Introduction aux
Bases de données Relationnelles

Département Génie Biologique
GB4 – année 2023–2024

Jean-Paul Comet[1]
Nadia Abchiche-Mimouni[1]

[1]Université Côte d'Azur

Partie 6 : Normalisation d'une relation

Normalisation d'une relation

- La normalisation des relations a pour but d'aider la conception de schémas relationnels.
- À travers cette normalisation, on va pouvoir améliorer, selon certains critères, la qualité d'un schéma relationnel.

Exemple. Lequel de ces deux schémas relationnels et le meilleur ? pourquoi ?

`Produit(NP,NomP,Couleur,Poids)`	`Produit(NP,NomP,Couleur,Poids)`
`Fournisseur(NF,NomF,Adr,Tel)`	`Fournisseur(NF,NomF,Adr)`
`Livraison(NP,NF,Date,Qté)`	`Livraison(NP,NF,Date,TelF,Qté)`

Normalisation d'une relation

- La normalisation des relations a pour but d'aider la conception de schémas relationnels.
- À travers cette normalisation, on va pouvoir améliorer, selon certains critères, la qualité d'un schéma relationnel.

Exemple. Lequel de ces deux schémas relationnels et le meilleur ? pourquoi ?

`Produit(NP,NomP,Couleur,Poids)`	`Produit(NP,NomP,Couleur,Poids)`
`Fournisseur(NF,NomF,Adr,Tel)`	`Fournisseur(NF,NomF,Adr)`
`Livraison(NP,NF,Date,Qté)`	`Livraison(NP,NF,Date,TelF,Qté)`

Le second soulève les problèmes suivants :

- Si un fournisseur ne livre plus, on perd son n^o de téléphone
- Si un fournisseur livre beaucoup, on duplique l'information du n^o de tél.
- Pour enregistrer une nouvelle livraison, il faut recopier le n^o de tél.
- Si un fournisseur change de n^o de tél, il faut propager la modification sur toutes les livraisons de ce fournisseur.

La *qualité* de la relation peut être évaluée par son *degré de normalisation* : une relation peut être

- $1^{\text{ère}}$, $2^{\text{ème}}$, $3^{\text{ème}}$ forme normale,
- en forme normale de Boyce-Codd ou
- en $4^{\text{ème}}$ forme normale.

Plus le degré de normalisation est élevé, plus la relation a de *bonnes* propriétés.

Graphe des dépendances fonctionnelles

Etant donné une relation `R(X,Y,Z)`, il existe une **dépendance fonctionnelle**, ou DF, de `Y` vers `Z`, (`Y` et `Z` étant soit des attributs soit des ensembles d'attributs) souvent notée $Y \rightarrow Z$, si : pour tout couple *envisageable* de n-uplets de R, s'ils ont même valeurs pour l'attribut Y, ils ont aussi même valeur pour Z.
On appelle Y la *source* de la dépendance fonctionnelle et Z la *cible*.

Exemple. Pour la relation `Produit(NP,NomP,Couleur,Poids)`, les dépendances sont les suivantes, si on considère que deux produits différents ne peuvent pas avoir le même nom :

- `NP` → `NomP`, `NP` → `Poids`, `NP` → `Couleur`,
- `NomP` → `NP`, `NomP` → `Couleur`, `NomP` → `Poids`,
- `NP` → `(NomP,Poids,Couleur)`
- `(NP,NomP)` → `Poids`, `(NP,NomP)` → `Couleur`,
- …

Dépendance Fonctionnelle

Une **dépendance fonctionnelle** $X \rightarrow B$ est **élémentaire** (on dit parfois complète, pleine) si, X et B étant resp. un groupe d'attributs et un attribut d'une même relation, B n'est pas fonctionnellement dépendant d'un sous-ensemble de X.

Exemple. Pour les relations $\left\{ \begin{array}{l} \texttt{Produit(NP,NomP,Couleur,Poids)} \\ \texttt{Livraison(NP,NF,Date,Qté)} \end{array} \right.$

- Les DF $\left\{ \begin{array}{l} \texttt{NP} \rightarrow \texttt{(Poids, Couleur)} \\ \texttt{(NP,NomP)} \rightarrow \texttt{Poids} \end{array} \right.$ ne sont pas élémentaires.
- La DF `(NP,NF,date)` → `Qté` est élémentaire (si un même fournisseur ne se déplace qu'une seule fois par jour pour un client et un produit).

Chaque DF exprime un fait du monde réel.

- Les DFs élémentaires expriment des faits élémentaires.
  `NP` → `Couleur` : chaque produit, identifié par un n^o, a une couleur bien caractérisée.
  `(NP,NF,date)` → `Qté` : un fournisseur ne peut pas livrer le même jour le même produit au même client avec des quantités différentes.

Propriétés :

- Si pour une relation donnée, on a les dépendances fonctionnelles $X \rightarrow Y$ et $Y \rightarrow Z$, alors on a aussi la dépendance fonctionnelle $X \rightarrow Z$ qui est dite **déduite** des deux autres.

Dépendance Fonctionnelle

Etant donnés une relation et un ensemble F de DF portant sur les attributs de cette relation, on appelle **graphe minimum** des DF de la relation, tout graphe (S, A) dont

- S est l'ensemble des attributs de la relation et
- A est un sous-ensemble de F de DF élémentaires non déduites, à partir desquelles toute dépendance de F peut être déduite.

Il est facile de savoir si une DF est déduite des autres ou non.
On construit le graphe de toutes les dépendances fonctionnelles.
Une DF $X \rightarrow Y$ est déduite s'il existe un autre chemin allant de X à Y.
Remarques.

1. Une **clé** est un sous-ensemble d'attributs respectant les 2 contraintes :
   - **Unicité :** 2 *n*-uplets distincts ne peuvent avoir la même valeur pour ces attributs
   - **Irréductibilité :** il n'existe pas de sous-ensemble strict de la clé garantissant la règle d'unicité.
2. Tout graphe de DF (minimum ou non) peut être utilisé pour la recherche des identifiants : les identifiants correspondent aux ensembles minimaux d'attributs X pour lesquels les chemins, partant de ces attributs, atteignent *tous* les autres attributs du graphe.
3. **Concept de sur-clé :**
   une surclé est un sous-ensemble d'attributs de la relation, respectant uniquement la règle d'unicité

Couverture minimale

- Une couverture minimale d'un ensemble de DF est un sous ensemble minimum de dépendances fonctionnelles élémentaires (DFE) qui permettent de générer toutes les autres.
- Tout ensemble de dépendances fonctionnelles possède une couverture minimale (pas forcement unique) composée de dépendances fonctionnelles dont les parties droites contiennent 1 seul attribut.

1$^{\text{er}}$ **Algorithme de calcul de couverture minimale**

1. Décomposition de chaque DF pour avoir un seul attribut à droite
2. Suppression des attributs en surnombre à gauche : Pour tout $X \rightarrow Y$, s'il existe un $Z \subset X$ tel que $Z \rightarrow Y$ alors remplacer $X \rightarrow Y$ par $Z \rightarrow Y$
3. Suppression des DF redondantes (qu'on peut obtenir par transitivité)

Exemple : $A \rightarrow B \mid B, C \rightarrow D \mid A, C \rightarrow B, D, E \mid D \rightarrow E$

1. Etape 1 : décomposition
2. Etape 2 : Suppression des attributs en surnombre à gauche
3. Etape 3 : Supprimer la redondance

⇒ la couverture minimale :

Couverture minimale

- Une couverture minimale d'un ensemble de DF est un sous ensemble minimum de dépendances fonctionnelles élémentaires (DFE) qui permettent de générer toutes les autres.
- Tout ensemble de dépendances fonctionnelles possède une couverture minimale (pas forcement unique) composée de dépendances fonctionnelles dont les parties droites contiennent 1 seul attribut.

1^{er} **Algorithme de calcul de couverture minimale**

1. Décomposition de chaque DF pour avoir un seul attribut à droite
2. Suppression des attributs en surnombre à gauche : Pour tout $X \to Y$, s'il existe un $Z \subset X$ tel que $Z \to Y$ alors remplacer $X \to Y$ par $Z \to Y$
3. Suppression des DF redondantes (qu'on peut obtenir par transitivité)

Exemple : $A \to B \mid B, C \to D \mid A, C \to B, D, E \mid D \to E$

1. Etape 1 : décomposition
   $A \to B \mid B, C \to D \mid {\color{red} A, C \to B \mid A, C \to D \mid A, C \to E} \mid D \to E$
2. Etape 2 : Suppression des attributs en surnombre à gauche
3. Etape 3 : Supprimer la redondance

$\Rightarrow$ la couverture minimale :

Couverture minimale

- Une couverture minimale d'un ensemble de DF est un sous ensemble minimum de dépendances fonctionnelles élémentaires (DFE) qui permettent de générer toutes les autres.
- Tout ensemble de dépendances fonctionnelles possède une couverture minimale (pas forcement unique) composée de dépendances fonctionnelles dont les parties droites contiennent 1 seul attribut.

