Pauta Ayudantía 5.A

IIC2413, Bases de Datos
Benjamín Saavedra

Enunciado

Contexto: Arriendo de bicicletas públicas

Un sistema de bicicletas públicas registra sus arriendos en una única relación:

$$\textbf{PrestamoBici}(\mathsf{bicicleta}, \mathsf{estacion}, \mathsf{comuna}, \mathsf{usuario}, \mathsf{fecha}) \qquad \mathsf{ar} = 5$$

Se sabe que: cada bicicleta tiene una estación base fija; cada estación pertenece a una comuna fija; y, para simplificar, cada usuario arrienda a lo más una bicicleta por día (es decir, usuario y fecha juntos determinan qué bicicleta se usó).

Instancia de ejemplo (opcional):

bicicleta	estacion	comuna	usuario	fecha
B12	E5	Providencia	U1	2026-09-01
B12	E5	Providencia	U2	2026-09-02
B07	E5	Providencia	U1	2026-09-03
B30	E9	Ñuñoa	U3	2026-09-01

Ejercicio A.1.

(a) Escriba las DFs descritas en el enunciado usando la notación formal.

(b) Encuentre la(s) llave(s) de **PrestamoBici** (indique qué atributos deben estar en toda llave y por qué).

(c) ¿Está **PrestamoBici** en BCNF? ¿En 3NF? Indique todas las violaciones encontradas.

(d) Aplique el algoritmo toBCNF completo.

(e) Encuentre una cobertura mínima de las DFs y aplique el Algoritmo de descomposición en 3NF. Compare el resultado con (d): ¿son las mismas relaciones?

(f) En la relación original **PrestamoBici** (sin descomponer), dé un ejemplo concreto de anomalía de **actualización**: ¿qué pasa si la estación E5 cambia de comuna? ¿Se resuelve este problema con su descomposición?

Solución

(a) Traducción del enunciado a DFs.

Leemos cada supuesto del enunciado como una dependencia funcional. La regla general es: la frase "cada A tiene un/una *único/fijo* B" dice exactamente que *conocer el valor de* A *deja determinado el valor de* B, que es la definición misma de $A \to B$.

- "Cada bicicleta tiene una estación base fija" $\Rightarrow$ el valor de estacion está *determinado* por el valor de bicicleta:

$$\mathsf{bicicleta} \to \mathsf{estacion}$$

- "Cada estación pertenece a una comuna fija" $\Rightarrow$

$$\mathsf{estacion} \to \mathsf{comuna}$$

- "Cada usuario arrienda a lo más una bicicleta por día" $\Rightarrow$ conocer el par (usuario, fecha) determina una única bicicleta:

$$\mathsf{usuario\,fecha} \to \mathsf{bicicleta}$$

Por lo tanto:

$$\mathcal{F} = \{\ \mathsf{bicicleta} \to \mathsf{estacion},\quad \mathsf{estacion} \to \mathsf{comuna},\quad \mathsf{usuario\,fecha} \to \mathsf{bicicleta}\ \}.$$

(b) Llaves de PrestamoBici.

El objetivo es encontrar todas las llaves de una relación de 5 atributos. Por definición (recordatorio de la Clase 07), $\bar{A}$ es **llave** de R si y sólo si: (i) $\bar{A}$ es **superllave** de R, y (ii) para todo $\bar{B}$ con $\{\bar{B}\} \subsetneq \{\bar{A}\}$, $\bar{B}$ no es superllave de R (es decir, ningún subconjunto propio ya alcanza). La fuerza bruta sería probar los $2^5 = 32$ subconjuntos de atributos, y para cada uno verificar (i) calculando su clausura y (ii) comparando contra todos sus subconjuntos, pero esto es carísimo incluso para 5 atributos, y crece exponencialmente con el número de atributos.

Nuestra estrategia será: primero acotar drásticamente el espacio de búsqueda *sin* calcular ninguna clausura (Paso 1), luego verificar la condición (i) sobre el candidato acotado (Paso 2), y finalmente verificar la condición (ii) sobre los pocos subconjuntos que quedan como candidatos razonables (Paso 3).

Paso 1: acotar los candidatos.

Antes de calcular ninguna clausura, miramos qué atributos nunca aparecen al lado derecho de ninguna DF de $\mathcal{F}$: eso es una condición necesaria para ser superllave, y es casi gratis de chequear, porque sólo hay que mirar los lados derechos de $\mathcal{F}$, sin calcular ninguna clausura. Para entender *por qué* es necesaria, conviene mirar el mecanismo del Algoritmo 1 y preguntarse: "¿de dónde puede salir un atributo nuevo durante el cálculo de una clausura?"

