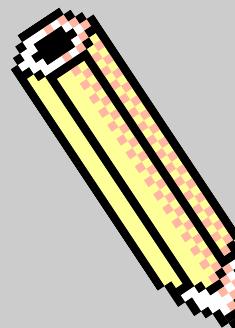


```
let  $F_c$  be a canonical cover for  $F$ ;
 $i := 0$ ;
for each functional dependency  $\alpha \rightarrow \beta$  in  $F_c$ 
     $i := i + 1$ ;
     $R_i := \alpha\beta$ ;
if none of the schemas  $R_j, j = 1, 2, \dots, i$  contains a candidate key for  $R$ 
    then
         $i := i + 1$ ;
         $R_i :=$  any candidate key for  $R$ ;
/* Optionally, remove redundant relations */
repeat
    if any schema  $R_j$  is contained in another schema  $R_k$ 
        then
            /* Delete  $R_j$  */
             $R_j := R_i$ ;
             $i := i - 1$ ;
until no more  $R_j$ s can be deleted
return  $(R_1, R_2, \dots, R_i)$ 
```

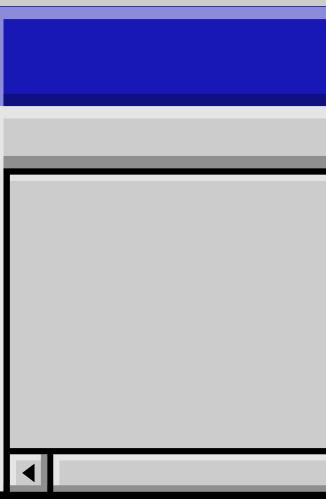


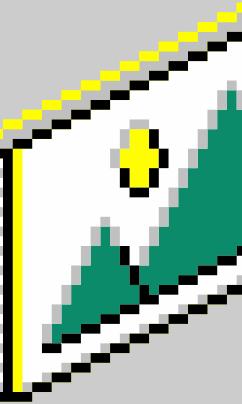
7.5.3

Correctness of the 3NF Algorithm



El algoritmo 3NF es una técnica para descomponer esquemas de bases de datos en formas que preserven las dependencias funcionales.





Supongamos que la dependencia que generó R_i en el algoritmo de síntesis es $\alpha \rightarrow \beta$. B debe estar en α o en β , ya que B está en R_i y $\alpha \rightarrow \beta$ generó R_i . Consideraremos los tres posibles casos:

CASO 1

- B está en α y en β .
En este caso, la dependencia $\alpha \rightarrow \beta$ no estaría en F_c , ya que B sería superfluo en β . Por lo tanto, este caso no puede ser válido.

CASO 2

- B está en β pero no en α . Consideraremos dos casos:
 - γ es una superclave. Se satisface la segunda condición de la 3NF.
 - Si γ no es una superclave, entonces α debe contener algún atributo que no esté en γ . Dado que $\gamma \rightarrow B$ está en F^+ , no puede depender de $\alpha \rightarrow \beta$, lo que implica que γ debe ser una superclave si B está en β , cumpliendo la segunda condición de la 3NF.

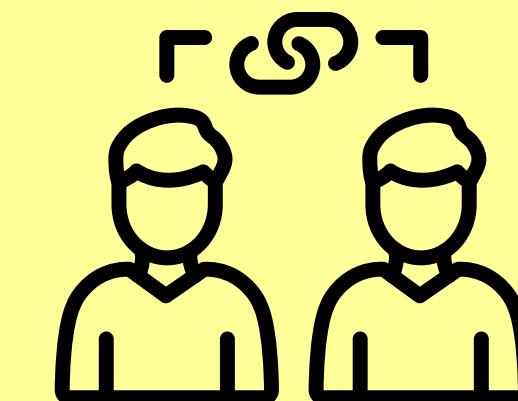
CASO 3

- B está en α pero no en β . Dado que α es una clave candidata, se satisface la tercera alternativa en la definición de 3NF.