1^{er} **Algorithme de calcul de couverture minimale**

1. Décomposition de chaque DF pour avoir un seul attribut à droite
2. Suppression des attributs en surnombre à gauche : Pour tout $X \to Y$, s'il existe un $Z \subset X$ tel que $Z \to Y$ alors remplacer $X \to Y$ par $Z \to Y$
3. Suppression des DF redondantes (qu'on peut obtenir par transitivité)

Exemple : $A \to B \mid B, C \to D \mid A, C \to B, D, E \mid D \to E$

1. Etape 1 : décomposition
   $A \to B \mid B, C \to D \mid A, C \to B \mid A, C \to D \mid A, C \to E \mid D \to E$
2. Etape 2 : Suppression des attributs en surnombre à gauche
   $A \to B \mid B, C \to D \mid A, C \to D \mid A, C \to E \mid D \to E$
3. Etape 3 : Supprimer la redondance

⇒ la couverture minimale :

Couverture minimale

- Une couverture minimale d'un ensemble de DF est un sous ensemble minimum de dépendances fonctionnelles élémentaires (DFE) qui permettent de générer toutes les autres.
- Tout ensemble de dépendances fonctionnelles possède une couverture minimale (pas forcement unique) composée de dépendances fonctionnelles dont les parties droites contiennent 1 seul attribut.

1^{er} **Algorithme de calcul de couverture minimale**

1. Décomposition de chaque DF pour avoir un seul attribut à droite
2. Suppression des attributs en surnombre à gauche : Pour tout $X \rightarrow Y$, s'il existe un $Z \subset X$ tel que $Z \rightarrow Y$ alors remplacer $X \rightarrow Y$ par $Z \rightarrow Y$
3. Suppression des DF redondantes (qu'on peut obtenir par transitivité)

Exemple : $A \rightarrow B \mid B, C \rightarrow D \mid A, C \rightarrow B, D, E \mid D \rightarrow E$

1. Etape 1 : décomposition
   $A \rightarrow B \mid B, C \rightarrow D \mid A, C \rightarrow B \mid A, C \rightarrow D \mid A, C \rightarrow E \mid D \rightarrow E$
2. Etape 2 : Suppression des attributs en surnombre à gauche
   $A \rightarrow B \mid B, C \rightarrow D \mid A, C \rightarrow D \mid A, C \rightarrow E \mid D \rightarrow E$
3. Etape 3 : Supprimer la redondance
   $(A, C \rightarrow D)$ redondante : se déduit de $(A \rightarrow B)$ et $(B, C \rightarrow D)$.
   $(A, C \rightarrow E)$ redondante : se déduit de $(A \rightarrow B)$, $(B, C \rightarrow D)$ et $(D \rightarrow E)$.

⇒ la couverture minimale :

Couverture minimale

- Une couverture minimale d'un ensemble de DF est un sous ensemble minimum de dépendances fonctionnelles élémentaires (DFE) qui permettent de générer toutes les autres.
- Tout ensemble de dépendances fonctionnelles possède une couverture minimale (pas forcement unique) composée de dépendances fonctionnelles dont les parties droites contiennent 1 seul attribut.

1^{er} **Algorithme de calcul de couverture minimale**

1. Décomposition de chaque DF pour avoir un seul attribut à droite
2. Suppression des attributs en surnombre à gauche : Pour tout $X \to Y$, s'il existe un $Z \subset X$ tel que $Z \to Y$ alors remplacer $X \to Y$ par $Z \to Y$
3. Suppression des DF redondantes (qu'on peut obtenir par transitivité)

Exemple : $A \to B \mid B, C \to D \mid A, C \to B, D, E \mid D \to E$

1. Etape 1 : décomposition
   $A \to B \mid B, C \to D \mid A, C \to B \mid A, C \to D \mid A, C \to E \mid D \to E$
2. Etape 2 : Suppression des attributs en surnombre à gauche
   $A \to B \mid B, C \to D \mid A, C \to D \mid A, C \to E \mid D \to E$
3. Etape 3 : Supprimer la redondance
   $(A, C \to D)$ redondante : se déduit de $(A \to B)$ et $(B, C \to D)$.
   $(A, C \to E)$ redondante : se déduit de $(A \to B)$, $(B, C \to D)$ et $(D \to E)$.

⇒ la couverture minimale : $A \to B \mid B, C \to D \mid D \to E$.

Décomposition d'une relation

- *décomposer* d'une relation $\equiv \left\{ \begin{array}{l} \text{transformer la relation en un ensemble} \\ \text{de relations satisfaisantes qui} \\ \text{contiennent les mêmes informations.} \end{array} \right.$
- On peut décomposer une relation en un ensemble de relations projetées, si à partir des relations projetées, on peut retrouver la relation initiale par jointure.

Une décomposition d'une relation $R(X, Y, Z)$ en deux relations $R_1 = \pi[X, Y]R$ et $R_2 = \pi[X, Z]R$ est dite **sans perte d'information** si $R = R_1 \bowtie R_2$

Exemple : Soit la relation `pers(`<u>`numPers`</u>`,nom,adresse)`.

- La décomposition de la relation `pers` en deux relations $\pi\begin{bmatrix} numPers, \\ nom \end{bmatrix}\texttt{pers}$ et $\pi\begin{bmatrix} numPers, \\ adresse \end{bmatrix}\texttt{pers}$, est sans perte d'information.

Soit la relation pers :

1234	Comet	Nice
2345	Comet	Paris

$\pi\begin{bmatrix} numPers, \\ nom \end{bmatrix}\texttt{pers} =$

1234	Comet
2345	Comet

$\pi\begin{bmatrix} numPers, \\ adresse \end{bmatrix}\texttt{pers} =$

1234	Nice
2345	Paris

$\pi\begin{bmatrix} numPers, \\ nom \end{bmatrix}\texttt{pers} \bowtie \pi\begin{bmatrix} numPers, \\ adresse \end{bmatrix}\texttt{pers} =$

1234	Comet	Nice
2345	Comet	Paris

Décomposition d'une relation

- *décomposer* d'une relation $\equiv$ $\left\{\begin{array}{l}\text{transformer la relation en un ensemble}\\ \text{de relations satisfaisantes qui}\\ \text{contiennent les mêmes informations.}\end{array}\right.$
- On peut décomposer une relation en un ensemble de relations projetées, si à partir des relations projetées, on peut retrouver la relation initiale par jointure.

Une décomposition d'une relation $R(X, Y, Z)$ en deux relations $R_1 = \pi[X, Y]R$ et $R_2 = \pi[X, Z]R$ est dite **sans perte d'information** si $R = R_1 \bowtie R_2$

Exemple : Soit la relation `pers(numPers,nom,adresse)`.

- La décomposition de la relation `pers` en deux relations $\pi\begin{bmatrix}numPers,\\ nom\end{bmatrix}$ `pers` et $\pi\begin{bmatrix}nom,\\ adresse\end{bmatrix}$ `pers`, est avec perte d'information.

Soit la relation pers :

1234	Comet	Nice
2345	Comet	Paris

$\pi\begin{bmatrix}numPers,\\ nom\end{bmatrix}$ `pers` =

1234	Comet
2345	Comet

$\pi\begin{bmatrix}nom,\\ adresse\end{bmatrix}$ `pers` =

Comet	Nice
Comet	Paris

$\pi\begin{bmatrix}numPers,\\ nom\end{bmatrix}$ `pers` $\bowtie$ $\pi\begin{bmatrix}nom,\\ adresse\end{bmatrix}$ `pers` =

1234	Comet	Nice
1234	Comet	Paris
2345	Comet	Nice
2345	Comet	Paris

Décomposition d'une relation

- *décomposer* d'une relation $\equiv \left\{ \begin{array}{l} \text{transformer la relation en un ensemble} \\ \text{de relations satisfaisantes qui} \\ \text{contiennent les mêmes informations.} \end{array} \right.$
- On peut décomposer une relation en un ensemble de relations projetées, si à partir des relations projetées, on peut retrouver la relation initiale par jointure.

Une décomposition d'une relation $R(X, Y, Z)$ en deux relations $R_1 = \pi[X, Y]R$ et $R_2 = \pi[X, Z]R$ est dite **sans perte d'information** si $R = R_1 \bowtie R_2$

Exemple : Soit la relation `pers(numPers,nom,adresse)`.