Repase el cuerpo del `repeat`: *si* $(\bar{B} \to \bar{C}) \in$ *reglas y* $\{\bar{B}\} \subseteq$ *clausura, entonces clausura* $\leftarrow$ *clausura* $\cup\{\bar{C}\}$. La única forma en que un atributo puede *entrar* a `clausura` en cualquier iteración es apareciendo en el lado derecho de alguna regla que se dispara: el algoritmo nunca agrega un atributo "porque sí", sólo atributos que alguna DF de $\mathcal{F}$ promete producir. Si un atributo nunca aparece al lado derecho de ninguna regla, entonces sencillamente no hay ninguna regla que pueda alguna vez agregarlo a una clausura que no lo tuviera ya desde el principio, así que ese atributo tiene que estar presente en cualquier conjunto que aspire a ser superllave (la demostración formal de esto es el Ejercicio B.2).

Miramos qué atributos aparecen al lado derecho de alguna DF de $\mathcal{F}$: aparecen estacion (en la 1ª), comuna (en la 2ª) y bicicleta (en la 3ª). Los atributos usuario y fecha nunca aparecen al lado derecho de ninguna DF. Por lo tanto:

usuario, fecha deben pertenecer a **toda** superllave (y por lo tanto a toda llave).

Esto reduce el problema de "¿cuál de los 32 subconjuntos es llave?" a uno mucho más chico: "partiendo de {usuario, fecha}, ¿alcanza por sí solo, o hay que agregarle algo más?".

Paso 2: calcular $\{\text{usuario}, \text{fecha}\}^+$.

La definición semántica de superllave ("$R \models \bar{A} \to \bar{A}^c$") cuantifica sobre todas las instancias posibles de R que satisfacen $\mathcal{F}$, así que no es algo que podamos "calcular" directamente sin más. La Proposición vista en clases (Clase 08/09) tiende un puente entre esa definición semántica y algo computable:

$$\bar{A} \text{ es superllave de } R \iff \bar{A}^+ = \text{att}(R).$$

Usamos el Algoritmo 1 (clausura), partiendo de reglas $= \mathcal{F}$ y clausura $= \{\text{usuario}, \text{fecha}\}$:

Iteración	Regla que se aplica	Clausura resultante
1	usuario fecha $\to$ bicicleta (LHS $\subseteq$ clausura)	{usuario, fecha, bicicleta}
2	bicicleta $\to$ estacion (LHS $\subseteq$ clausura)	{usuario, fecha, bicicleta, estacion}
3	estacion $\to$ comuna (LHS $\subseteq$ clausura)	{usuario, fecha, bicicleta, estacion, comuna}
4	(ninguna regla queda / ninguna aplica)	*sin cambios: se detiene*

La regla del Algoritmo 1 dice que $\bar{B} \to \bar{C}$ se puede usar exactamente cuando $\{\bar{B}\} \subseteq$ clausura; por eso bicicleta $\to$ estacion recién se vuelve utilizable en la iteración 2, una vez que bicicleta se agregó en la iteración 1. El orden de las iteraciones afecta *cuándo* se agrega cada atributo, pero no el resultado final.

Luego $\{\text{usuario}, \text{fecha}\}^+ = \text{att}(\textbf{PrestamoBici})$ (los 5 atributos), es decir, {usuario, fecha} es **superllave**: se cumple la condición (i) de la definición.

Paso 3: verificar minimalidad.

Falta la condición (ii): hay que revisar cada subconjunto propio de {usuario, fecha} y confirmar que ninguno es superllave.

¿Es superllave algún subconjunto propio de {usuario, fecha}? Los únicos candidatos no vacíos son {usuario} y {fecha}. (No hace falta probar el conjunto vacío: como se demuestra en el Ejercicio B.1, $\varnothing$ nunca es llave salvo en una relación de un solo atributo, y **PrestamoBici** tiene 5.) Calculamos:

$$\{\text{usuario}\}^+ = \{\text{usuario}\} \qquad \{\text{fecha}\}^+ = \{\text{fecha}\}$$

(en ambos casos ninguna regla de $\mathcal{F}$ tiene su lado izquierdo contenido en un conjunto de un solo atributo distinto de sí mismo, así que la clausura no crece). Ninguno de los dos es superllave, por lo tanto {usuario, fecha} es **minimal**.

{usuario, fecha} es una **llave** de **PrestamoBici**.

El Paso 1 no sólo encontró *un* candidato: demostró algo más fuerte, que toda superllave debe contener a {usuario, fecha}. Esto deja sólo dos posibilidades para cualquier otro candidato a llave: o es {usuario, fecha} mismo, o es un superconjunto propio suyo.

La segunda opción queda descartada por **monotonía** de la clausura (si $\{\bar{X}\} \subseteq \{\bar{Y}\}$ entonces $\bar{X}^+ \subseteq \bar{Y}^+$, demostrado en el Ejercicio B.3(a)): como {usuario, fecha} ya es superllave (Paso 2),

todo superconjunto suyo también lo es, y ningún superconjunto propio de una superllave es minimal. Concluimos, sin revisar ningún otro subconjunto, que $\{\mathsf{usuario}, \mathsf{fecha}\}$ es la **única** llave de **PrestamoBici**.

