Sea F = {KM S → T N, LP → M T S, LT → K, LN → S, M T → L, S → L, KT → L} un conjunto mínimo reducido y sea KM ST un conjunto de atributos. Sean las llaves: P U L, P U S, P U KT, P U M T

**3era forma normal**

**1.Abrir a derecha las df,** o sea, las dependencias que tienen mas de un atrib a la derecha las separamos en dependencias de 1 atrib a la der.

Ejemplo: KMS->TN va a pasar a ser KMS->T, KMS->N

F quedaría asi: F = {KM S → T, KM S → N, LP → M, LP → T, LP → S, LT → K, LN → S, M T → L,   
S → L, KT → L}

**2.Verificamos si R esta en 3era forma normal** (3FN), o sea, si en cada dependencia funcional el **antecedente es super llave** **y** el **consecuente** **es** **parte** de **alguna** **llave** (es primo).

**Ejemplo 1**:

KM S → T

KMS no es super llave porque no hay ninguna llave que sea KM o KS o MS

T es parte de la llave PUKT

Por lo tanto, KMS -> T está en 3FN

**Ejemplo 2**:

KM S → N

KMS no es super llave porque no hay ninguna llave que sea KM o KS o MS

N no es parte de ninguna llave (no es primo)

Por lo tanto, gracias a KM S → N, F no está en 3FN.

**3.** Si F no esta en 3FN entonces **hay** **que** **formar** **subesquemas** por **cada** **dependencia** **funcional**, **calculando** todas las df que se proyectan en cada uno.

**Agarramos la dependencia del nuevo subesquema y vemos si todas las letras aparecen a la izquierda en el conjunto F**

Recomendación: Comenzar con dependencias que mayor atrib tienen.

Para calcular la proyección de ΠKMST(F):

Creamos un subesquema para la df KM S → T:  
Subesquema 1: KMST

Lo siguiente es como calcular las **llaves** KM S → T en F:

K+F = K no se deduce ninguna df no trivial

M+F=M no se deduce ninguna df no trivial

S+F=SL -> podemos deducir S->L, pero esta dependencia no se va a proyecta porque L ∉ KMST

T+F=T no se deduce ninguna df no trivial

Combinaciones de 2 atributos:

KM +F = KM ⇒ no se deduce ninguna d.f. no trivial

KS+F = KSL ⇒ se deduce KS → L, pero no se proyecta porque L ∉ KM ST

KT +F = KT L ⇒ se deduce KT → L, pero no se proyecta porque L ∉ KM ST

MS+F = M SL ⇒ se deduce M S → L, pero no se proyecta porque L ∉ KM ST

MT +F = M T L**K** ⇒ **se deduce que M T →** **K** **se proyecta sobre KMST**, por lo tanto, lo incorporamos al subesquema 1 = {KM S → T, M T → K}

ST +F = ST LK ⇒ **se deduce que ST → K** **se proyecta sobre KMST**, por lo tanto, la incorporamos al subesquema 1 C: C = **{KM S → T, M T → K, ST → K}**

Combinaciones de 3 atributos:

KMS+F = KMS ⇒ se deduce ST ->K que ya está en ΠKMST(F)

KM T +F = KM T L ⇒ se deduce KM T → L, pero no se proyecta porque L ∉ KM ST

KST +F = KST L ⇒ se deduce KST → L, pero no se proyecta porque L ∉ KM ST

MST +F = M ST LKN ⇒ **se deduce MST → K, pero no se incorpora a C por ser** **redundante**

en C **ya que** K ∈ MST +C = MSTK (note que **ST → K ∈ C**)

Combinaciones de 4 atrib no tiene sentido porque va a dar todo trivial.