- La décomposition de la relation `pers` en deux relations $\pi\begin{bmatrix} numPers, \\ nom \end{bmatrix}$`pers` et $\pi\begin{bmatrix} nom, \\ adresse \end{bmatrix}$`pers`, est avec perte d'information.

Si la décomposition est **avec perte d'information**, la relation obtenue par jointure naturelle est composée de plus de *n*-uplets !

Soit la relation pers :

1234	Comet	Nice
2345	Comet	Paris

$\pi\begin{bmatrix} numPers, \\ nom \end{bmatrix}$`pers` =

1234	Comet
2345	Comet

$\pi\begin{bmatrix} nom, \\ adresse \end{bmatrix}$`pers` =

Comet	Nice
Comet	Paris

$\pi\begin{bmatrix} numPers, \\ nom \end{bmatrix}$`pers` $\bowtie$ $\pi\begin{bmatrix} nom, \\ adresse \end{bmatrix}$`pers` =

1234	Comet	Nice
1234	Comet	Paris
2345	Comet	Nice
2345	Comet	Paris

Theoreme de Heath

Theoreme de Heath Toute relation $R(X, Y, Z)$ est décomposable sans perte d'information en $R_1 = \pi[X, Y]R$ et $R_2 = \pi[X, Z]R$, s'il y a dans R une dépendance fonctionnelle de X vers Y

Preuve.

1. Montrons que $R \subseteq R_1 \bowtie R_2$:
   $R_1 \bowtie R_2$ contient au moins tous les *n*-uplets de R, puisque tout *n*-uplet xyz de R crée un *n*-uplet xy dans R_1 et un *n*-uplet xz dans R_2. Ces deux *n*-uplets sont ensuite concaténées dans la jointure naturelle en xyz.
2. Montrons par l'absurde que $R_1 \bowtie R_2 \setminus R$ est vide :
   Soit xyz un *n*-uplet de $R_1 \bowtie R_2$ qui n'appartient pas à R.
   - xyz provient de deux *n*-uplets xy de R_1 et xz de R_2.
   - Si xy est dans R_1, c'est qu'il y avait dans R un *n*-uplet xyz'.
     Si xz est dans R_2, c'est qu'il y avait dans R un *n*-uplet xy'z.
   - comme xyz $\notin R$, on a $z' \neq z$ et $y' \neq y$.

   Donc pour un même x, on a deux *n*-uplets xyz' et xy'z qui ont des valeurs différentes pour Y, ce qui contredit la dépendance fonctionnelle $X \rightarrow Y$.

$1^{\text{ère}}$ et $2^{\text{ème}}$ formes normales

Une relation est en **première forme normale** si chaque valeur de chaque attribut de chaque *n*-uplet est une valeur simple (tous les attributs sont simples et monovalués).

Une relation est en **deuxième forme normale** si elle est en première forme normale et si chaque attribut qui ne fait partie d'aucun identifiant, ne dépend pas d'une sous partie stricte d'une clef (c-à-d *dépend de tout identifiant entier*), autrement dit, si toutes les dépendances fonctionnelles entre la clé et les autres attributs sont élémentaires.

≪ *Tout l'identifiant est utile.* ≫

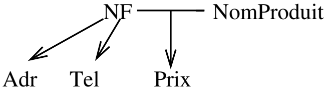
La méthode à suivre pour normaliser une relation est la suivante :

- vérifier que la relation est en première forme normale,
- établir un graphe minimum de dépendances fonctionnelles,
- déterminer tous ses identifiants,
- déterminer à l'aide du graphe sa forme normale,
- si la relation n'est pas (suffisamment) normalisée, décomposer à l'aide du graphe la relation en relations mieux normalisées.

$1^{\text{ère}}$ et $2^{\text{ème}}$ formes normales

Exemple. `Fournisseur1(NF,NomProduit,Adr,Tel,Prix)` supposée en 1FN.

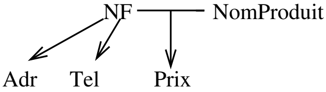
- Les DF : $\left\{ \begin{array}{l} \texttt{(nomProd,NF)} \longrightarrow \texttt{prix}, \\ \texttt{NF} \longrightarrow \texttt{Adr et} \\ \texttt{NF} \longrightarrow \texttt{Tel}. \end{array} \right.$



$1^{\text{ère}}$ et $2^{\text{ème}}$ formes normales

Exemple. $\texttt{Fournisseur1(}\underline{\texttt{NF,NomProduit}}\texttt{,Adr,Tel,Prix)}$ supposée en 1FN.

- Les DF : $\begin{cases} \texttt{(nomProd,NF)} \longrightarrow \texttt{prix}, \\ \texttt{NF} \longrightarrow \texttt{Adr} \text{ et} \\ \texttt{NF} \longrightarrow \texttt{Tel}. \end{cases}$



Limitations :

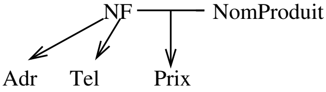
- Si 100 produits pour un fournisseur, on répétera 100 fois `NF`, `Adr` et `Tel`.
- Ajout d'un produit : il faut donner de nouveau `Adr` et `Tel`.
  Suppression momentanément de tous les produits : perte de `Adr` et `Tel`.
- Changement d'adresse d'un fournisseur : le faire pour tous les produits.

Raison : la relation n'est pas en $2^{\text{ème}}$ FN.

$1^{\text{ère}}$ et $2^{\text{ème}}$ formes normales

Exemple. $\mathtt{Fournisseur1(\underline{NF,NomProduit},Adr,Tel,Prix)}$ supposée en 1FN.

- Les DF : $\begin{cases} \mathtt{(nomProd,NF)} \longrightarrow \mathtt{prix}, \\ \mathtt{NF} \longrightarrow \mathtt{Adr} \text{ et} \\ \mathtt{NF} \longrightarrow \mathtt{Tel}. \end{cases}$



Limitations :

- Si 100 produits pour un fournisseur, on répétera 100 fois `NF`, `Adr` et `Tel`.
- Ajout d'un produit : il faut donner de nouveau `Adr` et `Tel`.
  Suppression momentanément de tous les produits : perte de `Adr` et `Tel`.
- Changement d'adresse d'un fournisseur : le faire pour tous les produits.

Raison : la relation n'est pas en $2^{\text{ème}}$ FN. **On décompose :**

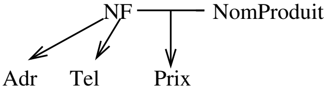
- il existe 3 DF dont 2 ont pour source un sous-ensemble de la source de l'autre. En effet, on a : $\mathtt{(nomProd,NF)} \longrightarrow \mathtt{prix}$, $\mathtt{NF} \longrightarrow \mathtt{Adr}$ et $\mathtt{NF} \longrightarrow \mathtt{Tel}$.

1. Pour chacune de ces sources de DF,
   - on crée une relation ayant pour attributs la source et tous les attributs en DF directe de cette source,
   - en s'assurant qu'une (ou +) de ces deux sources est contenue dans les attributs communs aux 2 relations créées (th. de Heath)

$1^{\text{ère}}$ et $2^{\text{ème}}$ formes normales

Exemple. `Fournisseur1(`$\underline{\texttt{NF,NomProduit}}$`,Adr,Tel,Prix)` supposée en 1FN.

- Les DF : $\left\{\begin{array}{l}(\texttt{nomProd,NF}) \longrightarrow \texttt{prix},\\ \texttt{NF} \longrightarrow \texttt{Adr} \text{ et}\\ \texttt{NF} \longrightarrow \texttt{Tel}.\end{array}\right.$



Limitations :

- Si 100 produits pour un fournisseur, on répétera 100 fois `NF`, `Adr` et `Tel`.
- Ajout d'un produit : il faut donner de nouveau `Adr` et `Tel`.
  Suppression momentanément de tous les produits : perte de `Adr` et `Tel`.
- Changement d'adresse d'un fournisseur : le faire pour tous les produits.

Raison : la relation n'est pas en $2^{\text{ème}}$ FN. **On décompose :**

- il existe 3 DF dont 2 ont pour source un sous-ensemble de la source de l'autre. En effet, on a : $(\texttt{nomProd,NF}) \longrightarrow \texttt{prix}$, $\texttt{NF} \longrightarrow \texttt{Adr}$ et $\texttt{NF} \longrightarrow \texttt{Tel}$.