(c) ¿BCNF? ¿3NF?

La definición de 3NF (recordatorio) necesita saber cuáles atributos son **primarios**, es decir, pertenecen a *alguna* llave, y eso exige conocer todas las llaves de la relación, trabajo que ya hicimos en (b). Por eso (c) no se puede resolver antes que (b). BCNF, en cambio, no necesita el concepto de "primario", así que la resolvemos primero, y reutilizamos exactamente el mismo tipo de cálculo (clausuras de lados izquierdos) para ambas preguntas.

Verificación de BCNF.

Por definición (recordatorio), R está en BCNF si y sólo si toda DF no trivial de R tiene lado izquierdo superllave. Ya vimos en (b) que "$\bar{A}$ es superllave" es equivalente, operacionalmente, a "$\bar{A}^+ = \mathsf{att}(R)$". Así que verificar BCNF se reduce, mecánicamente, a repetir el mismo procedimiento del Paso 2 de (b), calculando una clausura y comparando, mirando esta vez el lado izquierdo de cada DF de $\mathcal{F}$, en vez de un único candidato a llave.

- $\mathsf{bicicleta} \to \mathsf{estacion}$: calculamos $\{\mathsf{bicicleta}\}^+$: $\mathsf{bicicleta} \to \mathsf{estacion}$ agrega $\mathsf{estacion}$; luego $\mathsf{estacion} \to \mathsf{comuna}$ agrega comuna; ninguna otra regla aplica. Entonces

  $$\{\mathsf{bicicleta}\}^+ = \{\mathsf{bicicleta}, \mathsf{estacion}, \mathsf{comuna}\} \neq \mathsf{att}(\mathbf{PrestamoBici})$$

  (le faltan $\mathsf{usuario}, \mathsf{fecha}$) $\Rightarrow$ $\{\mathsf{bicicleta}\}$ no es superllave $\Rightarrow$ **violación de BCNF**.

- $\mathsf{estacion} \to \mathsf{comuna}$: $\{\mathsf{estacion}\}^+ = \{\mathsf{estacion}, \mathsf{comuna}\} \neq \mathsf{att}(\mathbf{PrestamoBici}) \Rightarrow \{\mathsf{estacion}\}$ no es superllave $\Rightarrow$ **violación de BCNF**.

- $\mathsf{usuario}\,\mathsf{fecha} \to \mathsf{bicicleta}$: por (b), $\{\mathsf{usuario}, \mathsf{fecha}\}^+ = \mathsf{att}(\mathbf{PrestamoBici}) \Rightarrow$ sí es superllave $\Rightarrow$ sin violación.

El test de verificación de BCNF depende únicamente del lado izquierdo de la DF, no de cuál sea el lado derecho: si $\{\mathsf{bicicleta}\}$ no es superllave, entonces cualquier DF no trivial con ese mismo lado izquierdo, sea $\mathsf{bicicleta} \to \mathsf{estacion}$, $\mathsf{bicicleta} \to \mathsf{comuna}$, o incluso $\mathsf{bicicleta} \to \mathsf{estacion}\,\mathsf{comuna}$, es automáticamente también una violación, por la misma razón (el mismo $\{\mathsf{bicicleta}\}^+$ ya calculado). Por eso alcanza con revisar una vez cada lado izquierdo distinto que aparece en $\mathcal{F}$: ya cubre, de manera implícita, todas las DFs derivadas que comparten ese lado izquierdo (justificación formal: Ejercicio B.3(c)).

PrestamoBici no está en BCNF (2 violaciones).

Verificación de 3NF.

La definición de 3NF "perdona" una violación de BCNF cuando el atributo del lado derecho es primario (pertenece a alguna llave). Esta cláusula extra es exactamente la diferencia entre BCNF y 3NF (recordatorio: BCNF $\Rightarrow$ 3NF, pero no al revés). Por eso, antes de poder aplicar el criterio de 3NF, tenemos que saber *cuáles* atributos son primarios, información que sólo podemos obtener a partir de la lista completa de llaves, calculada en (b).

Como la única llave es $\{\mathsf{usuario}, \mathsf{fecha}\}$, los atributos primarios son exactamente $\mathsf{usuario}$ y fecha. Los atributos $\mathsf{bicicleta}$, $\mathsf{estacion}$ y comuna no son primarios.