Luego las df y las llaves del subesquema 1 son:

(KMST) {KMS->T, MT->K, ST->K}

Calculamos las llaves:

KMS+F = KMST por lo tanto KMS es llave

MT+F = MTK no es llave

ST+F = STK

Luego combinaciones de 2 atrib

MTS+F = MTSK es llave

Las llaves son: KMS, MST

Subesquema 1: (KMST) ΠKMST (F) = {KM S → T, M T → K, ST → K} llaves : KMS, MST

**Esto se repite para todos los subesquemas**

Subesquema en el que pasa algo importante:

Subesquema 5: (LPS) {LP->S}

Para calcular la proyección de ΠLPS(F):

L+F = L

P+F = P

**IMPORTANTE:**

S+F = SNL ⇒ **se deduce S → L**, L pertenece LPS por lo tanto C: C={LP->S, S->L}

**Notemos que en F hay una dependencia que es tal cual es S->L por lo tanto no es necesario hacer subesquema para esa df.**

Combinaciones de 2:

LP+F = LPTSN… ⇒ se deduce LP → S que ya esta

LS+F = …

Subesquema 5: (LPS) {LP->S, S->L} sus llaves son: LP, LS

**4.Join sin perdida**: **Si** **ningún** **subesquema** **contiene** una **llave** de **F**, **entonces** se **agrega** un **subesquema** formado por los atrib de alguna **llave** de **F**:

Nuevo subesquema: **(PUS) {} llaves: PUS**

**5.Optimizacion**: Unimos subesquemas que comparten al menos una llave:

Ejemplo:

Subesquema 1: (KMST) {KM S → T, M T → K, ST → K} llaves: **KMS**, MST

Subesquema 2: (KMNS) {KMS->N} llaves: **KMS**

Como tienen llave en común unimos ambos esquemas:

Subesquema1+2: (KMNST) {**KMS->NT**, M T → K, ST → K} llaves: KMS, MST

(Aclaración: unimos KMS->T y KMS->N en KMS->NT)

F = {KM S → T, KM S → N, LP → M, LP → T, LP → S, LT → K, LN → S, M T → L,   
S → L, KT → L}

**6. Unir subesquemas que esten contenidos en otros**, en el ejemplo de la teoría tenemos:

Subesquema 7: (LMT) {MT->L} llaves: MT

Subesquema de unión hecha en la etapa anterior: (LMPST) {LP->MST, S->L} llaves: LP, LS

Como el subesquema 7 esta incluido en el subesquema de abajo los unimos:

Subesquema de la unión de ambos: (LPMST) {LP->MST, S->L, MT->L}

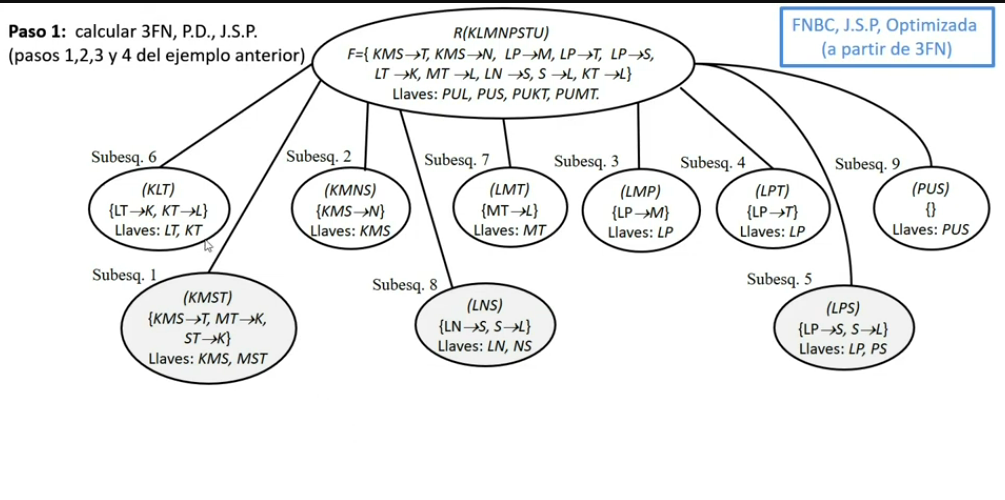
Luego la descomposición es ρ = (KMNST, KLT, LM PST, LNS, PUS) en 3FN, p.d., j.s.p., optimizada