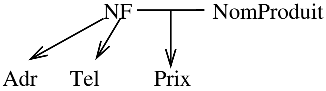
1. Pour chacune de ces sources de DF,
   - on crée une relation ayant pour attributs la source et tous les attributs en DF directe de cette source,
   - en s'assurant qu'une (ou +) de ces deux sources est contenue dans les attributs communs aux 2 relations créées (th. de Heath)

$$\Rightarrow \left\{\begin{array}{l}\texttt{Fournisseur(}\underline{\texttt{NF}}\texttt{,Adr,Tel)}\\ \texttt{Catalogue(}\underline{\texttt{NF,NomProduit}}\texttt{,Prix)}.\end{array}\right.$$

$1^{\text{ère}}$ et $2^{\text{ème}}$ formes normales

Exemple. `Fournisseur1(NF,NomProduit,Adr,Tel,Prix)` supposée en 1FN.

- Les DF : $\left\{ \begin{array}{l} \texttt{(nomProd,NF)} \longrightarrow \texttt{prix}, \\ \texttt{NF} \longrightarrow \texttt{Adr} \text{ et} \\ \texttt{NF} \longrightarrow \texttt{Tel}. \end{array} \right.$



Limitations :

- Si 100 produits pour un fournisseur, on répétera 100 fois `NF`, `Adr` et `Tel`.
- Ajout d'un produit : il faut donner de nouveau `Adr` et `Tel`.
  Suppression momentanément de tous les produits : perte de `Adr` et `Tel`.
- Changement d'adresse d'un fournisseur : le faire pour tous les produits.

Raison : la relation n'est pas en $2^{\text{ème}}$ FN. **On décompose :**

- il existe 3 DF dont 2 ont pour source un sous-ensemble de la source de l'autre. En effet, on a : `(nomProd,NF)` $\longrightarrow$`prix`, `NF` $\longrightarrow$`Adr` et `NF` $\longrightarrow$`Tel`.

1. Pour chacune de ces sources de DF,
   - on crée une relation ayant pour attributs la source et tous les attributs en DF directe de cette source,
   - en s'assurant qu'une (ou +) de ces deux sources est contenue dans les attributs communs aux 2 relations créées (th. de Heath)

$\Rightarrow \left\{ \begin{array}{l} \texttt{Fournisseur(NF,Adr,Tel)} \\ \texttt{Catalogue(NF,NomProduit,Prix)}. \end{array} \right.$

- On vérifie que cette décomposition est sans perte d'information, et
- sans perte de dépendances fonctionnelles (toutes les DF de la relation initiale se retrouvent, éventuellement pas transitivité, à partir des DF des 2 relations créées).

$1^{\text{ère}}$ et $2^{\text{ème}}$ formes normales

Autre exemple : `joueur(personne, sport, taille)`

- identifiant `(personne, sport)`
- La DF `personne`⟶`taille` fait que la relation n'est pas en $2^{\text{ème}}$ FN.

⇒ On décompose la relation en $\left\{ \begin{array}{l} \texttt{pratique(personne, sport)} \\ \texttt{hauteur(personne, taille)}. \end{array} \right.$

- La relation `joueur` est alors retrouvée par une jointure naturelle.

$3^{\text{ème}}$ forme normale

« La 3NF permet d'éliminer les DF transitives. »

Une relation est en **troisième forme normale** si elle est en deuxième forme normale et si chaque attribut qui ne fait partie d'aucun identifiant, *dépend uniquement et directement* de sur-clés entières. (Chaque attribut ne dépend pas d'un ensemble d'attributs qui ne soit pas une sur-clef. La dépendance est directe : elle n'est pas dû à la transitivité des DF).

Remarque : 3NF $\Rightarrow$ 2NF (même sans la première condition)
Exemple : `Fournisseur2(NF,Pays, Ville)` est en 2FN mais pas 3NF.
- Les DF : `NF` $\rightarrow$ `Ville` et `Ville` $\rightarrow$ `Pays`.
- La DF `NF` $\rightarrow$ `Pays` est déduite (si pas d'homonymie entre villes)
- Le graphe minimum des DF de `Fournisseur2` est donc : `NF` $\rightarrow$ `Ville` $\rightarrow$ `Pays`.

Il y a redondance : le pays d'une ville est répété.

On décompose donc en : $\left\{ \begin{array}{l} \texttt{Fourn(}\underline{\texttt{NF}}\texttt{,Ville)} \\ \texttt{Geo(}\underline{\texttt{Ville}}\texttt{,Pays)}. \end{array} \right.$

- sans perte d'information (Ville est identifiant pour Géo)
- sans perte de DF (toutes les DF de la relation initiale se retrouvent à partir des DF des deux relations créées).

$3^{\text{ème}}$ forme normale

Exemple :
`Voiture(Imm, Marque, Type, Puissance, Couleur)` n'est pas en 3FN.
`Type` (non clé) détermine les attributs `Marque` et `Puissance`.

Il faut décomposer la relation : $\left\{ \begin{array}{l} \texttt{Voiture (Imm, Type, Couleur)} \\ \texttt{Modèle (Type, Marque, Puissance)} \end{array} \right.$

Remarque :

- On peut toujours décomposer une relation en un ensemble de relations qui sont chacune en 3FN.
- Ce n'est pas le cas pour les formes normales suivantes. D'où l'intérêt de cette troisième forme normale.

$3^{\text{ème}}$ forme normale

Algorithme de décomposition 3NF (SPI et préservant les DF)
Soit un schéma de relation R, et une **couverture minimale** F' de dépendances fonctionnelles s'appliquant sur R. Une décomposition SPI et SPDF de R, qu'on appellera D, est construite de la façon suivante :

1. pour chaque DF $X \to A$, on crée une relation `Ri(X,A)`,
2. si on a plusieurs DF telles que $X \to A_1$, $X \to A_2$, ..., $X \to A_n$,
   alors on regroupe tous ces attributs dans une m̂ relation $\mathtt{R_j}$ $(\mathtt{X,A_1\ ,...,A_n})$,

$3^{\text{ème}}$ forme normale

Algorithme de décomposition 3NF (SPI et préservant les DF)
Soit un schéma de relation R, et une **couverture minimale** F' de dépendances fonctionnelles s'appliquant sur R. Une décomposition SPI et SPDF de R, qu'on appellera D, est construite de la façon suivante :

1. pour chaque DF $X \rightarrow A$, on crée une relation `Ri(X,A)`,
2. si on a plusieurs DF telles que $X \rightarrow A_1$, $X \rightarrow A_2$, ..., $X \rightarrow A_n$,
   alors on regroupe tous ces attributs dans une m̂ relation $\mathtt{R_j}\ (\mathtt{X,A_1\ ,\ldots,A_n})$,
3. Pour avoir une décomposition SPI :
   - il faut s'assurer qu'il y ait au-moins une clé de R dans au moins une des relations de décomposition.
   - Si ce n'est pas le cas, il faut soit ajouter une relation contenant une clé de R, soit ajouter des attributs dans une des relations de décomposition afin de satisfaire cette contrainte.

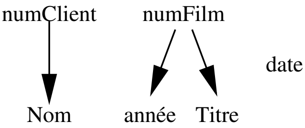
$3^{\text{ème}}$ forme normale

Exemple.

- $\texttt{Location(}\underline{\texttt{numFilm, numClient, date}}\texttt{, titre, année, nom)}$

  $$\text{Les DF : } \left\{ \begin{array}{l} \texttt{numFilm} \longrightarrow \texttt{titre}, \\ \texttt{numFilm} \longrightarrow \texttt{année}, \\ \texttt{numClient} \longrightarrow \texttt{nom}. \end{array} \right.$$

- Le graphe de DF :



$3^{\text{ème}}$ forme normale

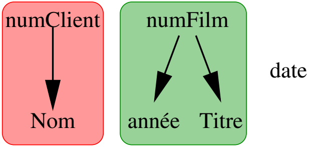
Exemple.

- $\texttt{Location(}\underline{\texttt{numFilm, numClient, date}}\texttt{, titre, année, nom)}$

  $$\text{Les DF : }\begin{cases} \texttt{numFilm} \longrightarrow \texttt{titre}, \\ \texttt{numFilm} \longrightarrow \texttt{année}, \\ \texttt{numClient} \longrightarrow \texttt{nom}. \end{cases}$$

- Les étapes 1 et 2 donnent 2 tables : $\begin{cases} \texttt{R1(}\underline{\texttt{numFilm}}\texttt{, titre, année)} \\ \texttt{R2(}\underline{\texttt{numClient}}\texttt{, nom)}. \end{cases}$

- Le graphe de DF :

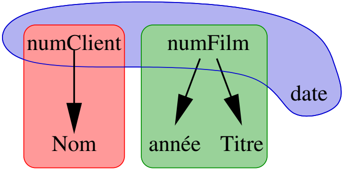


3$^{\text{ème}}$ forme normale

Exemple.

