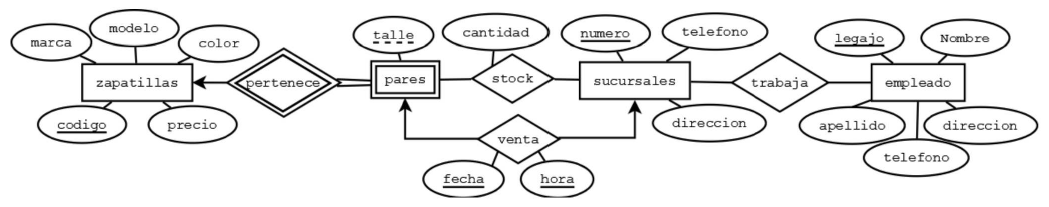
**Examen BdD N°1**

**Ejercicio 1.** El siguiente modelo E-R corresponde a una base de datos utilizada por una cadena deportiva que se dedica exclusivamente a la venta de zapatillas.



Se desean realizar algunas modificaciones y extensiones sobre la base de datos para contar con la siguiente información:

* Debido a un cambio en la política de precios, el precio de cada zapatilla puede variar según el talle. Se desea conocer el precio de cada talle disponible para cada zapatilla.
* Se desea conocer en qué sucursal trabaja cada empleado en un determinado día de la semana (lunes, martes,…). Si bien un empleado puede trabajar en muchas sucursales, solo puede trabajar en una única sucursal por día. En una sucursal trabajan muchos empleados por día.
* Para poder calcular la comisión que le corresponde a un empleado por cada venta que realiza, es necesario registrar qué empleado realizó cada venta. Se desea restringir que el empleado asociado a una venta trabaje en la sucursal donde se realiza la venta.
* Se quieren registrar los cambios de zapatillas realizados. Para cada cambio se desea conocer la fecha, hora y sucursal donde se realizó, qué par de zapatillas se devolvió y qué par de zapatillas se llevaron a cambio. Tenga en cuenta que en una sucursal, en una determinada fecha y hora, solo puede realizar un cambio.
* Por otra parte, se desean registrar devoluciones. A diferencia de un cambio, en una devolución un par de zapatillas se cambia por una nota de crédito. Cada nota de crédito tiene un número identificatorio único, un monto de dinero y una fecha de vencimiento. Para cada devolución es necesario registrar la fecha, hora y sucursal donde se realiza, el par de zapatillas que se devuelve y la nota de crédito correspondiente. Una nota de crédito está asociada siempre a una sola devolución y por cada devolución siempre se genera una sola nota de crédito.

**a)** Obtenga un modelo E-R completo, modificando el anterior para incorporar la información y restricciones pedidas.

**b)** Obtenga el modelo relacional al modelo E-R obtenido en el a) señalando una clave primaria para cada clave primaria para cada relación y las llaves foráneas correspondientes.

ZAPATILLAS(código, marca, modelo, color)

PARES(talle, precio)

FK(código) referencia a ZAPATILLAS(código)

STOCK(talle, número, cantidad)

FK(talle) referencia a PARES(talle)

FK(número) referencia a SUCURSALEE(número)

SUCURSALES(número, teléfono, dirección)

EMPLEADO(legajo, nombre, apellido, teléfono, dirección)

TRABAJA(día, legajo, número)

FK(legajo) referencia a EMPLEADO(legajo)

FK(número) referencia a SUCURSALES(número)

VENTA(fecha\_venta, hora\_venta, número, legajo, día, talle)

FK(legajo) referencia a EMPLEADOS(número)

FK(número) referencia a SUCURSALES(número)

FK(día) referencia a TRABAJA(día)

FK(talle) referencia a PARES(talle)

DEVOLUCION(fecha, hora, nro\_identificatorio)

NOTA\_DE\_CREDITO(nro\_identificatorio, monto, fecha\_venc)

SE\_REALIZA\_EN(nro\_identificatorio, número)

FK(nro\_identificatorio) referencia a NOTA\_DE\_CREDITO(nro\_identificatorio)

FK(número) referencia a SUCURSALES(número)

