IIC2413

AYUDANTÍA 3

Formas Normales

ANOMALÍAS

Rompen la consistencia y pueden causar problemas

PedidoResumen(id_pedido, cliente, comuna, costo_envío)

id	cliente	comuna	costo_envío
501	Ana	Ñuñoa	3500
502	Bruno	Ñuñoa	3500
503	Camila	La Florida	4200
504	Diego	Ñuñoa	3500

Redundancia: costo_envio = 3500 se repite 3 veces

Inserción: no puedo guardar costo de una comuna si no existe pedido

Actualización: si cambia el costo de alguna comuna, tengo que modificar todos los pedidos

Eliminación: si borro el único pedido de La Florida pierdo el dato

DESCOMPOSICIÓN

Evita posibles anomalias

comuna → costo_envío

Pedido(id_pedido, cliente, comuna)

CostoComuna(comuna, costo_envío)

La dependencia funcional marca la separación en la tabla

DEPENDENCIAS

X → Y significa: X determina Y

sku → nombre_producto, precio

patente → marca, modelo

sigla_curso → nombre, facultad

DEPENDENCIAS

Si X determina todo, es llave

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

$Y \subseteq X \Rightarrow X \rightarrow Y$

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

$X \rightarrow Y$ y $X \rightarrow Z \Rightarrow X \rightarrow Y,Z$

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

EJEMPLO

VentaBruta(id_venta, sku, nombre_producto, proveedor, email_proveedor, cantidad)

- sku → nombre_producto, proveedor
- proveedor → email_proveedor
- id_venta → sku, cantidad

Producto(sku, nombre_producto, proveedor)

Proveedor(proveedor, email_proveedor)

Venta(id_venta, sku, cantidad)

1NF

Cada celda debe tener un solo valor
2NF y 3NF mantienen las condiciones

id_playlist	canciones
10	A, B, C
11	D, E

id_playlist	canción
10	A
10	B
10	C
11	D
11	E

2NF

Evita dependencia parcial
3NF mantiene las condiciones

DetallePedido(id_pedido, sku, nombre_producto, cantidad)

Llave compuesta
{id_pedido, sku}

sku → nombre_producto

Solo hace falta sku para determinar el nombre del producto (dependencia parcial viola 2NF)

PedidoItem(id_pedido, sku, cantidad) + Producto(sku, nombre_producto)

3NF

Una dependencia X → Y es 3NF si:

- X es una superllave
- Y forma parte de una llave candidata minimal

Empleado(rut, depto, jefe)

rut → depto, jefe
depto → jefe

Si consideramos rut como llave, la primera dependencia cumple, pero la segunda no está en ningún caso

Empleado(emp_id, nombre, depto_id)
Departamento(depto_id, nombre_depto)

BCNF

Una dependencia X → Y es BCNF si:

- X es una superllave

Algoritmo para descomponer BCNF

1. Detectar la violación
2. Separar R1(X U Y)
3. Separar R2(A-Y)

La descomposición tiene que ser sin pérdida de información → si hacemos el Join volvemos a la tabla original

EJERCICIOS

EJERCICIOS

Usted ha sido contratado para analizar y rediseñar una base de datos utilizada por una empresa de transporte. Actualmente, la información sobre los envíos realizados por clientes se encuentra almacenada en una única tabla, como se muestra a continuación:

id_envío	cliente	rut_cliente	peso_kg	fecha	ciudad_entrega
1001	Sofía	15.815.461-7	10	2025-10-25	Santiago, Rancagua
1002	Paula	17.531.050-9	25	2025-11-06	Santiago, Talca, Curicó
1003	Ignacio	12.899.241-3	7	2025-03-17	Antofagasta, Calama
1004	Olivia	14.923.887-5	3	2025-01-21	Santiago, Temuco, Valdivia
1005	Consuelo	20.362.056-K	18	2025-09-08	Iquique, Arica
1006	Agustín	4.816.960-0	2	2025-08-30	Puerto Montt

Cuadro 5: Tabla `Envíos`

La columna `ciudad_entrega` contiene una lista con los nombres de las ciudades que forman parte del trayecto de entrega de cada envío. Un mismo envío puede pasar por una o más ciudades dependiendo de su destino, y el orden no importa.