- $\texttt{Location(}\underline{\texttt{numFilm, numClient, date}}\texttt{, titre, année, nom)}$

  Les DF : $\begin{cases} \texttt{numFilm} \longrightarrow \texttt{titre}, \\ \texttt{numFilm} \longrightarrow \texttt{année}, \\ \texttt{numClient} \longrightarrow \texttt{nom}. \end{cases}$
- Les étapes 1 et 2 donnent 2 tables : $\begin{cases} \texttt{R1(}\underline{\texttt{numFilm}}\texttt{, titre, année)} \\ \texttt{R2(}\underline{\texttt{numClient}}\texttt{, nom)}. \end{cases}$
- L'étape 3 donne une table supplémentaire :
  $\texttt{R2(}\underline{\texttt{numClient, numFilm, date}}\texttt{)}$.
- Le graphe de DF :



$3^{\text{ème}}$ forme normale

Remarques.

1. L'algorithme a besoin d'une couverture minimale de DF. La recherche d'un graphe de DF minimum peut être compliquée.
2. S'il y a dans `R` des attributs n'intervenant dans aucune DF, alors ils font partie de l'identifiant. Ils seront donc intégrés dans la table créée à l'étape 3.
3. L'étape 3 est importante pour pouvoir éliminer les *n*-uplets excédentaires lors de la jointure pour reconstituer la relation initiale.
4. Parfois cet algorithme mène à des décompositions redondantes.

   Exemple :

   - soit la relation `enseignement(numEtud,matière,prof)` chaque professeur n'enseigne qu'une seule matière.
   - DF : `(numEtud,matiere)` ⟶ `prof`   `prof` ⟶ `matiere`.
   - Cette relation est déjà en 3FN.
   - l'algorithme mène à : $\begin{cases} \texttt{(numEtud,matière,prof)} \\ \texttt{(prof,matière)}. \end{cases}$

     La deuxième relation est incluse dans la première.
5. Il faut donc ensuite supprimer du résultat les relations qui sont incluses dans d'autres.

$3^{\text{ème}}$ forme normale

Remarques.

1. L'algorithme a besoin d'une couverture minimale de DF. La recherche d'un graphe de DF minimum peut être compliquée.
2. S'il y a dans `R` des attributs n'intervenant dans aucune DF, alors ils font partie de l'identifiant. Ils seront donc intégrés dans la table créée à l'étape 3.
3. L'étape 3 est importante pour pouvoir éliminer les *n*-uplets excédentaires lors de la jointure pour reconstituer la relation initiale.
4. Parfois cet algorithme mène à des décompositions redondantes.

   Exemple :

   - soit la relation `enseignement(numEtud,matière,prof)` chaque professeur n'enseigne qu'une seule matière.
   - DF : $\texttt{(numEtud,matiere)} \longrightarrow \texttt{prof} \quad \texttt{prof} \longrightarrow \texttt{matiere}$.
   - Cette relation est déjà en 3FN.
   - l'algorithme mène à : $\left\{\begin{array}{l}\texttt{(numEtud,matière,prof)}\\ \texttt{(prof,matière)}.\end{array}\right.$

     La deuxième relation est incluse dans la première.
5. Il faut donc ensuite supprimer du résultat les relations qui sont incluses dans d'autres.

⚠ **L'algorithme n'est pas applicable si le graphe de DF n'est pas minimal.**

$3^{\text{ème}}$ forme normale

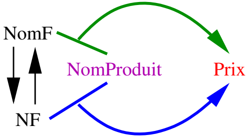
2ème algorithme de décomposition 3NF (sans couverture minimale) :

1. Repérer dans la relation `R` une DF $L \longrightarrow A$ qui ne satisfait pas aux contraintes de la 3ème forme normale : $A \notin L$, `L` n'est pas une surclef, et `A` ne fait pas partie d'un identifiant.
2. On projette alors la relation `R` en 2 tables :
   - une sur les attributs de `R` privé de `A`
   - l'autre sur les attributs $\{L,A\}$. Pour cette table, `L` est identifiant.
3. On réitère ce processus (en sachant que toute table binaire est en 3NF).

Forme normale de Boyce-Codd

Une relation est en **forme normale de Boyce-Codd** si elle est en troisième forme normale et si toute source de DF élémentaire est une *sur-clé*,
autrement dit,
une relation est en **FNBC** si et seulement si les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles une clé (ou sur-clé) détermine un attribut.

Le graphe de DF :



Exemple.
`Catalogue3(NF,NomF,NomProduit,Prix)`
en 3NF mais pas en FNBC
(pas d'homonyme chez les fournisseurs)

Les identifiants sont : `(NF,NomProduit)`
`(NomF,NomProduit)`

Il y a donc deux graphes de DF minimum :

$$NF \leftrightarrow NomF, \quad (NomProduit, NomF) \rightarrow Prix$$

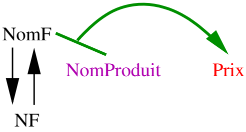
ou alors

$$NF \leftrightarrow NomF, \quad (NomProduit, NF) \rightarrow Prix$$

Forme normale de Boyce-Codd

Une relation est en **forme normale de Boyce-Codd** si elle est en troisième forme normale et si toute source de DF élémentaire est une *sur-clé*,
autrement dit,
une relation est en **FNBC** si et seulement si les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles une clé (ou sur-clé) détermine un attribut.

Le graphe de DF :



Exemple.
`Catalogue3(NF,NomF,NomProduit,Prix)`
en 3NF mais pas en FNBC
(pas d'homonyme chez les fournisseurs)
Les identifiants sont : `(NF,NomProduit)` `(NomF,NomProduit)`

Il y a donc deux graphes de DF minimum :

$$NF \leftrightarrow NomF, \quad (NomProduit, NomF) \rightarrow Prix$$

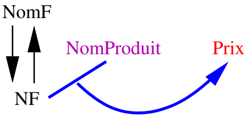
ou alors

$$NF \leftrightarrow NomF, \quad (NomProduit, NF) \rightarrow Prix$$

Forme normale de Boyce-Codd

Une relation est en **forme normale de Boyce-Codd** si elle est en troisième forme normale et si toute source de DF élémentaire est une *sur-clé*,
autrement dit,
une relation est en **FNBC** si et seulement si les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles une clé (ou sur-clé) détermine un attribut.

Le graphe de DF :



Exemple.
`Catalogue3(NF,NomF,NomProduit,Prix)`
en 3NF mais pas en FNBC
(pas d'homonyme chez les fournisseurs)
Les identifiants sont : `(NF,NomProduit)`
`(NomF,NomProduit)`

Il y a donc deux graphes de DF minimum :

$$NF \leftrightarrow NomF, \quad (NomProduit, NomF) \rightarrow Prix$$

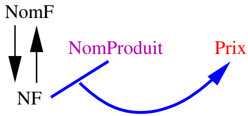
ou alors

$$NF \leftrightarrow NomF, \quad (NomProduit, NF) \rightarrow Prix$$

Forme normale de Boyce-Codd

Une relation est en **forme normale de Boyce-Codd** si elle est en troisième forme normale et si toute source de DF élémentaire est une *sur-clé*,
autrement dit,
une relation est en **FNBC** si et seulement si les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles une clé (ou sur-clé) détermine un attribut.

Le graphe de DF :



Exemple.
`Catalogue3(NF,NomF,NomProduit,Prix)`
en 3NF mais pas en FNBC
(pas d'homonyme chez les fournisseurs)

Les identifiants sont : `(NF,NomProduit)`
`(NomF,NomProduit)`

Il y a donc deux graphes de DF minimum :

$$NF \leftrightarrow NomF, \quad (NomProduit, NomF) \rightarrow Prix$$

ou alors

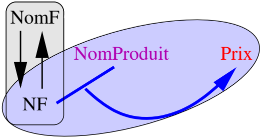
$$NF \leftrightarrow NomF, \quad (NomProduit, NF) \rightarrow Prix$$

- La relation `Catalogue3` n'est pas en FNBC : l'attribut NF est source complète de DF.
- Dans `Catalogue3`, il y a redondance entre NF et NomF

Forme normale de Boyce-Codd

Une relation est en **forme normale de Boyce-Codd** si elle est en troisième forme normale et si toute source de DF élémentaire est une *sur-clé*,
autrement dit,
une relation est en **FNBC** si et seulement si les seules dépendances fonctionnelles élémentaires sont celles dans lesquelles une clé (ou sur-clé) détermine un attribut.