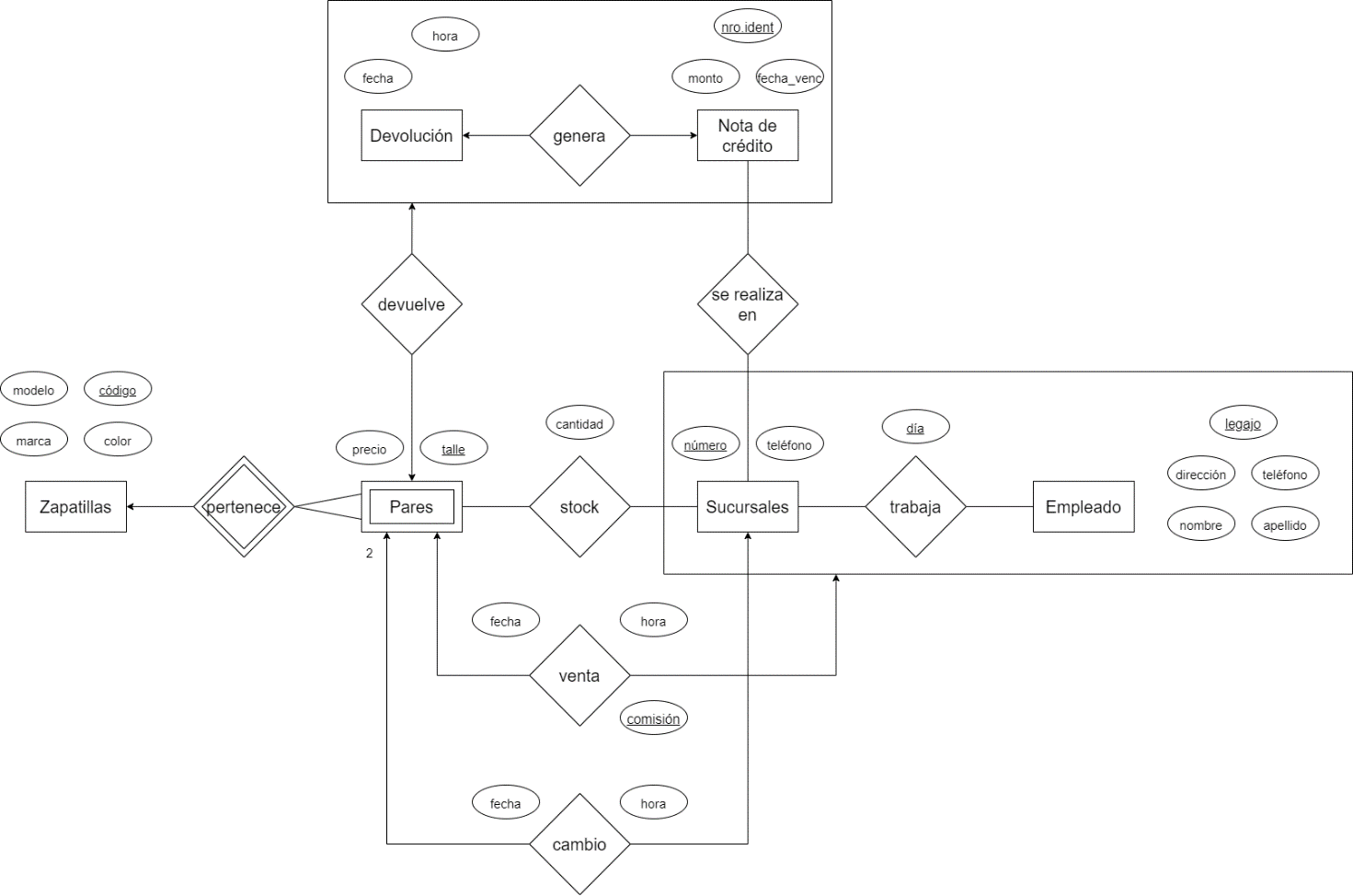
DEVUELVE(fecha, hora, nro\_identificatorio, talle)

FK(fecha) referencia a DEVOLUCIÓN(fecha)

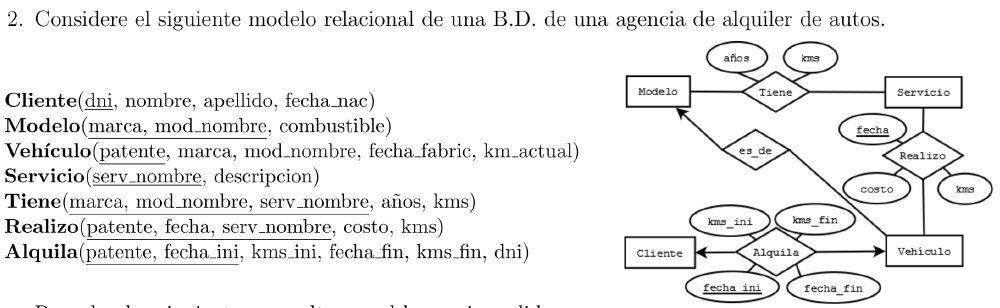
FK(hora) referencia a DEVOLUCIÓN(hora)