Revisamos otra vez las mismas DFs (reutilizando las clausuras ya calculadas arriba), pero ahora con el criterio de 3NF ("el lado izquierdo es superllave, o el atributo del lado derecho es primario"):

- bicicleta → estacion: {bicicleta} no es superllave (visto arriba) y estacion no es primario ⇒ **viola 3NF** también.
- estacion → comuna: {estacion} no es superllave y comuna no es primario ⇒ **viola 3NF** también.
- usuario fecha → bicicleta: el lado izquierdo ya es superllave ⇒ sin violación (ni siquiera necesitamos mirar si bicicleta es primario: la condición de 3NF es un "o", y la primera mitad ya se cumple).

PrestamoBici tampoco está en 3NF: en ambas violaciones el atributo del lado derecho (estacion y comuna) no pertenece a ninguna llave.

Tiene sentido: la única diferencia entre BCNF y 3NF es que 3NF perdona una violación si el atributo del lado derecho es primario. Ni estacion ni comuna pertenecen a la única llave ($\{\mathsf{usuario}, \mathsf{fecha}\}$), así que esa cláusula nunca se activa aquí y ambas violaciones de BCNF persisten en 3NF.

(d) Algoritmo toBCNF.

Ya sabemos (parte c) que **PrestamoBici** no está en BCNF, así que debemos descomponerla. Usamos el algoritmo visto en clases:

Algoritmo de descomposición en BCNF. `input`: una relación R y sus DFs $\mathcal{F}$. `output`: una descomposición $\{(R_1, \mathcal{F}_1), \ldots, (R_k, \mathcal{F}_k)\}$ tal que cada R_i está en BCNF según $\mathcal{F}_i$.

1. **function** `toBCNF`$(R, \mathcal{F})$
2. **if** R está en BCNF **then return** $\{(R, \mathcal{F})\}$
3. sea $\bar{A} \to \bar{B} \in \mathcal{F}$ tal que $\bar{A}$ no es superllave
4. calcule $\bar{A}^+$ usando el **Algoritmo 1**
5. $\bar{A}^- \leftarrow (\mathsf{att}(R) \setminus \bar{A}^+) \cup \{\bar{A}\}$
6. $R^+ \leftarrow \pi_{\bar{A}^+}(R)$; $\quad R^- \leftarrow \pi_{\bar{A}^-}(R)$
7. calcule $\mathcal{F}^+, \mathcal{F}^-$ (dependencias de R^+, R^- según $\mathcal{F}$) usando el **Algoritmo 2**
8. **return** `toBCNF`$(R^+, \mathcal{F}^+) \cup$ `toBCNF`$(R^-, \mathcal{F}^-)$

La partición en R^+ y R^- (líneas 5–6) logra dos cosas a la vez: es **reversible** (sin pérdida de información), porque $\bar{A}$ queda como atributo compartido entre ambas piezas; y R^+ tiene más chances de quedar en BCNF que R, porque por construcción $\bar{A}$ ya es superllave *dentro* de R^+ ($\mathsf{att}(R^+) = \bar{A}^+$). Esto se aplica recursivamente (línea 8) hasta que todas las piezas activan el caso base (línea 2).

Llamada `toBCNF`(PrestamoBici, $\mathcal{F}$). $R :=$ **PrestamoBici** no está en BCNF (parte c), así que no se activa el caso base. Elegimos (línea 3) la violación

$$\bar{A} \to \bar{B} \;:=\; \mathsf{bicicleta} \to \mathsf{estacion} \qquad (\bar{A} = \{\mathsf{bicicleta}\} \text{ no es superllave}).$$

Ya calculamos en (c) (línea 4, con el Algoritmo 1):

$$\bar{A}^+ = \{\mathsf{bicicleta}\}^+ = \{\mathsf{bicicleta}, \mathsf{estacion}, \mathsf{comuna}\}.$$

Agrupamos en una sola relación a $\bar{A}$ junto con todo lo que $\bar{A}$ determina (su clausura completa, no sólo $\bar{B}$). La razón es que, dentro de esta nueva relación R^+, $\bar{A}$ deja de ser un problema: por

construcción, la clausura de $\bar{A}$ *restringida a los atributos de* R^+ es exactamente $\mathsf{att}(R^+)$ (porque $\mathsf{att}(R^+) = \bar{A}^+$), así que $\bar{A}$ será superllave de R^+. En otras palabras: resolvemos la violación original poniendo a $\bar{A}$ y a todo lo que determina en su propia "mesa", donde $\bar{A}$ manda sobre todo lo que hay.

$$R^+ := \pi_{\bar{A}^+}(R) = \pi_{\mathsf{bicicleta,estacion,comuna}}(\textbf{PrestamoBici}) \;=:\; \textbf{Bicicletas}(\mathsf{bicicleta}, \mathsf{estacion}, \mathsf{comuna}).$$