**Algoritmo 3FN:**

1. Abrir a derecha las df y poner al principio del conj las df con más atributos a izquierda
2. Verificamos si F está en 3FN, o sea, si c/df tiene al antecedente como super llave y al consecuente como parte de alguna llave
3. Si F no está en 3FN entonces hay que formar subesquemas por cada dependencia funcional como “calculando llaves” y luego calcular llaves
4. Si ningún subesquema contiene una llave de F, entonces se agrega un subesquema formado por los atrib de alguna llave de F
5. Unimos subesquemas que comparten al menos una llave
6. Unir subesquemas que estén contenidos en otros

**FNBC**

Nos paramos en el punto 4 del ejemplo anterior y vemos que subesquemas ya están en FNBC



1.Tenemos que ver para cada subesquema que las dependencias respeten la FNBC, para esto el lado izquierdo de la dependencia tiene que ser super llave.

Ejemplo: Como se ve en el subesquema 6 LT y KT son super llaves entonces ya está en FNBC. Ídem para los subesquemas que están pintados de gris.

Ejemplo 2: Los subesquemas pintados de gris no respetan la FNBC.

2. Descomponemos los subesquemas que no respetan la FNBC con el algoritmo burbuja.

**Algoritmo burbuja**: Elegimos una df de una subesquema que no respeta FNBC y armamos un subesquema nuevo, luego teniendo en cuenta el lado derecho de la dependencia del nuevo subesquema, lo quitamos del esquema original.

Ejemplo: El esquema 1: (KMS) {KMS->T, **MT->K, ST->K**} llaves: KMS

Elegimos la df que tengan la menor cantidad de atrib posibles y que el lado derecho aparezca menos en df, en este caso son lo mismo:

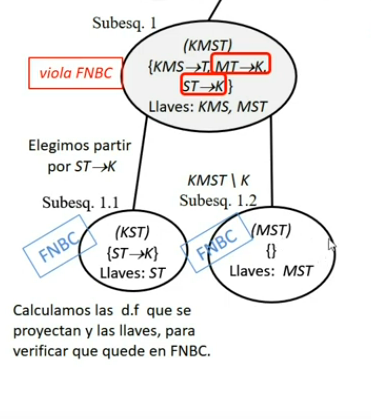
Elegimos: ST->K y armamos el subesquema (KST) {ST->K} calculamos las llaves y estas tienen que respetar la forma FNBC, en este caso las llaves serian: ST.

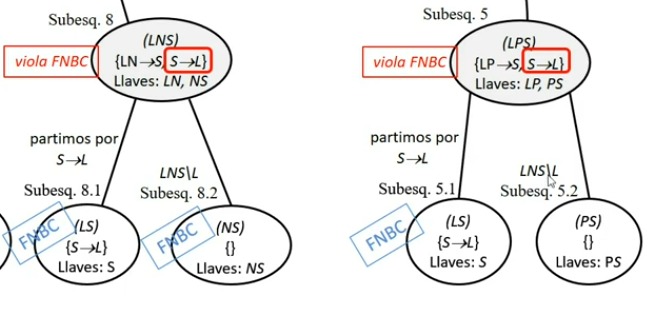
**Importante si el subesquema que elegimos no queda en FNBC entonces tenemos que agarrar otra dependencia y hacer el subesquema de nuevo**

Luego teniendo en cuenta el lado derecho de la dependencia del nuevo subesquema, lo quitamos del esquema original:  
Esquema nuevo: (MST) {calculamos sus df} y calculamos sus llaves: MST.

Si el subesquema derecho no cumple con la FNBC entonces hay que repetir el algoritmo.

