|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | 1. **Cubrimiento Mínimo**    1. Obtener un Cubrimiento Cerrado en Atributos    2. Eliminar dependencias funcionales redundantes 2. **Cubrimiento Reducido**    1. Reducir a izquierda: eliminar atributos extraños a izquierda    2. Reducir a derecha: eliminar atributos extraños a derecha |  |
|  |  |  |
|  | *Ejemplo*: Esquema R(ABCDE), conjunto de DF’s F = { A 🡪 C, D 🡪 A, CD 🡪 A, DAE 🡪 B }   * 1. **Cubrimiento cerrado en atributos**: (A)F+ = AC, (D)F+ = DAC, (CD)F+ = CDA, (DAE)F+ = DAEBC   Luego, el cubrimiento cerrado en atributos es M1 = { A 🡪 AC, D 🡪 ACD, CD 🡪 ACD, DAE 🡪 ABCDE }   * 1. **Eliminar dependencias funcionales redundantes**. Tomo una dependencia y la tacho del conjunto Mi. Luego, calculo la clausura del atributo a izquierda de esa dependencia con respecto al conjunto Mi restante:   (A)M1 \ (A 🡪 AC)+ = A ⊄ AC. Luego, A 🡪 AC no es redundante.  (D)M1 \ (D 🡪 ACD)+ = D ⊄ ACD. Luego, D 🡪 ACD no es redundante.  (CD)M1 \ (CD 🡪 ACD)+ = CDA ⊆ ACD. Luego, CD 🡪 ACD es redundante.  (DAE)M1 \ (DAE 🡪 ABCDE)+ = DAEC ⊄ ABCDE. Luego, DAE 🡪 ABCDE no es redundante.  Luego, el cubrimiento mínimo es M2 = { A 🡪 AC, D 🡪 ACD, DAE 🡪 ABCDE }   * 1. **Reducir a izquierda**: eliminar atributos extraños a izquierda en M2.   A 🡪 AC y D 🡪 ACD no tienen atributos extraños a izquierda dado que tienen un único atributo a izquierda.  Para DAE 🡪 ABCDE calculamos la clausura del lado izquierdo respecto a M2 quitando un atributo a la vez:  (AE)M2+ = AEC ⊄ ABCDE. Luego D no es extraño a izquierda.  (AD)M2+ = ADC ⊄ ABCDE. Luego E no es extraño a izquierda.  (DE)M2+ = DEACB ⊆ ABCDE. Luego A es extraño a izquierda.  Luego, el cubrimiento mínimo reducido a izquierda es M3 = { A 🡪 AC, D 🡪 ACD, DE 🡪 ABCDE }  2.2 **Reducir a derecha**: eliminar atributos extraños a derecha en M3.  Primero, remuevo los atributos trivialmente extraños a derecha. Es decir, A 🡪 AC ahora será A 🡪 C (que no tiene atributos extraños a derecha porque la dependencia consta de un único atributo a derecha), D 🡪 ACD será D 🡪 AC, etc. Luego, M4 = { A 🡪 C, D 🡪 AC, DE 🡪 ABC }.  Luego, tomo las dependencias con más de un atributo a derecha y calculo la clausura del lado izquierdo de la siguiente manera: reemplazo la dependencia analizada en Mi por la dependencia sin el atributo a derecha estudiado.  Para D 🡪 AC, tenemos:  (D)M4 \ { D 🡪 AC } U { D 🡪 C }+ = DC ⊄ AC. Luego, A no es extraño a derecha.  (D)M4 \ { D 🡪 AC } U { D 🡪 A }+ = DAC ⊆ AC. Luego, C es extraño a derecha.  Luego, M5 = { A 🡪 C, D 🡪 A, DE 🡪 ABC }.  Para DE 🡪 ABC, tenemos:  (DE)M3\* \ { DE 🡪 ABC } U { D 🡪 BC }+ = DEABC ⊆ ABC. Luego, A es extraño a derecha.  Luego, M6 = { A 🡪 C, D 🡪 A, DE 🡪 BC }.  (DE)M3\* \ { DE 🡪 BC } U { D 🡪 C }+ = DEAC ⊄ BC. Luego, B no es extraño a derecha.  (DE)M3\* \ { DE 🡪 BC } U { D 🡪 B }+ = DEBCA ⊆ BC. Luego, C es extraño a derecha.  Luego, M7 = { A 🡪 C, D 🡪 A, DE 🡪 B }.  