FK(nro\_identificatorio) referencia a NOTA\_DE\_CREDITO(nro\_identificatorio)

FK(talle) referencia a PARES(talle)



**Ejercicio 2.** Considere el siguiente modelo relacional de una B.D. de una agencia de alquiler de autos.



Resuelva las siguientes consultas en el lenguaje pedido:

**a)** A.R.: Vehículos a los cuales se les realizó un servicio que NO correspondía con su modelo (relación Tiene). Devuelva la patente del vehículo, marca, modelo, nombre del servicio, fecha y costo

ServiciosRealizados 🡨 Πpatente, serv\_nombre(Vehículo ⋈ Realizo)

ServiciosPosibles 🡨 Πpatente, serv\_nombre(Vehículo ⋈ Tiene)

ServicioDeDistintoModelo 🡨 ServiciosRealizados \ ServiciosPosibles

Πtiene.patente, tiene.marca, tiene.mod\_nombre, realizo.serv\_nombre, realizo.fecha, realizo.costo( ServiciosDeDistintoModelo ⋈ Realizo ⋈ Vehículo )

**b)** A.R. o SQL (a elección): DNI de los clientes que alquilaron un vehículo naftero (combustible = “nafta”) y luego alquilaron un vehículo gasolero (combustible = “gasoil”) de la misma marca. Puede utilizar los operadores relacionales (>, <, =, …) para comparar fechas.

Tabla (Modelo ⋈ Alquila ⋈ Vehículo) =

**Marca | mod\_nombre |** combustible | **patente** | fecha\_ini | fecha\_fin | kms\_ini | kms\_fin | dni | fecha\_fabric | kms\_actual

AlquilaNaftero 🡨 𝜎combustible = “nafta”(Modelo ⋈ Alquila ⋈ Vehículo)

AlquilaGasolero 🡨 𝜎combustible = “gasoil”(Modelo ⋈ Alquila ⋈ Vehículo)

Πalquila.dni(𝜎AlquilaNaftero.dni = AlquilaGasolero.dni ∧ AlquilaNaftero.fecha < AlquilaGasolero.fecha ∧ AlquilaNaftero.marca = AlquilaGasolero.marca

∧ AlquilaNaftero.patente ≠ AlquilaGasolero.patente(AlquilaNaftero x AlquilaGasolero))

SELECT alquilaNaftero.dni

FROM (Alquila A1 NATURAL JOIN Vehiculo V1 NATURAL JOIN Modelo M1) AS alquilaNaftero,

(Alquila A2 NATURAL JOIN Vehiculo V2 NATURAL JOIN Modelo M2) AS alquilaGasolero

WHERE alquilaNaftero.dni = alquilaGasolero.dni AND alquilaNaftero.fecha < alquilaGasolero.fecha AND alquilaNaftero.marca = alquilaGasolero.marca AND alquilaNaftero.patente ≠ alquilaGasolero.patente AND

M1.combustible = “nafta” AND M2.combustible = “gasoil”;

**c)** SQL:

i) Devuelva para cada alquiler el dni del cliente y el kilometraje recorrido (kms.fin – kms.ini)

SELECT dni, (kms\_fin – kms\_ini) AS kms

FROM Alquila

INTO temp kilometraje

ii) Devuelva para cada cliente la cantidad total de kilómetros recorridos en todos sus alquileres, incluyendo la consulta i) como una subconsulta en la cláusula FROM. Deberá devolver el nombre y apellido del cliente y el campo con los kilómetros deberá llamarse “total kilómetros recorridos”.