Le graphe de DF :



Exemple.
`Catalogue3(NF,NomF,NomProduit,Prix)`
en 3NF mais pas en FNBC
(pas d'homonyme chez les fournisseurs)
Les identifiants sont : `(NF,NomProduit)`
`(NomF,NomProduit)`

Il y a donc deux graphes de DF minimum :

$$NF \leftrightarrow NomF, \quad (NomProduit, NomF) \rightarrow Prix$$

ou alors

$$NF \leftrightarrow NomF, \quad (NomProduit, NF) \rightarrow Prix$$

- La relation `Catalogue3` n'est pas en FNBC : l'attribut NF est source complète de DF.
- Dans `Catalogue3`, il y a redondance entre NF et NomF
- On décompose donc en : $\left\{ \begin{array}{l} \texttt{Catalogue(}\underline{\texttt{NF,NomProduit}}\texttt{,Prix)} \\ \texttt{Fournisseur(}\underline{\texttt{NF}}\texttt{,}\underline{\texttt{NomF}}\texttt{).} \end{array} \right.$

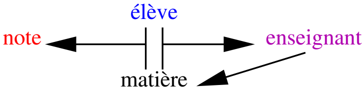
Forme normale de Boyce-Codd

Autres exemples.

1. `Collège(élève, matière, enseignant, note)` avec les DP suivantes :

$$\begin{cases} (\texttt{élève},\ \texttt{matière}) \rightarrow \texttt{enseignant},\ \texttt{note} \\ \texttt{enseignant} \rightarrow \texttt{matière}. \end{cases}$$

Les identifiants : `(élève, matière)` et `(élève, enseignant)`.
3NF.



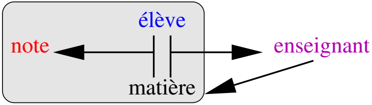
Forme normale de Boyce-Codd

Autres exemples.

1. `Collège(élève, matière, enseignant, note)` avec les DP suivantes :

$$\begin{cases} (\texttt{élève}, \texttt{matière}) \to \texttt{enseignant}, \texttt{note} \\ \texttt{enseignant} \to \texttt{matière}. \end{cases}$$

Les identifiants : `(élève, matière)` et `(élève, enseignant)`.
3NF.



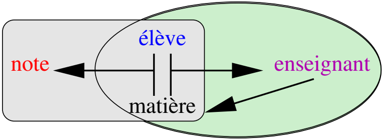
La relation `collège` est décomposée en :

$$\begin{cases} \texttt{notes}(\underline{\texttt{élève}, \texttt{matière}}, \texttt{note}) \\ \texttt{Enseignement} \ (\underline{\texttt{élève}, \texttt{enseignant}}, \texttt{matière}). \end{cases}$$

Forme normale de Boyce-Codd

Autres exemples.

1. `Collège(élève, matière, enseignant, note)` avec les DP suivantes :
   $$\begin{cases} (\texttt{élève}, \texttt{matière}) \to \texttt{enseignant}, \texttt{note} \\ \texttt{enseignant} \to \texttt{matière}. \end{cases}$$
   Les identifiants : `(élève, matière)` et `(élève, enseignant)`.
   3NF.



La relation `collège` est décomposée en :

$$\begin{cases} \texttt{notes}(\underline{\texttt{élève}, \texttt{matière}}, \texttt{note}) \\ \texttt{Enseignement}\ (\underline{\texttt{élève}, \texttt{enseignant}}, \texttt{matière}). \end{cases}$$

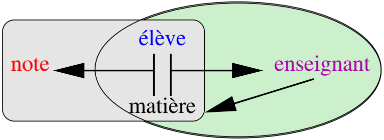
Forme normale de Boyce-Codd

Autres exemples.

1. Collège(élève, matière, enseignant, note) avec les DP suivantes :

$$\begin{cases} (\texttt{élève, matière}) \rightarrow \texttt{enseignant, note} \\ \texttt{enseignant} \rightarrow \texttt{matière}. \end{cases}$$

Les identifiants : (élève, matière) et (élève, enseignant).
3NF.



La relation collège est décomposée en :

$$\begin{cases} \texttt{notes}(\underline{\texttt{élève, matière}}, \texttt{note}) \\ \texttt{Enseignement } (\underline{\texttt{élève, enseignant}}, \texttt{matière}). \end{cases}$$

Enseignement ($\underline{\texttt{élève, enseignant}}$, Matière) n'est pas FNBC.

Forme normale de Boyce-Codd

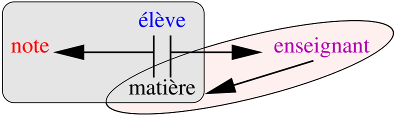
Autres exemples.

① Collège(élève, matière, enseignant, note) avec les DP suivantes :

$$\begin{cases} (\text{élève}, \text{matière}) \rightarrow \text{enseignant}, \text{note} \\ \text{enseignant} \rightarrow \text{matière}. \end{cases}$$

Les identifiants : (élève, matière) et (élève, enseignant).

3NF.



La relation collège est décomposée en :

$$\begin{cases} \text{notes}(\underline{\text{élève}, \text{matière}}, \text{note}) \\ \text{Enseignement } (\underline{\text{élève}, \text{enseignant}}, \text{matière}). \end{cases}$$

Enseignement (élève, enseignant, Matière) n'est pas FNBC.

On la décompose :

$$\begin{cases} \text{notes}(\underline{\text{élève}, \text{matière}}, \text{note}), \\ \text{Enseignant } (\underline{\text{enseignant}}, \text{Matière}) \\ \text{Enseignement } (\underline{\text{élève}, \text{enseignant}}). \end{cases}$$

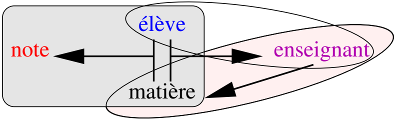
Forme normale de Boyce-Codd

Autres exemples.

1. Collège(élève, matière, enseignant, note) avec les DP suivantes :

   (élève, matière) → enseignant, note
   enseignant → matière.

   Les identifiants : (élève, matière) et (élève, enseignant).
   3NF.



La relation collège est décomposée en :

notes($\underline{\text{élève, matière}}$, note)
Enseignement ($\underline{\text{élève, enseignant}}$, matière).

Enseignement ($\underline{\text{élève, enseignant}}$, Matière) n'est pas FNBC.

On la décompose :

notes($\underline{\text{élève, matière}}$, note),
Enseignant ($\underline{\text{enseignant}}$, Matière)
Enseignement ($\underline{\text{élève, enseignant}}$).

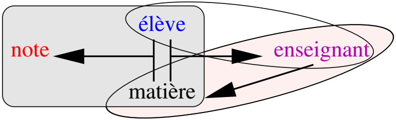
Forme normale de Boyce-Codd

Autres exemples.

① Collège(élève, matière, enseignant, note) avec les DP suivantes :

$$\begin{cases} (\text{élève, matière}) \rightarrow \text{enseignant, note} \\ \text{enseignant} \rightarrow \text{matière.} \end{cases}$$

Les identifiants : (élève, matière) et (élève, enseignant).
3NF.



La relation collège est décomposée en :

$$\begin{cases} \text{notes}(\underline{\text{élève, matière}}, \text{note}) \\ \text{Enseignement } (\underline{\text{élève, enseignant}}, \text{matière}). \end{cases}$$

Enseignement $(\underline{\text{élève, enseignant}}$, Matière) n'est pas FNBC.

On la décompose : $\begin{cases} \text{notes}(\underline{\text{élève, matière}}, \text{note}), \\ \text{Enseignant } (\underline{\text{enseignant}}, \text{Matière}) \\ \text{Enseignement } (\underline{\text{élève, enseignant}}). \end{cases}$

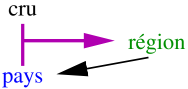
La dépendance fonctionnelle (élève, matière) → enseignant est perdue.

Forme normale de Boyce-Codd

② `Vins(Cru, Pays, Région)` avec les DF :
$Region \rightarrow Pays$ et $(Cru, Pays) \rightarrow Region$
3NF

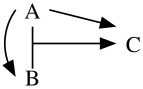
`Vins` est décomposée en : $\left\{ \begin{array}{l} \texttt{Crus (}\underline{\texttt{Cru, Région}}\texttt{)} \\ \texttt{Régions(}\underline{\texttt{Région}}\texttt{, Pays)}. \end{array} \right.$

Le graphe de DF :



La dépendance fonctionnelle $(Cru, Pays) \rightarrow Region$ est perdue.

③ `R₁(A, B, C)` avec les DF : $\left\{ \begin{array}{l} A \rightarrow B, C \\ (A, B) \rightarrow C \end{array} \right.$



La DF $A \rightarrow C$ est déduite de $A \rightarrow B$ et de $(A, B) \rightarrow C$.
L'identifiant est A.
Les DF élémentaires ont donc toutes pour source une sur-clé
(une clé pour $A \rightarrow B$, une sur-clé strict pour $(A, B) \rightarrow C$).
R_1 est donc en FNBC.