Finalmente, el cubrimiento mínimo reducido de F es **M7 = { A 🡪 C, D 🡪 A, DE 🡪 B }**. |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | **CÁLCULO DE LLAVES**  *EJEMPLO*: Sea F = { KMS 🡪 TN, LP 🡪 MTS, LT 🡪 K, LN 🡪 S, MT 🡪 L, S 🡪 L, KT 🡪 L } un conjunto mínimo reducido definido sobre el esquema R(KLMNPSTU). Para encontrar todas las llaves candidatas distinguimos tres conjuntos de atributos del esquema R(KLMNPSTU):   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Siempre** | **Tal vez** | **Nunca** |  |  | ***Nota***: si el conjunto de atributos  de la columna ‘siempre’ constituye una llave,  entonces podemos asegurar que es la única,  ya que cualquier atributo que incorporemos  a este conjunto generaría una superllave. | | Atributos que  aparecen solo  a izquierda  o no aparecen  en ningún lugar | Atributos que  aparecen  a izquierda  y a derecha | Atributos que  solo aparecen  a derecha |  | | PU | KLMNST |  |  |   Atributos en ‘siempre’:  PUF+ = PU. Como PU ⊄ R, no es llave.  3 atributos:  PUKF+ = PKU. Como PKU ⊄ R, no es llave.  **PULF**+ = PULMTSKN. Como KLMNPSTU ⊆ R, es llave.  PUMF+ = PMU. Como PMU ⊄ R, no es llave.  PUNF+ = PNU. Como PNU ⊄ R, no es llave.  **PUSF**+ = PUSLMTKN. Como KLMNPSTU ⊆ R, es llave.  PUTF+ = PUT. Como PUT ⊄ R, no es llave.  4 atributos. Combinaciones de 3 atributos que no son llave: PUK, PUM, PUN, PUT.  PUKMF+ = PUKM. Como PKMU ⊄ R, no es llave.  PUKNF+ = PUKN. Como PKNU ⊄ R, no es llave.  **PUKTF**+ = PUKTLMSN. Como KLMNPSTU ⊆ R, es llave.  PUMNF+ = PUMN. Como PMNU ⊄ R, no es llave.  **PUMTF**+ = PUMTLKSN. Como KLMNPSTU ⊆ R, es llave.  PUNTF+ = PUNT. Como PMNU ⊄ R, no es llave.  5 atributos. Combinaciones de 4 atributos que no son llave: PUKM, PUKN, PUMN, PUNT.  PUKMNF+ = PUKMN. Como PKMNU ⊄ R, no es llave.  PUKMT, PUKNT y PUMNT no se tienen en cuenta ya que son superllaves, es decir, contienen las llaves PUKT y PUMT previamente calculadas.  6 atributos. Combinaciones de 5 atributos que no son llave. En este caso, cualquier atributo que se le añada a PUKMN nos llevará a una superllave, con lo cual, terminamos la iteración.  Luego, las llaves candidatas son: **PUL, PUS, PUKT, PUMT**. |  |
|  |  |  |

|  |  |
| --- | --- |
| **3FN, J.S.P., P.D., optimizada** | |
|  | |
| *EJEMPLO:*  **R(KLMNPSTU),**  **F = { KMS 🡪 TN, LP 🡪 MTS, LT 🡪 K, MT 🡪 L, LN 🡪 S, S 🡪 L, KT 🡪 L },**  **llaves: PUL, PUS, PUKT, PUMT** | |
|  |  |
| **1.** | Abrir df’s a derecha. Es decir, separo aquellas dependencias con más de un atributo del lado derecho.  F = { KMS 🡪 T, KMS 🡪 N, LP 🡪 M, LP 🡪 T, LP 🡪 S, LT 🡪 K, MT 🡪 L, LN 🡪 S, S 🡪 L, KT 🡪 L } |
|  |  |
| **2.** | Verificar si ya está en 3FN, en cuyo caso, no hay que hacer nada. En este caso, KMS 🡪 N no verifica 3FN porque KMS no es superllave (es decir, ninguna de las llaves halladas forma parte de ella) y porque N no es primo (es decir, no es un atributo de ninguna de las llaves halladas). Para verificar que vale 3FN, cada df debe satisfacer al menos una de tales condiciones. |
|  |  |