SELECT dni, nombre, apellido, sum(kms) AS total\_kilómetros\_recorridos

FROM kilometraje NATURAL JOIN Cliente

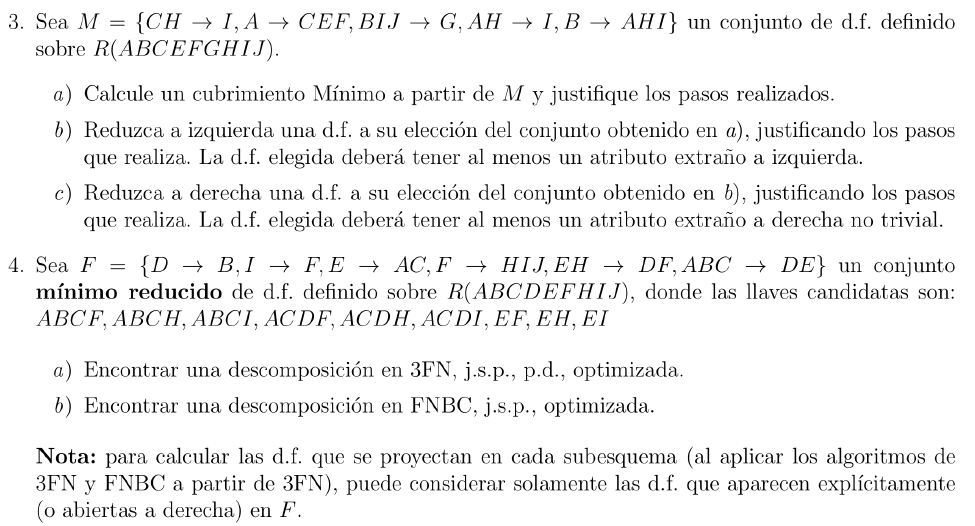
GROUP by dni, nombre, apellido

INTO temp infoCliente

SELECT nombre, apellido, total\_kilómetros\_recorridos

FROM infoCliente

**Ejercicio 3**



(CH)+ = CHI, (A)+ = ACEF, (BIJ)+ = BIJGAHCEF, (AH)+ = AHICEF, (B)+ = BAHICEF

Luego, el **conjunto cerrado en atributos** es

M1 = { CH 🡪 CHI, A 🡪 ACEF, BIJ 🡪 ABCEFGHIJ, AH 🡪 ACEFHI, B 🡪 ABCEFHI }

**Eliminación de dependencias redundantes:**

(CH)M1 / { CH 🡪 CHI }+ = CH. Como CH ⊄ CHI, CH 🡪 CHI no es redundante.

(A)M1 / { A 🡪 ACEF }+ = A. Como A ⊄ ACEF, A 🡪 ACEF no es redundante.

(BIJ)M1 / { BIJ 🡪 ABCEFGHIJ }+ = BIJACEFH. Como BIJACEFH ⊄ ABCEFGHIJ, BIJ 🡪 ABCEFGHIJ no es redundante.

(AH)M1 / { AH 🡪 ACEFHI }+ = AHCEFI. Como AHCEFI ⊆ ACEFHI, AH 🡪 ACEFHI es redundante.

Luego, M2 = { CH 🡪 CHI, A 🡪 ACEF, BIJ 🡪 ABCEFGHIJ, B 🡪 ABCEFHI }

(B)M2 / { B 🡪 ABCEFHI }+ = B. Como B ⊄ ABCEFGHIJ, B 🡪 ABCEFGHIJ no es redundante.

Luego, el cubrimiento mínimo para M es M2 = { CH 🡪 CHI, A 🡪 ACEF, BIJ 🡪 ABCEFGHIJ, B 🡪 ABCEFHI }.

**Reducción a izquierda.**

Para CH 🡪 CHI tenemos:

(H)M2+ = H. Como H ⊄ CHI, C no es extraño a izquierda.

(C)M2+ = C. Como C ⊄ CHI, H no es extraño a izquierda.

Para BIJ 🡪 ABCEFGHIJ tenemos:

(IJ)M2+ = IJ. Como H ⊄ CHI, B no es extraño a izquierda.

(BJ)M2+ = BJACEFHIG. Como BJACEFHIG ⊆ ABCEFGHIJ, I es extraño a izquierda.

Luego, M3 = { CH 🡪 CHI, A 🡪 ACEF, BJ 🡪 ABCEFGHIJ, B 🡪 ABCEFHI }

(BI)M3+ = BIACEFH. Como BIACEFH ⊄ ABCEFGHIJ, J no es extraño a izquierda.

**Reducción a derecha**. Remuevo los atributos triviales a derecha, tal que

M3 = { CH 🡪 I, A 🡪 CEF, BJ 🡪 ACEFGHI, B 🡪 ACEFHI }

Para A 🡪 CEF, tenemos:

(A)M3 / { A 🡪 CEF } U { A 🡪 EF } + = AEF. Como AEF ⊄ CEF, A no es extraño a derecha.

(A)M3 / { A 🡪 CEF } U { A 🡪 CF } + = ACF. Como ACF ⊄ CEF, C no es extraño a derecha.

(A)M3 / { A 🡪 CEF } U { A 🡪 CE } + = ACE. Como A ⊄ CEF, F no es extraño a derecha.

Para BJ 🡪 ACEFGHI, tenemos:

(BJ)M3 / { BJ 🡪 ACEFGHI } U { BJ 🡪 CEFGHI } + = BJCEFGHIA. Como BIJCEFGHA ⊆ ACEFGHI, A es extraño a derecha.