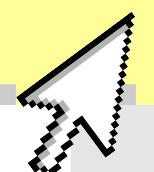


7.6

"Optimización del Diseño de Bases de Datos mediante Dependencias Multivaluadas y Cuarta Forma Normal (4NF)"



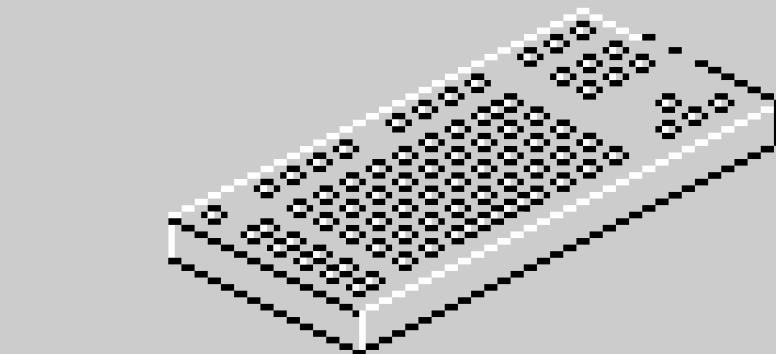
[Back to Agenda Page](#)





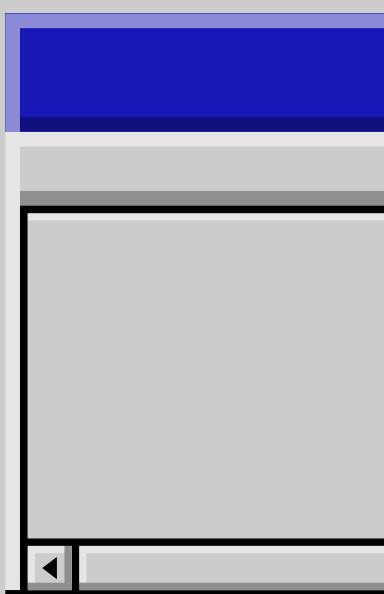
Dependencias Funcionales vs. Multivaluadas

¿Qué son las dependencias funcionales y multivaluadas? Las dependencias funcionales determinan las relaciones entre los atributos de una relación, mientras que las dependencias multivaluadas requieren la presencia de ciertas tuplas para satisfacer condiciones específicas.

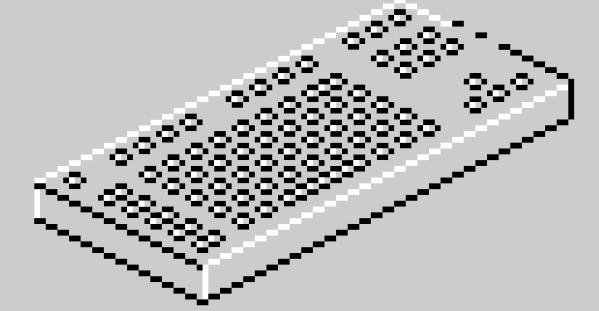


-- Ejemplo de dependencia funcional

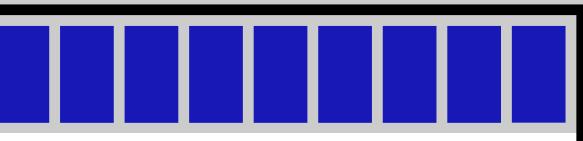
- `SELECT nombre, direccion FROM instructores WHERE ID = 1;`



Cuarta Forma Normal (4NF)

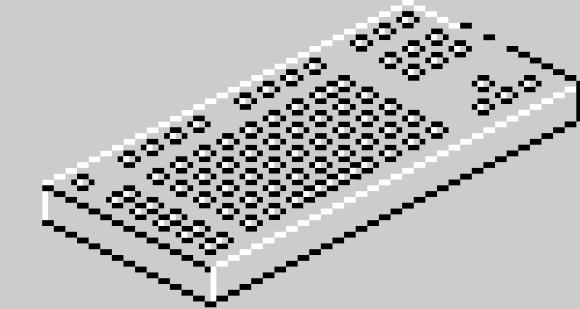


La Cuarta Forma Normal (4NF) es una forma normal más restrictiva que la Forma Normal de Boyce-Codd (BCNF). Garantiza la eliminación de la redundancia de información en las bases de datos al tener en cuenta dependencias multivaluadas.