| **3.** | Armar un subesquema por cada df, calculando todas las dependencias que se proyectan en cada uno. |
| KSM 🡪 T | Izq(KMST) = KMST  KF+ = K, no deduce ninguna df no trivial  MF+ = M, no deduce ninguna df no trivial  SF+ = SL, deduce S 🡪 L, pero no se proyecta porque L ∉ KMST  TF+ = T, no deduce ninguna df no trivial  KMF+ = KM, no deduce ninguna df no trivial  KSF+ = KSL, deduce KS 🡪 L, pero no se proyecta porque L ∉ KMST  KTF+ = KTL, deduce KT 🡪 L, pero no se proyecta porque L ∉ KMST  MSF+ = MSL, deduce MS 🡪 L, pero no se proyecta porque L ∉ KMST  MTF+ = MTLK, deduce MT 🡪 K, que se proyecta sobre KMST  STF+ = STLK, deduce ST 🡪 K, que se proyecta sobre KMST  KMSF+ es trivial  KMTF+ = KMTL, deduce KMT 🡪 L, pero no se proyecta porque L ∉ KMST  KSTF+ = KSTL, deduce KST 🡪 L, pero no se proyecta porque L ∉ KMST  MSTF+ = MSTLKN, deduce MST 🡪 K, pero no se incorpora por ser redundante con ST 🡪 K  Luego, las dependencias de F proyectadas en KMST son:  **ΠKMST(F) = { KMS 🡪 T, MT 🡪 K, ST 🡪 K }** |
|  | |
| KMS 🡪 N | KF+ = K, no deduce ninguna df no trivial  MF+ = M, no deduce ninguna df no trivial  SF+ = SL, deduce S 🡪 L, pero no se proyecta porque L ∉ KMSN  NF+ = N, no deduce ninguna df no trivial  KMF+ = KM, no deduce ninguna df no trivial  KSF+ = KSL, deduce KS 🡪 L, pero no se proyecta porque L ∉ KMSN  KNF+ = KNL, no deduce ninguna df no trivial  MSF+ = MSL, deduce MS 🡪 L, pero no se proyecta porque L ∉ KMSN  MNF+ = MN, no deduce ninguna df no trivial  SNF+ = SNL, deduce SN 🡪 L, pero no se proyecta porque L ∉ KMSN  KMSF+ es trivial  KMNF+ = KMN, no deduce ninguna df no trivial  KSNF+ = KSNL, deduce KSN 🡪 L, pero no se proyecta porque L ∉ KMSN  MSNF+ = MSNL, deduce MSN 🡪 L, pero no se proyecta porque L ∉ KMSN  Luego, las dependencias de F proyectadas en KMSN son: **ΠKMSN(F) = { KMS 🡪 N }** |
|  | |
| LP 🡪 M | LF+ = L, PF+ = P y MF+ = M no deducen ninguna df no trivial.  LPF+ = LP es trivial  LMF+ = LM, no deduce ninguna df no trivial  Luego, las dependencias de F proyectadas en LPM son: **ΠLPM(F) = { LP 🡪 M }** |
|  | |
| LP 🡪 T | LF+ = L, PF+ = P y TF+ = T no deducen ninguna df no trivial  LPF+ es trivial  LTF+ = LTK, deduce LT 🡪 K, pero no se proyecta porque K ∉ LPT  Luego, las dependencias de F proyectadas en LPT son: **ΠLPT(F) = { LP 🡪 T }** |
|  | |
| LP 🡪 S | LF+ = L y PF+ = P no deducen ninguna df no trivial  SF+ = SL, deduce S 🡪 L, que se proyecta sobre LPS  LPF+ = LP es trivial  LSF+ = LS, no deducen ninguna df no trivial  Luego, las dependencias de F proyectadas en LPS son: **ΠLPS(F) = { S 🡪 L, LP 🡪 S }** |
|  | |
| LT 🡪 K  y  KT 🡪 L | LF+ = L, TF+ = T y KF+ = K no deducen ninguna df no trivial  LTF+ = LT es trivial  LKF+ = LK, no deduce ninguna df no trivial  Luego, las dependencias de F proyectadas en LTK son: **ΠLTK(F) = { LT 🡪 K, KT 🡪 L }** |
|  | |
| MT 🡪 L | MF+ = M, TF+ = T y LF+ = L no deducen ninguna df no trivial  MTF+ = MT es trivial  MLF+ = ML, no deducen ninguna df no trivial  Luego, las dependencias de F proyectadas en MTL son: **ΠMTL(F) = { MT 🡪 L }** |