EJERCICIOS

id_envío	cliente	rut_cliente	peso_kg	fecha	ciudad_entrega
1001	Sofía	15.815.461-7	10	2025-10-25	Santiago, Rancagua
1002	Paula	17.531.050-9	25	2025-11-06	Santiago, Talca, Curicó
1003	Ignacio	12.899.241-3	7	2025-03-17	Antofagasta, Calama
1004	Olivia	14.923.887-5	3	2025-01-21	Santiago, Temuco, Valdivia
1005	Consuelo	20.362.056-K	18	2025-09-08	Iquique, Arica
1006	Agustín	4.816.960-0	2	2025-08-30	Puerto Montt

Cuadro 5: Tabla `Envíos`

a) (5 pts) Identifique todas las dependencias funcionales presentes en la tabla.

EJERCICIOS

a) (5 pts) Identifique todas las dependencias funcionales presentes en la tabla.

Solución:

- `id_envío` → `cliente, rut_cliente, peso_kg, fecha, ciudad_entrega`
- `rut_cliente` → `cliente`

EJERCICIOS

id_envío	cliente	rut_cliente	peso_kg	fecha	ciudad_entrega
1001	Sofía	15.815.461-7	10	2025-10-25	Santiago, Rancagua
1002	Paula	17.531.050-9	25	2025-11-06	Santiago, Talca, Curicó
1003	Ignacio	12.899.241-3	7	2025-03-17	Antofagasta, Calama
1004	Olivia	14.923.887-5	3	2025-01-21	Santiago, Temuco, Valdivia
1005	Consuelo	20.362.056-K	18	2025-09-08	Iquique, Arica
1006	Agustín	4.816.960-0	2	2025-08-30	Puerto Montt

Cuadro 5: Tabla `Envíos`

b) (5 pts) Indique si la tabla se encuentra en 1NF, 2NF o 3NF. Justifique su respuesta.

EJERCICIOS

b) (5 pts) Indique si la tabla se encuentra en 1NF, 2NF o 3NF. Justifique su respuesta.

La tabla no se encuentra en 1NF, ya que el atributo `ciudad_entrega` contiene listas de ciudades, lo que viola el principio de atomicidad de la 1FN. Por lo tanto, tampoco cumple con 2NF ni 3NF.

EJERCICIOS

id_envío	cliente	rut_cliente	peso_kg	fecha	ciudad_entrega
1001	Sofía	15.815.461-7	10	2025-10-25	Santiago, Rancagua
1002	Paula	17.531.050-9	25	2025-11-06	Santiago, Talca, Curicó
1003	Ignacio	12.899.241-3	7	2025-03-17	Antofagasta, Calama
1004	Olivia	14.923.887-5	3	2025-01-21	Santiago, Temuco, Valdivia
1005	Consuelo	20.362.056-K	18	2025-09-08	Iquique, Arica
1006	Agustín	4.816.960-0	2	2025-08-30	Puerto Montt

Cuadro 5: Tabla `Envíos`

c) (5 pts) Proponga una descomposición del esquema que cumpla con BCNF, mostrando el nuevo conjunto de tablas y sus llaves primarias. Si no es posible, justifique por qué no es posible.

EJERCICIOS

c) (5 pts) Proponga una descomposición del esquema que cumpla con BCNF, mostrando el nuevo conjunto de tablas y sus llaves primarias. Si no es posible, justifique por qué no es posible.

Se propone la siguiente descomposición en BCNF:

- `Cliente(rut_cliente: string, cliente: string)`
  Donde `rut_cliente` → `cliente`

- `Envío(id_envío: int, rut_cliente: string, peso_kg: int, fecha: date)`
  Donde `id_envío` → `rut_cliente`, `peso_kg`, `fecha`
- `EnvíoCiudad(id_envío: int, nombre_ciudad: string)`
  No hay dependencias funcionales no triviales fuera de la llave primaria.

En la tabla `Cliente`, el determinante de la única dependencia funcional es `rut_cliente`, el cual es llave primaria. En la tabla `Envío` ocurre lo mismo con la llave `id_envío`. En `EnvíoCiudad`, todos los atributos dependen únicamente de la llave compuesta (`id_envío`, `nombre_ciudad`). Por ende, la descomposición propuesta garantiza que toda dependencia funcional esté determinada por una llave candidata, lo que cumple con BCNF.

Nota de corrección: Se asume que el nombre de una ciudad es único. En caso de asumir lo contrario, basta con definir una nueva tabla que relacione el nombre de una ciudad con un id serial, e incluir el id en la tabla `EnvíoCiudad` en lugar del nombre de la ciudad.