Luego, M4 = { CH 🡪 I, A 🡪 CEF, BJ 🡪 CEFGHI, B 🡪 ACEFHI }

(BJ)M4 / { BJ 🡪 CEFGHI } U { BJ 🡪 EFGHI } + = BJEFGHIAC. Como BJEFGHIAC ⊆ CEFGHI, C es extraño a derecha.

Luego, M5 = { CH 🡪 I, A 🡪 CEF, BJ 🡪 EFGHI, B 🡪 ACEFHI }

(BJ)M5 / { BJ 🡪 EFGHI } U { BJ 🡪 FGHI } + = BJFGHIACE. Como BIJFGHIACE ⊆ EFGHI, E es extraño a derecha.

Luego, M6 = { CH 🡪 I, A 🡪 CEF, BJ 🡪 FGHI, B 🡪 ACEFHI }

(BJ)M6 / { BJ 🡪 FGHI } U { BJ 🡪 GHI } + = BJGHIACEF. Como BJGHIACEF ⊆ FGHI, F es extraño a derecha.

Luego, M7 = { CH 🡪 I, A 🡪 CEF, BJ 🡪 GHI, B 🡪 ACEFHI }

(BJ)M7 / { BJ 🡪 GHI } U { BJ 🡪 HI } + = BJHI. Como BJHIACEF ⊄ GHI, G no es extraño a derecha.

(BJ)M7 / { BJ 🡪 GHI } U { BJ 🡪 GI } + = BJGIACEFH. Como BJGIACEFH ⊆ GHI, H es extraño a derecha.

Luego, M8 = { CH 🡪 I, A 🡪 CEF, BJ 🡪 GI, B 🡪 ACEFHI }

(BJ)M8 / { BJ 🡪 GI } U { BJ 🡪 G } + = BJGACEFHI. Como BJGACEFHI ⊆ GI, I es extraño a derecha.

Luego, M9 = { CH 🡪 I, A 🡪 CEF, BJ 🡪 G, B 🡪 ACEFHI }

Para B 🡪 ACEFHI, tenemos:

(B)M8 / { B 🡪 ACEFHI } U { B 🡪 CEFHI } + = BCEFHI. Como BCEFHI ⊄ ACEFHI, A no es extraño a derecha.

(B)M8 / { B 🡪 ACEFHI } U { B 🡪 AEFHI } + = BAEFHIC. Como BAEFHI ⊆ ACEFHI, C es extraño a derecha.

Luego, M9 = { CH 🡪 I, A 🡪 CEF, BJ 🡪 G, B 🡪 AEFHI }

(B)M9 / { B 🡪 AEFHI } U { B 🡪 AFHI } + = BAFHICE. Como BAFHICE ⊆ AEFHI, E es extraño a derecha.

Luego, M10 = { CH 🡪 I, A 🡪 CEF, BJ 🡪 G, B 🡪 AFHI }

(B)M10 / { B 🡪 AFHI } U { B 🡪 AHI } + = BAHICEF. Como BAHICEF ⊆ AFHI, F es extraño a derecha.

Luego, M11 = { CH 🡪 I, A 🡪 CEF, BJ 🡪 G, B 🡪 AHI }

(B)M11 / { B 🡪 AHI } U { B 🡪 AI } + = BAICEF. Como BAICEF ⊄ AHI, H no es extraño a derecha.

(B)M11 / { B 🡪 AHI } U { B 🡪 AH } + = BAHICEF. Como BAHCEF ⊆ AHI, I es extraño a derecha.

Luego, M12 = { CH 🡪 I, A 🡪 CEF, BJ 🡪 G, B 🡪 AH }

Finalmente, el conjunto mínimo reducido para M es **M12 = { CH 🡪 I, A 🡪 CEF, BJ 🡪 G, B 🡪 AH }**.