|  | |
| LN 🡪 S | LF+ = L y NF+ = N, no deducen ninguna df trivial  SF+ = S, deduce S 🡪 L, que se proyecta sobre LNS  LNF+ = LN es trivial  LSF+ = LS, no deducen ninguna df trivial  Luego, las dependencias de F proyectadas en LNS son: **ΠLNS(F) = { LN 🡪 S, S 🡪 L }** |
|  | |
| S 🡪 L | Izq(SL) = LS  LF+ = L no deduce ninguna df no trivial  SF+ es trivial  Luego, las dependencias de F proyectadas en LS son: **ΠLS(F) = { S 🡪 L }** |
|  | |
| Descomposición en subesquemas Si:   * S1: (KMST), dependencias: ΠKMST(F) = { KMS 🡪 T, MT 🡪 K, ST 🡪 K }, llaves: KMS, MST * S2: (KMNS), dependencias: ΠKMNS(F) = { KMS 🡪 N }, llaves: KMS * S3: (LMP), dependencias: ΠLMP(F) = { LP 🡪 M }, llaves: LP * S4: (LPT), dependencias: ΠLPT(F) = { LP 🡪 T }, llaves: LP * S5: (LPS), dependencias: ΠLPS(F) = { LP 🡪 S, S 🡪 L }, llaves: LP, PS * S6: (KLT), dependencias: ΠKLT(F) = { KT 🡪 L, LT 🡪 K }, llaves: KT, LT * S7: (LMT), dependencias: ΠLMT(F) = { MT 🡪 L }, llaves: MT * S8: (LNS), dependencias: ΠLNS(F) = { S 🡪 L, LN 🡪 S }, llaves: LN, NS | |
|  |  |
| **4.** | J.S.P (Join sin pérdida). Dado que ningún subesquema contiene una llave de R, se agrega un subesquema adicional formado por los atributos de alguna llave a elección.   * S9: (PUS), dependencias: ΠPUS(F) = { }, llaves: PUS   Nota: sobre el esquema PUS no se proyectará ninguna df dado que PUS es una llave y por lo tanto los atributos que la conforman no se pueden determinar entre sí. |
|  | |
| **5.** | Optimización. Se unen los esquemas que comparten al menos una llave:   * S1 U S2: (KMNST), dependencias: ΠKMST(F) U ΠKMNS(F) =   { KMS 🡪 T, MT 🡪 K, ST 🡪 K, KMS 🡪 N }, llaves: KMS, MST   * S3 U S4 U S5: (LMPST), dependencias: ΠLMP(F) U ΠLPT(F) U ΠLPS(F) = { LP 🡪 MST, S 🡪 L },   llaves: LP, PS   * S6: (KLT), dependencias: ΠKLT(F) = { KT 🡪 L, LT 🡪 K }, llaves: KT, LT * S7: (LMT), dependencias: ΠLMT(F) = { MT 🡪 L }, llaves: MT * S8: (LNS), dependencias: ΠLNS(F) = { S 🡪 L, LN 🡪 S }, llaves: LN, NS * S9: (PUS), dependencias: ΠPUS(F) = { }, llaves: PUS |
|  |  |
| **6.** | Unir los subesquemas que estén contenidos uno dentro del otro:  (LMT) ⊆ (LMPST), por lo tanto se unen para formar el subesquema (LMPST)  con dependencias { LP 🡪 MST, S 🡪 L, MT 🡪 L } y llaves LP, PS, MPT, tal que:   * S1 U S2: (KMNST), dependencias: ΠKMST(F) U ΠKMNS(F) =   { KMS 🡪 T, MT 🡪 K, ST 🡪 K, KMS 🡪 N }, llaves: KMS, MST   * S3 U S4 U S5 U S7: (LMPST), dependencias: ΠLMP(F) U ΠLPT(F) U ΠLPS(F) U ΠLMT(F) =   { LP 🡪 MST, S 🡪 L, MT 🡪 L }, llaves: LP, PS, MT   * S6: (KLT), dependencias: ΠKLT(F) = { KT 🡪 L, LT 🡪 K }, llaves: KT, LT * S8: (LNS), dependencias: ΠLNS(F) = { S 🡪 L, LN 🡪 S }, llaves: LN, NS * S9: (PUS), dependencias: ΠPUS(F) = { }, llaves: PUS |
|  |  |
| Finalmente se obtiene la siguiente descomposición en 3FN, J.S.P, P.D., optimizada:  ***ρ* = (KMNST, KLT, LMPST, LNS, PUS)** | |

|  |  |
| --- | --- |
| **FNBC, J.S.P. P.D., optimizada**  Descomposición en FNBC a partir de 3FN, j.s.p, p.d. de un esquema R con un conjunto de df F. | |