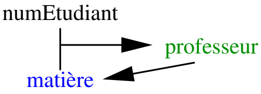
Forme normale de Boyce-Codd

Théorème de décomposition en FNBC : Toute relation admet au moins une décomposition en FNBC qui est sans perte d'informations ; cependant, une telle décomposition ne préserve généralement pas les dépendances fonctionnelles.

Exemple ne préservant pas les DF.

```
Enseignement(numEtudiant,
             matière,
             professeur)
```

avec les DF ci-contre.



$$\text{Si on décompose en } \begin{cases} \texttt{Enseigne}(\underline{\texttt{professeur}}, \texttt{matière}) \\ \texttt{suit}(\underline{\texttt{numEtudiant, professeur}}) \end{cases}$$

on ne préserve pas la DF (`numEtudiant`, `matière`) → `professeur`.

En effet, on peut dire qu'un étudiant suit 2 cours sur la même matière avec 2 professeurs différents. Cela n'était pas possible dans le schéma initial.

Forme normale de Boyce-Codd

Algorithme de décomposition FNBC (SPI) Soit un schéma de relation R, et un ensemble F de dépendances fonctionnelles s'appliquant sur R. Une décomposition SPI de R, qu'on appellera D, est construite de manière itérative :

1. D est initialisée à R,
2. Soit T une relation de D qui ne soit pas $FNBC$.
   → il y a une DF $X \rightarrow A$ qui n'est pas de la forme `clef`$\longrightarrow$`attribut`.
   On décompose alors T en :
   - T_1 contenant A et les attributs de X, et
   - T_2 contenant tous les attributs de T sauf A,
3. Dans D, la relation T est supprimée et on ajoute T_1 et T_2, et on boucle sur 2 jusqu'à ce que toutes les relations soient FNBC.

Conclusion

1. Cet algorithme permet de trouver une décomposition sans perte d'information qui soit FNBC.
2. Il en existe généralement plusieurs et le résultat de l'algorithme dépend de l'ordre que l'on choisit pour traiter les dépendances fonctionnelles ne satisfaisant pas la propriété de Boyce-Codd.
3. La décomposition ne préserve pas nécessairement les dépendances fonctionnelles.

Forme normale de Boyce-Codd

Exemple. On considère la relation

`T(numClient, nomClient, numAppart, adrAppart, dateDeb, dateFin, loyer, numProp, nomProp)`

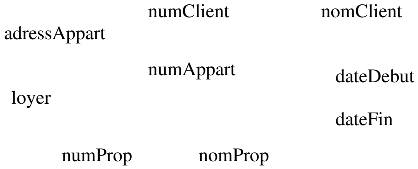
numClient
nomClient
adressAppart
numAppart
dateDebut
loyer
dateFin
numProp
nomProp

Forme normale de Boyce-Codd

Exemple. On considère la relation et les dépendances fonctionnelles suivantes :

$$\texttt{T(numClient, nomClient, numAppart, adrAppart,}$$
$$\texttt{dateDeb, dateFin, loyer, numProp, nomProp)}$$

1. `numClient`→`nomClient` ;
2. (`numClient`, `numAppart`) → `dateDeb`, `dateFin` ;
3. $\texttt{numAppart} \rightarrow \left\{\begin{array}{l}\texttt{adrAppart,}\\ \texttt{loyer,}\\ \texttt{numProp,}\\ \texttt{nomProp}\end{array}\right.$
4. `numProp`→`nomProp`

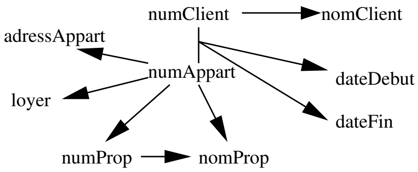


Forme normale de Boyce-Codd

Exemple. On considère la relation et les dépendances fonctionnelles suivantes :

`T(numClient, nomClient, numAppart, adrAppart, dateDeb, dateFin, loyer, numProp, nomProp)`

1. `numClient`→`nomClient` ;
2. (`numClient`, `numAppart`) → `dateDeb`, `dateFin` ;
3. `numAppart`→ { `adrAppart`, `loyer`, `numProp`, `nomProp` }
4. `numProp`→`nomProp`



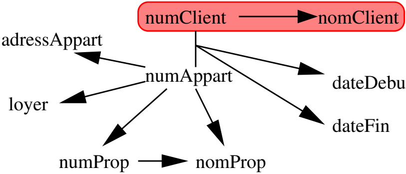
Il n'y a qu'un seul identifiant : (`numClient`, `numAppart`).
Les DFs 1, 3 et 4 ne satisfont pas la propriété voulue.

Forme normale de Boyce-Codd

Exemple. On considère la relation et les dépendances fonctionnelles suivantes :

`T(numClient, nomClient, numAppart, adrAppart, dateDeb, dateFin, loyer, numProp, nomProp)`

1. numClient→nomClient ;
2. (numClient, numAppart) → dateDeb, dateFin ;
3. numAppart→ { adrAppart, loyer, numProp, nomProp }
4. numProp→nomProp



Il n'y a qu'un seul identifiant : (numClient, numAppart).
Les DFs 1, 3 et 4 ne satisfont pas la propriété voulue.

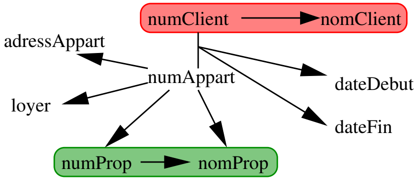
- 1ère étape : DF 1. On obtient R1(numClient, nomClient)

Forme normale de Boyce-Codd

Exemple. On considère la relation et les dépendances fonctionnelles suivantes :

`T(numClient, nomClient, numAppart, adrAppart, dateDeb, dateFin, loyer, numProp, nomProp)`

1. `numClient` → `nomClient` ;
2. `(numClient, numAppart)` → `dateDeb, dateFin` ;
3. `numAppart` → { `adrAppart`, `loyer`, `numProp`, `nomProp` }
4. `numProp` → `nomProp`



Il n'y a qu'un seul identifiant : (`numClient`, `numAppart`).
Les DFs 1, 3 et 4 ne satisfont pas la propriété voulue.

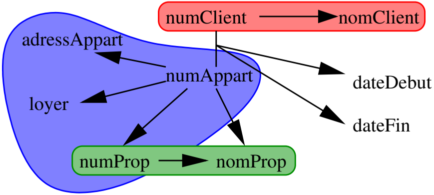
- 1ère étape : DF 1. On obtient `R1(numClient, nomClient)`
- 2ème étape : DF 4. { `R1(numClient, nomClient)`
`R2(numProp, nomProp)`

Forme normale de Boyce-Codd

Exemple. On considère la relation et les dépendances fonctionnelles suivantes :

`T(numClient, nomClient, numAppart, adrAppart,`
`dateDeb, dateFin, loyer, numProp, nomProp)`

1. `numClient` → `nomClient` ;
2. `(numClient, numAppart)` → `dateDeb`, `dateFin` ;
3. `numAppart` → { `adrAppart`, `loyer`, `numProp`, `nomProp` }
4. `numProp` → `nomProp`



Il n'y a qu'un seul identifiant : `(numClient, numAppart)`.
Les DFs 1, 3 et 4 ne satisfont pas la propriété voulue.

- 1ère étape : DF 1. On obtient `R1(numClient, nomClient)`
- 2ème étape : DF 4. { `R1(numClient, nomClient)`
  `R2(numProp, nomProp)`
- 3ème étape : DF 3. { `R1(numClient, nomClient),`
  `R2(numProp, nomProp),`
  `R3(numAppart, adrAppart, loyer, numProp)`
  `T'''(numClient, numAppart, dateD, dateF)`

Forme normale de Boyce-Codd

Exemple. On considère la relation et les dépendances fonctionnelles suivantes :

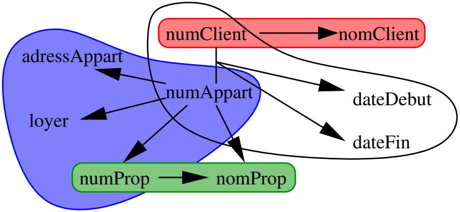
`T(numClient, nomClient, numAppart, adrAppart,`
`dateDeb, dateFin, loyer, numProp, nomProp)`

① `numClient`→`nomClient` ;

② `(numClient, numAppart)` → `dateDeb, dateFin` ;

③ `numAppart`→ { `adrAppart,`
`loyer,`
`numProp,`
`nomProp`

④ `numProp`→`nomProp`



Il n'y a qu'un seul identifiant : `(numClient, numAppart)`.
Les DFs 1, 3 et 4 ne satisfont pas la propriété voulue.