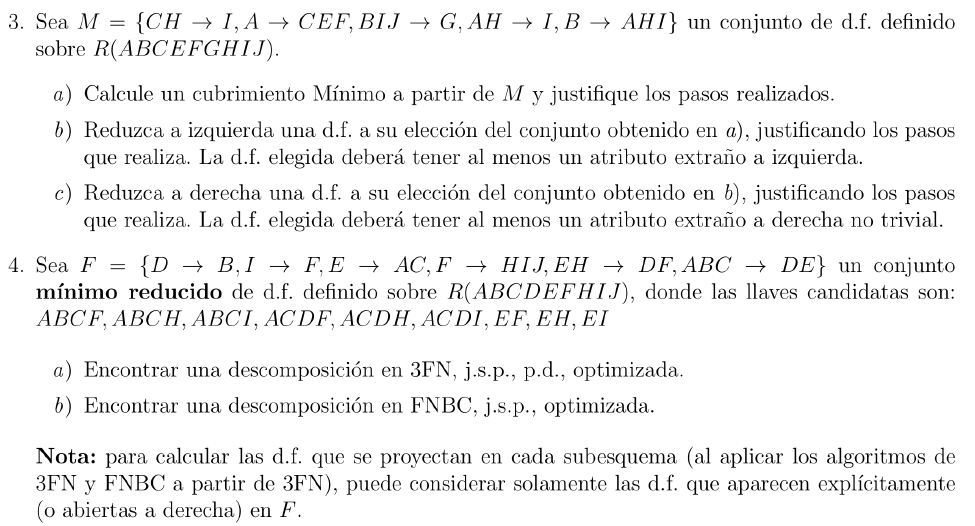
Llaves (no está solicitado en el ejercicio, pero lo hice igual).

Prioridad de atributos: BJ | ACH | IF

**BJ**M12+ = BJAHGCEFI. Como BJAHGCEFI ⊆ R, es llave.

Luego, **BJ** es llave candidata para la relación.

**Ejercicio 4**



**a)** Encontrar una descomposición en 3FN, j.s.p., p.d., optimizada

1. DF’s abiertas a derecha:

F1 = { ABC 🡪 D, ABC 🡪 E, EH 🡪 D, EH 🡪 F, D 🡪 B, I 🡪 F, E 🡪 A, E 🡪 C, F 🡪 H, F 🡪 I, F 🡪 J }

1. ABC 🡪 D no respeta 3FN ya que no su lado izquierdo no constituye una superllave.
2. Descomposición en subesquemas:

**ABC 🡪 D**. Izq(ABCD) = ABCD

AF1+ = A, BF1+ = B y CF1+ = C no deducen ninguna df no trivial

DF1+ = DB, se deduce D 🡪 B, que se proyecta sobre ABCD **(no será necesario descomponer la df D 🡪 B)**

ABF1+ = AB, ACF1+ = AC, BCF1+ = BC y BDF1+ = BD no deducen ninguna df no trivial

ADF1+ = ADB, se deduce AD 🡪 B, pero no se incorpora por ser redundante con D 🡪 B

CDF1+ = CDB, se deduce CD 🡪 B, pero no se incorpora por ser redundante con D 🡪 B

ABCF1+ es trivial

ABDF1+ = ABD y BCDF1+ = BCD no deducen ninguna df no trivial

ACDF1+ = ACDB, se deduce ACD 🡪 B, pero no se incorpora por ser redundante con D 🡪 B

Luego, las dependencias de F1 que se proyectan sobre ABCD son **ΠABCD(F1) = { ABCE 🡪 D, D 🡪 B }**

Llaves:

ACF1+ = AC. Como AC ⊄ R, no es llave.

ACBF1+ = ACBD. Como ABCD ⊆ R, es llave.

ACDF1+ = ACDB. Como ABCD ⊆ R, es llave.

Llaves: ABC, ACD

**ABC 🡪 E**. Izq(ABCE) = ABCE.

AF1+ = A, BF1+ = B y CF1+ = C no deducen ninguna df no trivial

EF1+ = EAC, se deduce E 🡪 AC, que se proyecta sobre ABCE

ABF1+ = AB, ACF1+ = AC y BCF1+ = BC no deducen ninguna df no trivial

BEF1+ = BEACD, se deduce BE 🡪 AC, no se incorpora por ser redundante con E 🡪 AC

AEF1+ = AEC, se deduce AE 🡪 C, que se proyecta sobre ABCE

CEF1+ = CEA, se deduce CE 🡪 A, que se proyecta sobre ABCE

ABCF1+ es trivial

ABEF1+ = ABECD, se deduce ABE 🡪 C, pero no se incorpora por ser redundante con AE 🡪 C

BCEF1+ = BCEA, se deduce BCE 🡪 A, pero no se incorpora por ser redundante con CE 🡪 A

ACEF1+ = ACE no deduce ninguna df no trivial

Luego, las dependencias de F1 que se proyectan sobre ABCE son

**ΠABCE(F1) = { ABC 🡪 E, E 🡪 AC, AE 🡪 C, CE 🡪 A }**

Llaves:

BF1+ = B. Como B ⊄ R, no es llave.

BAF1+ = BA. Como BA ⊄ R, no es llave.