Con el **Algoritmo 2** (línea 7): para cada subconjunto de $\{\mathsf{bicicleta}, \mathsf{estacion}, \mathsf{comuna}\}$ se calcula su clausura con **todo** $\mathcal{F}$ y se guarda lo que aporta dentro de ese mismo conjunto. Aquí $\mathsf{bicicleta} \to \mathsf{estacion}$ y $\mathsf{estacion} \to \mathsf{comuna}$ ya están literalmente en $\mathcal{F}$ con ambos lados dentro de R^+; también se obtendría $\mathsf{bicicleta} \to \mathsf{comuna}$ (derivada de las otras dos), pero es redundante frente a ellas, así que basta con:

$$\mathcal{F}^+ = \{\mathsf{bicicleta} \to \mathsf{estacion},\quad \mathsf{estacion} \to \mathsf{comuna}\}.$$

Los atributos que $\bar{A}$ no determina ($\mathsf{usuario}$, fecha) tienen que quedar en alguna relación, pero si dejáramos sólo esos atributos en R^- (sin incluir a $\bar{A}$ también ahí), R^+ y R^- no tendrían ningún atributo en común, y no habría forma de reconstruir R haciendo $R^+ \bowtie R^-$: se perdería por completo la información de *qué bicicleta* corresponde a *qué préstamo*. Por eso la línea 5 agrega $\bar{A}$ explícitamente también a $\bar{A}^-$: actúa como el atributo "puente" que permite reconectar ambas piezas más adelante.

$$\begin{aligned}\bar{A}^- &:= (\mathsf{att}(R) \setminus \bar{A}^+) \cup \{\bar{A}\}\\ &= (\{\mathsf{bicicleta}, \mathsf{estacion}, \mathsf{comuna}, \mathsf{usuario}, \mathsf{fecha}\} \setminus \{\mathsf{bicicleta}, \mathsf{estacion}, \mathsf{comuna}\}) \cup \{\mathsf{bicicleta}\}\\ &= \{\mathsf{usuario}, \mathsf{fecha}\} \cup \{\mathsf{bicicleta}\} = \{\mathsf{usuario}, \mathsf{fecha}, \mathsf{bicicleta}\}.\end{aligned}$$

$$\begin{aligned}R^- &:= \pi_{\bar{A}^-}(R) = \pi_{\mathsf{usuario,fecha,bicicleta}}(\textbf{PrestamoBici})\\ &=: \textbf{Prestamos}(\mathsf{usuario}, \mathsf{fecha}, \mathsf{bicicleta}).\end{aligned}$$

Con el **Algoritmo 2**: $\mathsf{usuario}\,\mathsf{fecha} \to \mathsf{bicicleta}$ sí tiene todos sus atributos en $\mathsf{att}(\textbf{Prestamos})$ (recuerde, $\{\mathsf{usuario}, \mathsf{fecha}, \mathsf{bicicleta}\}$), así que se proyecta. ¿Se proyecta también $\mathsf{bicicleta} \to \mathsf{usuario}$ o $\mathsf{bicicleta} \to \mathsf{fecha}$? No: ya vimos que $\{\mathsf{bicicleta}\}^+$ es $\{\mathsf{bicicleta}, \mathsf{estacion}, \mathsf{comuna}\}$, que no contiene ni a $\mathsf{usuario}$ ni a fecha. Entonces

$$\mathcal{F}^- = \{\mathsf{usuario}\,\mathsf{fecha} \to \mathsf{bicicleta}\}.$$

La línea 8 hace llamadas recursivas: $\texttt{toBCNF}(\textbf{Bicicletas}, \mathcal{F}^+) \cup \texttt{toBCNF}(\textbf{Prestamos}, \mathcal{F}^-)$.

Llamada $\texttt{toBCNF}(\textbf{Bicicletas}, \mathcal{F}^+)$.

Haber resuelto una violación (la de $\mathsf{bicicleta} \to \mathsf{estacion}$) no garantiza que $R^+ = \textbf{Bicicletas}$ ya esté en BCNF: $\mathcal{F}^+$ puede seguir teniendo otras DFs cuyo lado izquierdo no es superllave dentro de estos 3 atributos, aunque sí lo fuera considerando toda la relación original. Por eso el caso base (línea 2) se revisa de nuevo en cada llamada.

Con $\mathcal{F}^+ = \{\mathsf{bicicleta} \to \mathsf{estacion}, \mathsf{estacion} \to \mathsf{comuna}\}$:

- ¿$\{\mathsf{bicicleta}\}$ superllave de **Bicicletas**? Su clausura *dentro de estos 3 atributos* es todo $\mathsf{att}(\textbf{Bicicletas}) = \{\mathsf{bicicleta}, \mathsf{estacion}, \mathsf{comuna}\}$ ⇒ **sí**; sin violación.
- ¿$\{\mathsf{estacion}\}$ superllave de **Bicicletas**? Su clausura es $\{\mathsf{estacion}, \mathsf{comuna}\}$, que no es todo $\mathsf{att}(\textbf{Bicicletas})$ (falta $\mathsf{bicicleta}$) ⇒ **no** ⇒ violación. No se activa el caso base: hay que seguir descomponiendo.