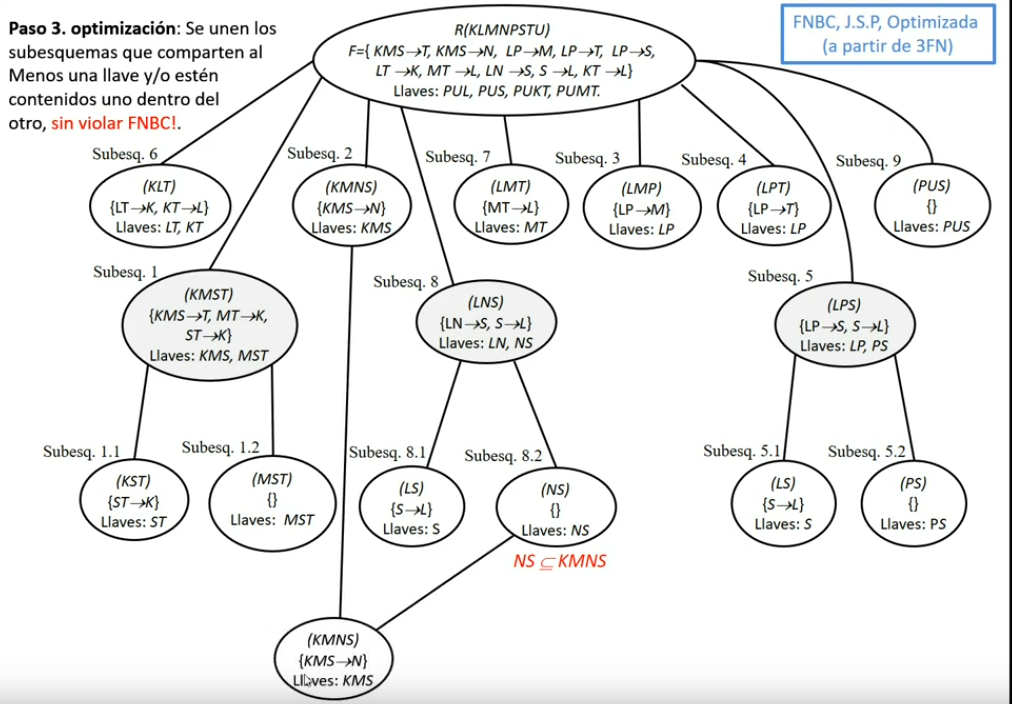
Visualmente:



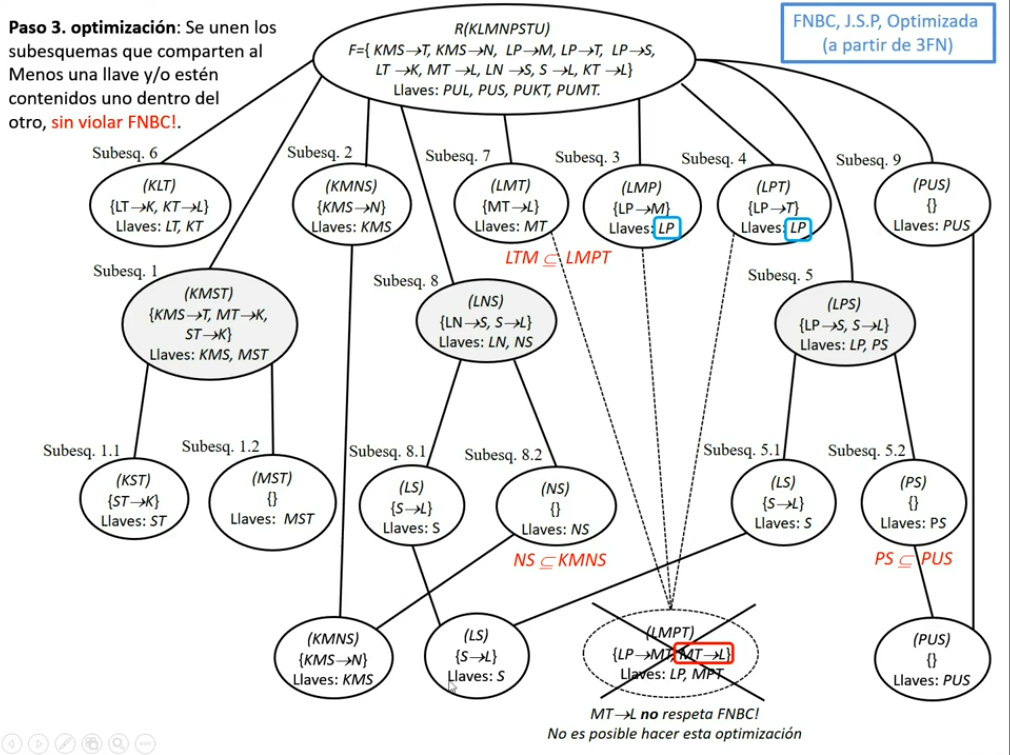
Esto se repite para los otros casos:  


3. Optimización: Se unen subesquemas que compartan al menos una llave o esten contenidos uno dentro del otro, sin violar FNBC.