BCF1+ = BC. Como BC ⊄ R, no es llave.

BEF1+ = BEAC. Como ABCE ⊆ R, es llave.

BACF1+ = BACE. Como ABCE ⊆ R, es llave.

Llaves: BE, ABC

**EH 🡪 D**. Izq(EHD) = EDH

EF1+ = EAC, se deduce E 🡪 AC, pero no se proyecta porque A,C ⊄ EDH

DF1+ = DB, se deduce D 🡪 B, pero no se proyecta porque B ⊄ EDH

HF1+ = H no deduce ninguna df no trivial

EDF1+ = EDACB, se deduce ED 🡪 ACB, pero no se proyecta porque A,B,C ⊄ EDH

EHF1+ es trivial

DHF1+ = DHB, se deduce DH 🡪 B, pero no se proyecta porque B ⊄ EDH

Luego, las dependencias de F1 que se proyectan sobre EDH son **ΠEDH(F1) = { EH 🡪 D }**

Llaves: EH

**EH 🡪 F**. Izq(EHF) = EFH

EF1+ = EAC, se deduce E 🡪 AC, pero no se proyecta porque A,C ⊄ EFH

FF1+ = FHIJ, se deduce F 🡪 H, que se proyecta sobre EFH **(no será necesario descomponer la df F 🡪 H)**

HF1+ = H no deduce ninguna df no trivial

EHF1+ es trivial

EFF1+ = EFHIJ, se deduce EF 🡪 H, pero no se incorpora por ser redundante con F 🡪 H

FHF1+ = FHIJ, se deduce FH 🡪 IJ, pero no se proyecta porque I,J ⊄ EFH

Luego, las dependencias de F1 que se proyectan sobre EFH son **ΠEFH(F1) = { EH 🡪 F, F 🡪 H }**

Llaves:

EF1+ = E. Como E ⊄ R, no es llave.

EFF1+ = EFH. Como EFH ⊆ R, es llave.

EHF1+ = EHF. Como EHF ⊆ R, es llave.

Llaves: EF, EH

**I 🡪 F y F 🡪 I.** Izq(IF) = FI

FF1+ = FHI, se deduce F 🡪 I, que se proyecta sobre FI

IF1+ es trivial

Luego, las dependencias de F1 que se proyectan sobre FI son **ΠFI(F1) = { I 🡪 F, F 🡪 I }**

Llaves: F, I

**E 🡪 A**. Izq(EA) = EA

AF1+ = A no deduce ninguna df no trivial

EF1+ es trivial

Luego, las dependencias de F1 que se proyectan sobre EA son **ΠEA(F1) = { E 🡪 A }**

Llaves: E

**E 🡪 C**. Izq(EC) = EC

CF1+ = C no deduce ninguna df no trivial

EF1+ es trivial

Luego, las dependencias de F1 que se proyectan sobre EC son **ΠEC(F1) = { E 🡪 C }**

Llaves: E

**F 🡪 J**. Izq(FJ) = FJ

JF1+ = J no deduce ninguna df no trivial

FF1+ es trivial

Luego, las dependencias de F1 que se proyectan sobre FJ son **ΠFJ(F1) = { F 🡪 J }**

Llaves: F

Subesquemas SI:

S1: (ABCD) dependencias ΠABCD(F1) = { ABC 🡪 C, D 🡪 B }, llaves: ABC, ACD

S2: (ABCE) dependencias ΠABCE(F1) = { ABC 🡪 E, E 🡪 AC, AE 🡪 C, CE 🡪 A }, llaves: BE, ABC

S3: (EDH) dependencias ΠEDH(F1) = { EH 🡪 D }, llaves: EH

S4: (EFH) dependencias ΠEFH(F1) = { EH 🡪 F, F 🡪 H }, llaves: EF, EH

S5: (FI) dependencias ΠFI(F1) = { F 🡪 I, I 🡪 F }, llaves: F, I

S6: (AE) dependencias ΠAE(F1) = { E 🡪 A }, llaves: E

S7: (CE) dependencias ΠCE(F1) = { E 🡪 C }, llaves: E

S8: (FJ) dependencias ΠFJ(F1) = { F 🡪 J }, llaves: F

1. J.S.P. Llaves de F: ABCF, ABCH, ABCI, ACDF, ACDH, ACDI, EF, EH, EI

Como la llave EH proveniente del conjunto F es llave en S3, no es necesario agregar un subesquema adicional.