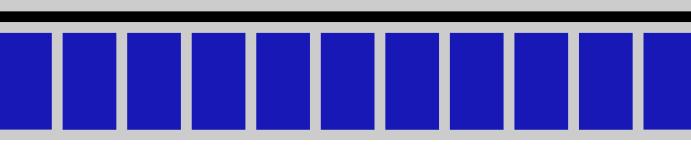
Descomposición en 4NF

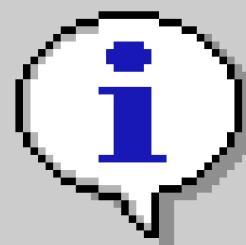
La descomposición en 4NF es un proceso que utiliza un algoritmo similar al de BCNF, pero considerando dependencias multivaluadas. Se asegura la preservación de dependencias y la ausencia de pérdida de información.



```
-- Ejemplo de descomposición en 4NF
• ⊖ CREATE TABLE Instructores_Departamentos (
    ID INT,
    Departamento VARCHAR(100),
    PRIMARY KEY (ID, Departamento)
);

• ⊖ CREATE TABLE Instructores_Direcciones (
    ID INT,
    Direccion VARCHAR(100),
    PRIMARY KEY (ID, Direccion)
);
```





Ejemplo Práctico

Veamos un ejemplo práctico de descomposición en 4NF. Utilizaremos una base de datos de una universidad donde los instructores pueden estar asociados con múltiples departamentos y tener múltiples direcciones.