Elegimos $\bar{A} \to \bar{B} := \mathsf{estacion} \to \mathsf{comuna}$ (dentro de **Bicicletas**). $\bar{A}^+ = \{\mathsf{estacion}\}^+ = \{\mathsf{estacion}, \mathsf{comuna}\}$.

$$R^{++} := \pi_{\mathsf{estacion},\mathsf{comuna}}(\textbf{Bicicletas}) =: \textbf{Estaciones}(\mathsf{estacion}, \mathsf{comuna}),$$
$$\mathcal{F}^{++} = \{\mathsf{estacion} \to \mathsf{comuna}\}.$$

$\bar{A}^- = (\{\mathsf{bicicleta}, \mathsf{estacion}, \mathsf{comuna}\} \setminus \{\mathsf{estacion}, \mathsf{comuna}\}) \cup \{\mathsf{estacion}\} = \{\mathsf{bicicleta}, \mathsf{estacion}\}$,

$R^{+-} := \pi_{\mathsf{bicicleta},\mathsf{estacion}}(\textbf{Bicicletas}) =: \textbf{Bicis}(\mathsf{bicicleta}, \mathsf{estacion})$, $\quad \mathcal{F}^{+-} = \{\mathsf{bicicleta} \to \mathsf{estacion}\}$.

Ambas relaciones tienen exactamente 2 atributos. Como se vio en clases, toda relación binaria está automáticamente en BCNF. Esto activa el caso base (línea 2) de inmediato para ambas, sin repetir el cálculo de clausura: $\texttt{toBCNF}(\textbf{Estaciones}, \mathcal{F}^{++}) = \{(\textbf{Estaciones}, \mathcal{F}^{++})\}$ y $\texttt{toBCNF}(\textbf{Bicis}, \mathcal{F}^{+-}) = \{(\textbf{Bicis}, \mathcal{F}^{+-})\}$.

Llamada $\texttt{toBCNF}(\textbf{Prestamos}, \mathcal{F}^-)$. Con $\mathcal{F}^- = \{\mathsf{usuario\,fecha} \to \mathsf{bicicleta}\}$: la clausura de $\{\mathsf{usuario}, \mathsf{fecha}\}$ *dentro de estos 3 atributos* es todo $\mathsf{att}(\textbf{Prestamos}) = \{\mathsf{usuario}, \mathsf{fecha}, \mathsf{bicicleta}\}$ $\Rightarrow$ es superllave $\Rightarrow$ **Prestamos** ya está en BCNF: se activa el caso base (línea 2) de inmediato, sin descomponer más.

Descomposición BCNF final:

$\{$ **Bicis**$(\mathsf{bicicleta}, \mathsf{estacion})$, **Estaciones**$(\mathsf{estacion}, \mathsf{comuna})$,
Prestamos$(\mathsf{usuario}, \mathsf{fecha}, \mathsf{bicicleta})$ $\}$

No importa, en este caso, que hayamos elegido $\mathsf{bicicleta} \to \mathsf{estacion}$ primero y no $\mathsf{estacion} \to \mathsf{comuna}$: si hubiésemos partido al revés, el resultado final, salvo el orden en que se listan las relaciones, habría sido el mismo. La razón de fondo es que $\bar{A}^+$ siempre se calcula respecto a todo $\mathcal{F}$, y no sólo respecto a la regla puntual que disparó la división: ambas violaciones "ven" el mismo conjunto de reglas disponible, así que terminan reconstruyendo la misma partición por caminos distintos.

(e) Cobertura mínima y descomposición en 3NF.

El algoritmo de descomposición en 3NF (a diferencia de toBCNF) exige explícitamente que su input sea una cobertura mínima. Esto es porque la demostración de que cada relación construida $R_{\bar{A}B}$ termina en 3NF depende de que $\bar{A}$ sea efectivamente una llave (minimal) de $R_{\bar{A}B}$, y eso sólo está garantizado si la regla $\bar{A} \to B$ viene de una cobertura mínima.

Por definición (recordatorio de la Clase 10), un conjunto de DFs $\mathcal{F}$ es **minimal** si se cumplen las tres condiciones siguientes:

(1) Toda dependencia en $\mathcal{F}$ tiene la forma $\bar{A} \to B$, con B un único atributo.

(2) Para toda $\bar{A} \to B \in \mathcal{F}$, se tiene $\mathcal{F} \not\equiv \mathcal{F} \setminus \{\bar{A} \to B\}$ (ninguna regla es redundante).

(3) Para toda $\bar{A} \to B \in \mathcal{F}$ y todo $\bar{A}' \subsetneq \bar{A}$, se tiene $\mathcal{F} \not\equiv (\mathcal{F} \setminus \{\bar{A} \to B\}) \cup \{\bar{A}' \to B\}$ (ningún atributo del lado izquierdo de una regla sobra).