1. Optimización.

S1 U S2: (ABCDE) dependencias ΠABCD(F1) U ΠABCE(F1) =

{ ABC 🡪 CE, D 🡪 B, E 🡪 AC, AE 🡪 C, CE 🡪 A }, llaves: ABC, ACD, BE

S3 U S4: (EDFH) dependencias ΠEDH(F1) U ΠEFH(F1) = { EH 🡪 DF }, llaves: EF, EH

S5 U S8: (FIJ) dependencias ΠFI(F1) U ΠFJ(F1) = { F 🡪 IJ, I 🡪 F }, llaves: F, I

S6 U S7: (ACE) dependencias ΠAE(F1) U ΠAC(F1) = { E 🡪 AC }, llaves: E

1. Subesquemas contenidos dentro de otro:

ACE ⊆ ABCDE: (ABCDE) dependencias ΠABCD(F1) U ΠABCE(F1) U ΠAE(F1) U ΠAC(F1) =

{ ABC 🡪 CE, D 🡪 B, E 🡪 AC, AE 🡪 C, CE 🡪 A, E 🡪 AC }, llaves: ABC, ACD, BE, E

Finalmente se obtiene la siguiente descomposición en 3FN, J.S.P, P.D., optimizada:

***ρ* = (ABCDE, EDFH, FIJ)**

**b)** Encontrar una descomposición en FNBC, j.s.p. , optimizada

Subesquemas Si (calculados en el inciso anterior):

S1: (ABCD) dependencias ΠABCD(F1) = { **ABC** 🡪 C, **D** 🡪 B }, llaves: **ABC**, ACD

S2: (ABCE) dependencias ΠABCE(F1) = { **ABC** 🡪 E, **E** 🡪 AC, **AE** 🡪 C, **CE** 🡪 A }, llaves: BE, **ABC**

S3: (EDH) dependencias ΠEDH(F1) = { **EH** 🡪 D }, llaves: **EH**

S4: (EFH) dependencias ΠEFH(F1) = { **EH** 🡪 F, **F** 🡪 H }, llaves: EF, **EH**

S5: (FI) dependencias ΠFI(F1) = { **F** 🡪 I, **I** 🡪 F }, llaves: **F**, **I**

S6: (AE) dependencias ΠAE(F1) = { **E** 🡪 A }, llaves: **E**

S7: (CE) dependencias ΠCE(F1) = { **E** 🡪 C }, llaves: **E**

S8: (FJ) dependencias ΠFJ(F1) = { **F** 🡪 J }, llaves: **F**

Subesquemas que no respetan FNBC:

* S1 (ABCD), con llaves ABC y ACD, no respeta FNBC porque ΠABCD(F1) contiene la dependencia D 🡪 B, y D no es superllave. Luego, (ABCD) se descompone en dos subesquemas:
  + S1.1: (DB), dependencias: ΠDB(ΠABCD(F1)) = { **D** 🡪 B }, llave: **D**
  + S1.2: (ACD), dependencias: ΠACD(ΠABCD(F1)) = { }, llave: **ACD**
* S2 (ABCE), con llaves BE y ABC, no respeta FNBC porque ΠABCE(F1) contiene la dependencia E 🡪 AC, y E no es superllave. Luego, (ABCE) se descompone en dos subesquemas:
  + S2.1: (ACE), dependencias: ΠACE(ΠABCE(F1)) = { **E** 🡪 AC }, llave: **E**
  + S2.2: (BE), dependencias: ΠBE(ΠABCE(F1)) = { }, llave: **BE**
* S4 (EFH), con llaves EF y EH, no respeta FNBC porque ΠEFH(F1) contiene la dependencia F 🡪 H, y F no es superllave. Luego, (EFH) se descompone en dos subesquemas:
  + S4.1: (FH), dependencias: ΠFH(ΠEFH(F1)) = { **F** 🡪 H }, llave: **F**
  + S4.2: (EF), dependencias: ΠEF(ΠEFH(F1)) = { }, llave: **EF**

Optimización.

* S6, S7 y S2.1 comparten la llave E. Además, S6 y S7 están contenidos dentro de S2.1.

Al unirlos se genera el esquema (ACE), de dependencias ΠACE(F1) = { E 🡪 AC }, llaves: E.

* S5, S8 y S4.1 comparten la llave F.

Al unirlos se genera el esquema (FHIJ), de dependencias ΠFHIJ(F1) = { F 🡪 HI, I 🡪 F }, de dependencias, llaves: F, I.

Finalmente se obtiene la siguiente descomposición en FNBC, J.S.P, optimizada:

***ρ* = (EDH, DB, ACD, BE, EF, ACE, FHIJ)**