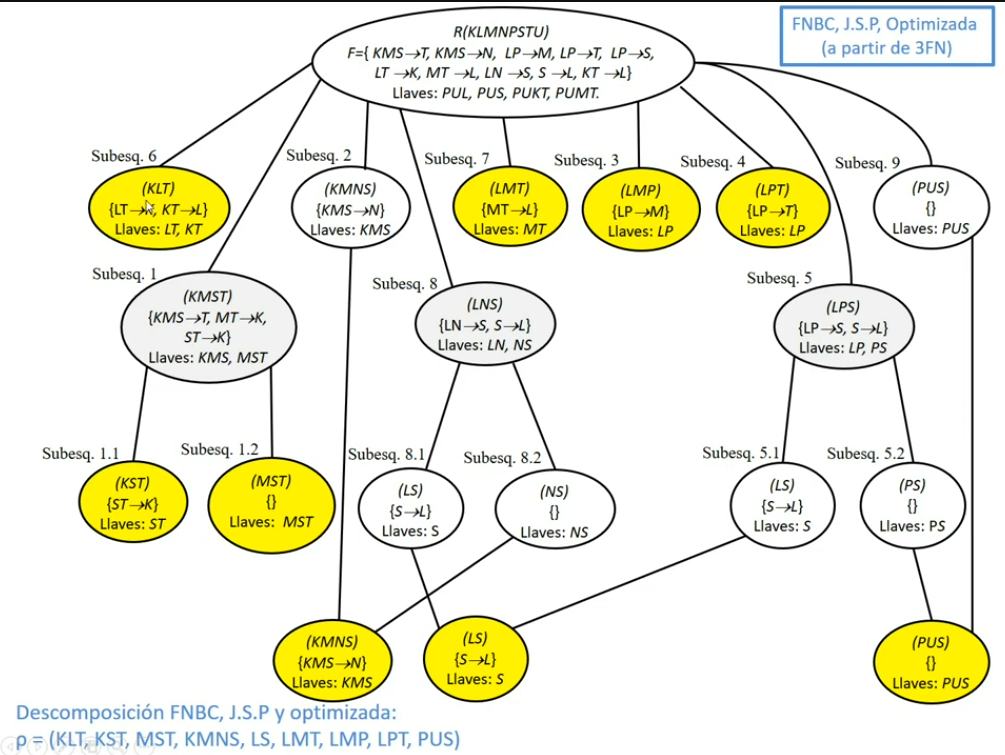
Ejemplo el subesquema 2 y el 8.2 se unen:



Seguimos uniendo y nos damos cuenta que al unir el subesquema 3 y 4, este nuevo subesquema también incluye al 7, al intentar unir los 3 vemos que no respeta la FNBC por lo tanto esta última unión no se agrega.



Por lo tanto, quedaría asi:



**Algoritmo FNBC:**

1. Tenemos que ver para cada subesquema que las dependencias respeten la FNBC, para esto, el lado izquierdo de la dependencia tiene que ser super llave.
2. Descomponemos los subesquemas que no respetan la FNBC con el algoritmo burbuja.
   1. Algoritmo burbuja: Partimos en 2 al esquema que tiene df que no respeta la FNBC, el subesquema izq tendrá la dependencia, su conjunto y se calcula sus llaves (si no queda en FNBC probar con otra df hacer el subesquema). El subesquema derecho tendrá el conjunto de antes restándole de la dependencia del subequema izq el consecuente, y se calcula las df y las llaves.
3. Unimos subesquemas que tengan mismas llaves o que los conjuntos esten incluidos en otros subesquemas, esto solo si respeta la FNBC