Dicho de otro modo: $\mathcal{F}$ es minimal si toda regla y toda parte izquierda de una regla es necesaria. Si usáramos un $\mathcal{F}$ que no cumpliera esto (lados derechos compuestos, atributos de más a la izquierda, o reglas que sobran), el algoritmo podría producir relaciones que en realidad no están en 3NF, o generar más tablas de las necesarias. La condición (1) se revisa directo (Paso 1); las condiciones

(2) y (3) (Pasos 2 y 3) se apoyan ambas en la misma propiedad de la clausura, que conviene dejar clara antes de aplicarla.

La clausura es **monótona**: más reglas disponibles, o más atributos de partida, nunca la achican, solo la pueden agrandar (o dejarla igual). Esto se ve directo en el mecanismo del `repeat`: una regla de más, o un atributo de partida de más, solo agrega oportunidades para que dispare alguna regla, nunca las quita.

Al revés: quitar una regla, o achicar el lado izquierdo de una regla, nunca agranda la clausura, solo la achica o la deja igual. Entonces, si la clausura *más chica* igual alcanza a B, esa regla o ese atributo no eran necesarios para llegar a B. Las condiciones (2) y (3) preguntan exactamente eso:

- **Condición (2) (Paso 2): ¿sobra la regla $\bar{A} \to B$?**

$$B \in \bar{A}^+ \quad \text{calculado con} \quad \mathcal{F} \setminus \{\bar{A} \to B\} \ ?$$

  Si sí, la regla es redundante: se puede derivar sin ella.

- **Condición (3) (Paso 3): ¿sobra el atributo C en $\bar{A}$?**

$$B \in (\bar{A} \setminus \{C\})^+ \quad \text{calculado con} \quad \mathcal{F} \ ?$$

  Si sí, C sobra: $\bar{A} \setminus \{C\}$ ya alcanza para llegar a B.

En ambos casos basta con una sola clausura (el Algoritmo 1); no hace falta verificar nada más (la monotonía respecto al conjunto de partida se demuestra formalmente en el Ejercicio B.3(a); la monotonía respecto al conjunto de reglas se sigue del mismo argumento de mecanismo).

Paso 1: condición (1), lados derechos con un solo atributo.

El algoritmo de descomposición en 3NF construye una relación por cada regla $\bar{A} \to B$ con B un único atributo: está definido así, literalmente (recordatorio: $R_{\bar{A}B} = \pi_{\bar{A},B}(R)$). Si una DF tuviera un lado derecho compuesto, como $\bar{A} \to B_1 B_2$, no sabríamos aplicarle la regla del algoritmo directamente sin dividirla primero (con la regla de dividir/juntar vista en la Parte B de la ayudantía).

Las tres DFs de $\mathcal{F}$ ya cumplen esto (ninguna tiene más de un atributo a la derecha), así que en este caso concreto no hay nada que dividir.

Paso 2: condición (2), eliminar DFs redundantes.

Para cada DF, calculamos la clausura de su lado izquierdo usando sólo las otras dos DFs:

- $\mathsf{bicicleta} \to \mathsf{estacion}$: usando $\{\mathsf{estacion} \to \mathsf{comuna},\ \mathsf{usuario\,fecha} \to \mathsf{bicicleta}\}$, ninguna regla tiene lado izquierdo $\subseteq \{\mathsf{bicicleta}\}$, así que $\{\mathsf{bicicleta}\}^+ = \{\mathsf{bicicleta}\}$. Como $\mathsf{estacion} \notin \{\mathsf{bicicleta}\}^+$, **no** es redundante.

- $\mathsf{estacion} \to \mathsf{comuna}$: usando $\{\mathsf{bicicleta} \to \mathsf{estacion},\ \mathsf{usuario\,fecha} \to \mathsf{bicicleta}\}$, $\{\mathsf{estacion}\}^+ = \{\mathsf{estacion}\}$. Como $\mathsf{comuna} \notin \{\mathsf{estacion}\}^+$, **no** es redundante.

- $\mathsf{usuario\,fecha} \to \mathsf{bicicleta}$: usando $\{\mathsf{bicicleta} \to \mathsf{estacion},\ \mathsf{estacion} \to \mathsf{comuna}\}$, ninguna regla tiene lado izquierdo $\subseteq \{\mathsf{usuario}, \mathsf{fecha}\}$, así que $\{\mathsf{usuario}, \mathsf{fecha}\}^+ = \{\mathsf{usuario}, \mathsf{fecha}\}$. Como $\mathsf{bicicleta} \notin \{\mathsf{usuario}, \mathsf{fecha}\}^+$, **no** es redundante.

Paso 3: condición (3), quitar atributos que sobran de los lados izquierdos.