|  | |
| *EJEMPLO:*  **F = { KMS 🡪 TN, LP 🡪 MTS, LT 🡪 K, MT 🡪 L, LN 🡪 S, S 🡪 L, KT 🡪 L },**  **esquema R(KLMNPSTU), llaves: PUL, PUS, PUKT, PUMT** | |
|  |  |
| **1.** | Calculo una descomposición 3FN, p.d., j.s.p. (pasos 1, 2, 3 y 4 del ejemplo anterior):   * S1: (KMST), dependencias: ΠKMST(F) = { **KMS** 🡪 T, **MT** 🡪 K, **ST** 🡪 K }, llaves: **KMS**, MST * S2: (KMNS), dependencias: ΠKMNS(F) = { **KMS** 🡪 N }, llaves: **KMS** * S3: (LMP), dependencias: ΠLMP(F) = { **LP** 🡪 M }, llaves: **LP** * S4: (LPT), dependencias: ΠLPT(F) = { **LP** 🡪 T }, llaves: **LP** * S5: (LPS), dependencias: ΠLPS(F) = { **LP** 🡪 S, **S** 🡪 L }, llaves: **LP**, PS * S6: (KLT), dependencias: ΠKLT(F) = { **KT** 🡪 L, **LT** 🡪 K }, llaves: **KT**, **LT** * S7: (LMT), dependencias: ΠLMT(F) = { **MT** 🡪 L }, llaves: **MT** * S8: (LNS), dependencias: ΠLNS(F) = { **S** 🡪 L, **LN** 🡪 S }, llaves: **LN**, NS * S9: (PUS), dependencias: ΠPUS(F) = { }, llaves: **PUS** |
|  |  |
| **2.** | Descomposición de subesquemas que no respetan FNBC   * S1 (KMST), con llaves KMS y MST, no respeta FNBC porque ΠKMST(F)   contiene la dependencia ST 🡪 K y ST no es superllave.  Luego, (KMST) se descompone en dos subesquemas:   * + S1.1: (KST), dependencias: ΠKST(ΠKMST(F)) = { **ST** 🡪 K }, llaves: **ST**   + S1.2: (MST), dependencias: ΠMST(ΠKMST(F)) = { }, llaves: MST * S5 (LPS), con llaves LP y PS, no respeta FNBC porque ΠLPS(F)   contiene la dependencia S 🡪 L y S no es superllave.  Luego, (LPS) se descompone en dos subesquemas:   * + S5.1: (SL), dependencias: ΠSL(ΠLPS(F)) = { **S** 🡪 L }, llaves: **S**   + S5.2: (PS), dependencias: ΠPS(ΠLPS(F)) = { }, llaves: PS * S8 (LNS), con llaves LN y NS, no respeta FNBC porque ΠLNS(F)   contiene la dependencia S 🡪 L y S no es superllave.  Luego, (LNS) se descompone en dos subesquemas:   * + S8.1: (SL), dependencias: ΠSL(ΠLNS(F)) = { **S** 🡪 L }, llaves: **S**   + S8.1: (NS), dependencias: ΠNS(ΠLNS(F)) = { }, llaves: NS * Los subesquemas 2, 3, 4, 6, 7 y 9 respetan FNBC y no hay que descomponerlos. |
|  | |
| **3.** | Optimización.   * S3 (LMP) y S4 (LPT) comparten la llave LP. Al unirlos, se genera el subesquema (LMPT) que contiene a S7 (LMT), quedando: (LMPT), dependencias ΠLMPT(F) = { LP 🡪 MT, MT 🡪 L }, llaves: LP, MPT. La dependencia MT 🡪 L viola FNBC porque MT no es superllave, por lo tanto no se puede hacer la optimización. * S8.1 (LS) y S5.1 (LS) comparten llave y están contenidos uno dentro del otro (son iguales), por lo tanto quedan reunidos en uno solo. * S8.2 (NS) está contenido dentro de S2 (KMNS), por lo tanto quedan reunidos en un solo esquema (KMNS) con dependencias: { KMS 🡪 N } y llave KMS. * S5.2 (PS) está contenido dentro de S9 (PUS), por lo tanto quedan reunidos en un solo esquema (PUS) sin dependencias y llave PUS. |
|  | |
| Finalmente se obtiene la siguiente descomposición en FNBC, J.S.P, optimizada:  ***ρ* = (KLT, KST, MST, MSN, KMSN, LS, LMT, LMP, LPT, PUS)** | |