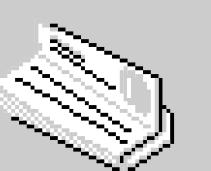
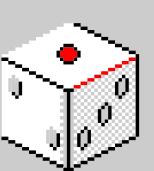
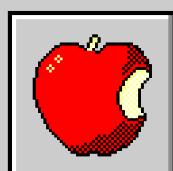
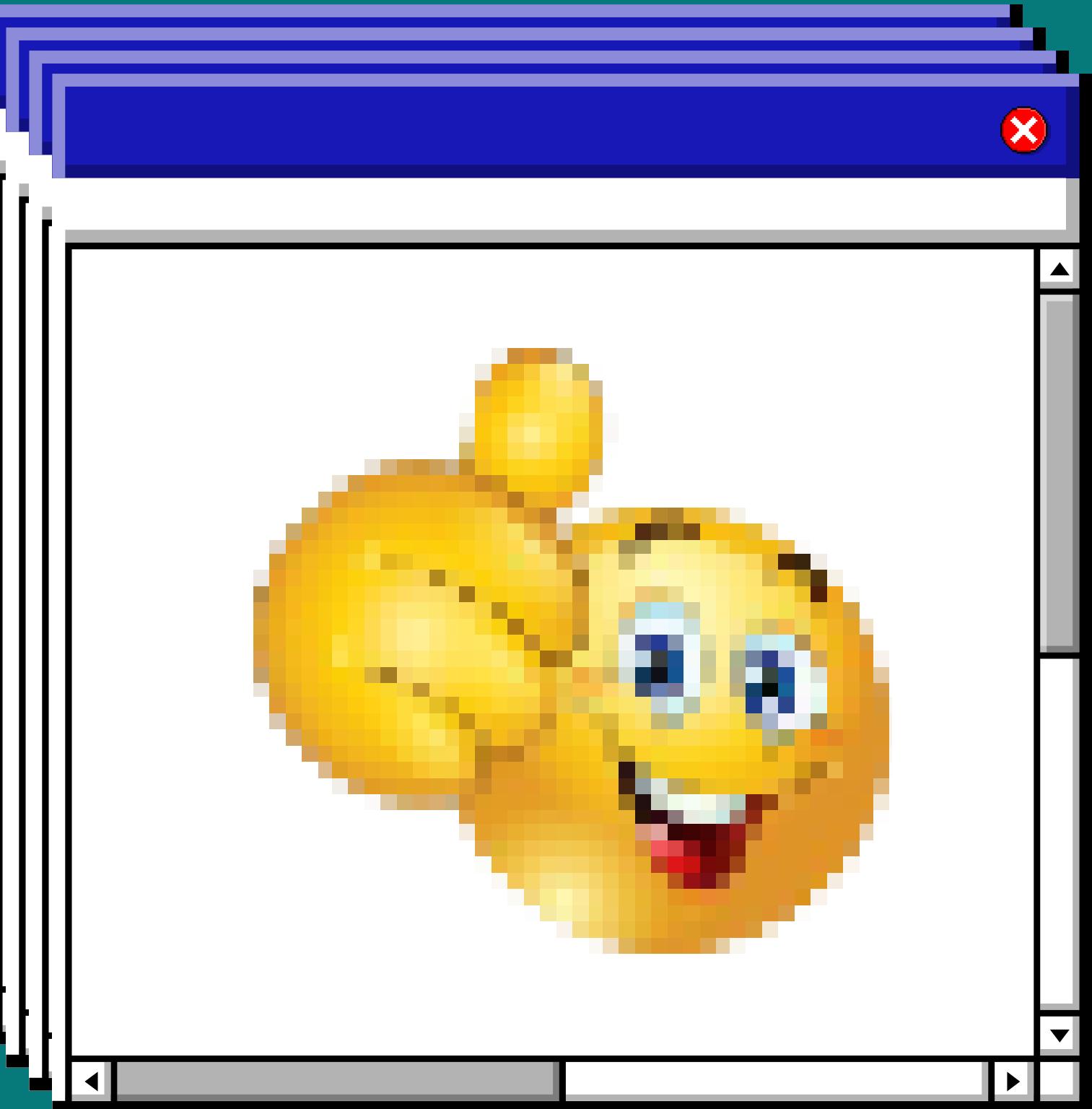
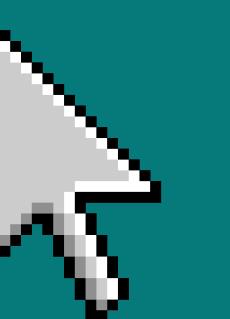
```
-- Tabla original de instructores
• CREATE TABLE Instructores (
    ID INT PRIMARY KEY,
    Nombre VARCHAR(100),
    Departamento VARCHAR(100),
    Direccion VARCHAR(100)
);

-- Insertar datos de ejemplo
• INSERT INTO Instructores (ID, Nombre, Departamento, Direccion) VALUES
(1, 'Juan Pérez', 'Matemáticas', 'Calle A, Ciudad X'),
(1, 'Juan Pérez', 'Física', 'Calle B, Ciudad Y');

-- Descomposición en 4NF
-- Tabla de ID e información de departamento
• CREATE TABLE Instructores_Departamentos (
    ID INT,
    Departamento VARCHAR(100),
    PRIMARY KEY (ID, Departamento),
    FOREIGN KEY (ID) REFERENCES Instructores(ID)
);

-- Tabla de ID y dirección
• CREATE TABLE Instructores_Direcciones (
    ID INT,
    Direccion VARCHAR(100),
    PRIMARY KEY (ID, Direccion),
    FOREIGN KEY (ID) REFERENCES Instructores(ID)
);
```

¡Gracias por
su atención!



[Back to Agenda Page](#)