La única DF con lado izquierdo compuesto es $\mathsf{usuario\,fecha} \to \mathsf{bicicleta}$:

- ¿Sobra usuario? (¿alcanza con fecha $\to$ bicicleta sola?) Calculamos $\{\text{fecha}\}^+$ usando todo $\mathcal{F}$: ninguna regla tiene su lado izquierdo contenido en $\{\text{fecha}\}$, así que $\{\text{fecha}\}^+ = \{\text{fecha}\}$. Como bicicleta $\notin \{\text{fecha}\}^+$, usuario **no** sobra (no se puede quitar).
- ¿Sobra fecha? Análogamente, $\{\text{usuario}\}^+ = \{\text{usuario}\}$, y bicicleta $\notin \{\text{usuario}\}^+$, así que fecha **tampoco** sobra.

El lado izquierdo usuario fecha ya es mínimal, lo que además confirma, por otro camino, que $\{\text{usuario}, \text{fecha}\}$ es realmente una llave (coherente con lo encontrado en (b)).

$\mathcal{F}$ **ya es una cobertura mínima**:
$\mathcal{F}_{\min} = \mathcal{F} = \{\text{bicicleta} \to \text{estacion},\ \text{estacion} \to \text{comuna},\ \text{usuario fecha} \to \text{bicicleta}\}$.

Algoritmo de descomposición en 3NF.

La idea del algoritmo (a diferencia de toBCNF, que *parte* la relación por sus violaciones) es *construir*, desde cero, una relación por cada "unidad de información" que la cobertura mínima identificó como independiente: cada regla $\bar{A} \to B$ dice "$\bar{A}$ determina a B", así que $\bar{A}$ y B deberían vivir juntos en su propia tabla. Se agrega además una relación con alguna llave de R para garantizar, sin importar qué tan fragmentado quede el resto, que la descomposición completa siga siendo sin pérdida.

Por cada $\bar{A} \to B \in \mathcal{F}_{\min}$ creamos $R_{\bar{A}B} = \pi_{\bar{A},B}(R)$:

$$\text{bicicleta} \to \text{estacion} \ \Rightarrow\ R_1 = \textbf{Bicis}(\text{bicicleta}, \text{estacion})$$

$$\text{estacion} \to \text{comuna} \ \Rightarrow\ R_2 = \textbf{Estaciones}(\text{estacion}, \text{comuna})$$

$$\text{usuario fecha} \to \text{bicicleta} \ \Rightarrow\ R_3 = \textbf{Prestamos}(\text{usuario}, \text{fecha}, \text{bicicleta})$$

y agregamos, para la llave $\bar{K} = \{\text{usuario}, \text{fecha}\}$ encontrada en (b):

$$R_4 = (\text{usuario}, \text{fecha}).$$

Comparación con (d). R_1, R_2, R_3 coinciden exactamente con las de toBCNF. R_4, en cambio, es redundante: $\{\text{usuario}, \text{fecha}\} \subsetneq \text{att}(R_3)$, y proyectar R_3 sobre $\{\text{usuario}, \text{fecha}\}$ da exactamente R_4, así que no aporta nada nuevo y se puede omitir. (El algoritmo la agrega siempre porque, en el caso general, es la única forma de garantizar que la descomposición sea sin pérdida; aquí no hacía falta porque R_3 ya la contenía.)

Descartando la relación redundante R_4, la descomposición 3NF **coincide exactamente** con la descomposición BCNF de (d).

Esto no está garantizado en general (**Showtimes** de la Clase 10 es un caso donde 3NF y BCNF sí difieren): aquí coincide porque la cobertura mínima ya tenía la misma estructura que toBCNF armó a mano en (d), lo cual es una buena señal de que ambos cálculos están correctos.

(f) Anomalía de actualización (en el esquema original, sin descomponer).

En la instancia de ejemplo, la estación E5 aparece en tres filas de **PrestamoBici**, repitiendo cada vez comuna = Providencia: consecuencia directa de que estacion $\to$ comuna viola BCNF (parte c).

bicicleta	estacion	comuna	usuario	fecha
B12	E5	**Providencia**	U1	2026-09-01
B12	E5	**Providencia**	U2	2026-09-02
B07	E5	**Providencia**	U1	2026-09-03

Si E5 cambia de comuna (a "Recoleta"), hay que actualizar las tres filas a la vez; el motor no puede impedir que se actualice sólo una o dos, dejando la relación afirmando simultáneamente que E5 está en Providencia y en Recoleta. Esa es la **anomalía de actualización**.

¿Se resuelve con la descomposición? Sí: en **Estaciones**, estacion es llave, así que cada estación aparece en una sola fila; basta un único `UPDATE` y no hay forma de que la relación quede contradictoria (exactamente lo que dice la Clase 09: BCNF elimina las anomalías de actualización).