- 1ère étape : DF 1. On obtient `R1(numClient, nomClient)`
- 2ème étape : DF 4. { `R1(numClient, nomClient)`
  `R2(numProp, nomProp)`
- 3ème étape : DF 3. { `R1(numClient, nomClient),`
  `R2(numProp, nomProp),`
  `R3(numAppart, adrAppart, loyer, numProp)`
  `T'''(numClient, numAppart, dateD, dateF)`

En Bref

Graphe de DF

1FN

2FN

3FN

FNBC

En Bref

Graphe de DF

Détermination des identifiants

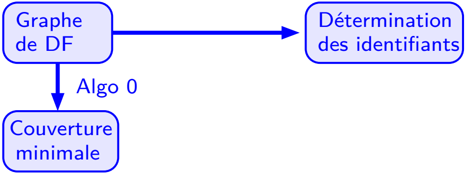
1FN

2FN

3FN

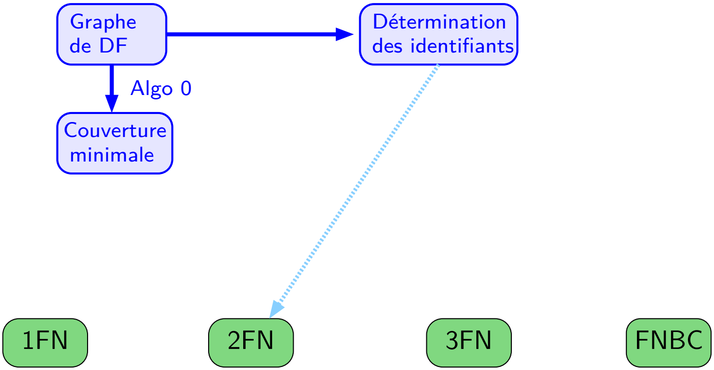
FNBC

En Bref


Graphe de DF
Détermination des identifiants
Algo 0
Couverture minimale


1FN
2FN
3FN
FNBC

En Bref



Graphe de DF

Détermination des identifiants

Algo 0

Couverture minimale

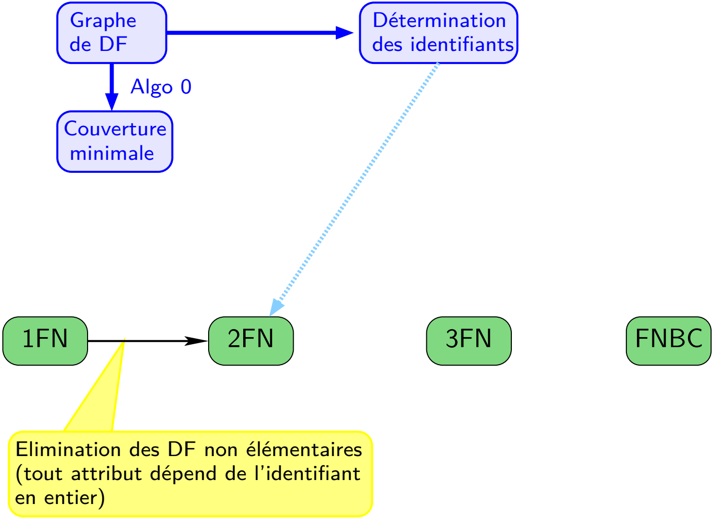
1FN

2FN

3FN

FNBC

En Bref



Graphe de DF

Algo 0

Couverture minimale

Détermination des identifiants

1FN

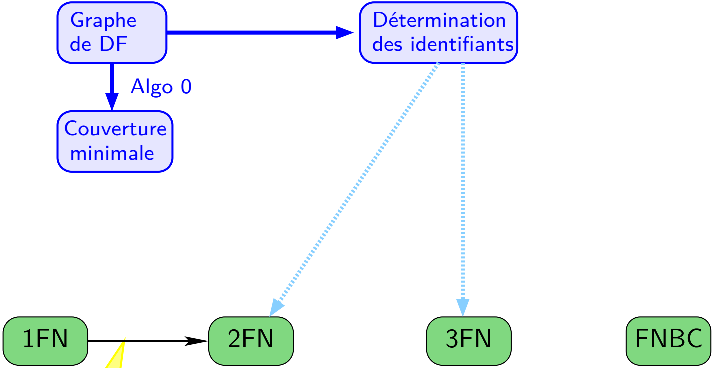
2FN

3FN

FNBC

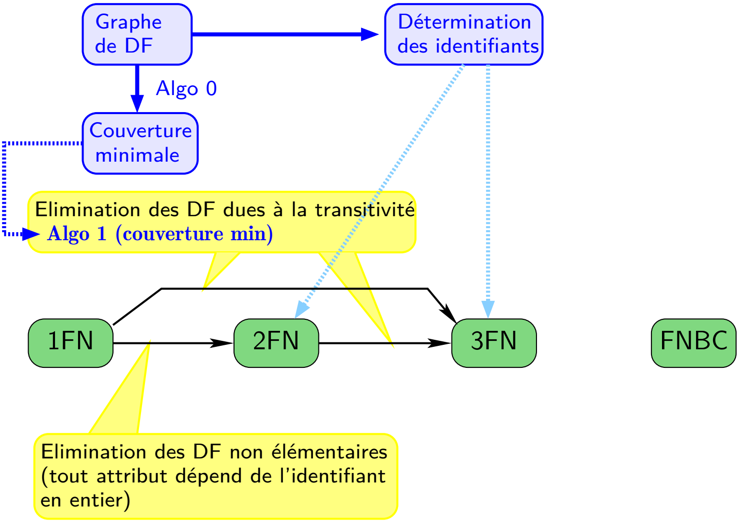
Elimination des DF non élémentaires (tout attribut dépend de l'identifiant en entier)

En Bref



Elimination des DF non élémentaires (tout attribut dépend de l'identifiant en entier)

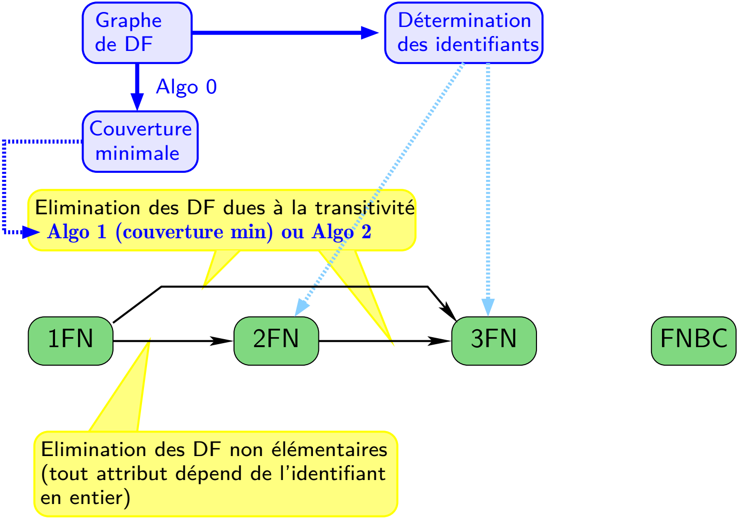
En Bref



Elimination des DF dues à la transitivité **Algo 1 (couverture min)**

Elimination des DF non élémentaires (tout attribut dépend de l'identifiant en entier)

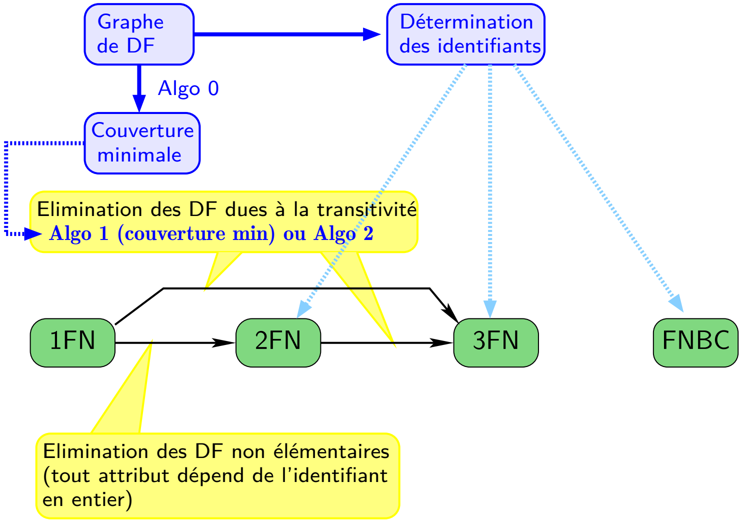
En Bref



Elimination des DF dues à la transitivité
Algo 1 (couverture min) ou Algo 2

Elimination des DF non élémentaires (tout attribut dépend de l'identifiant en entier)

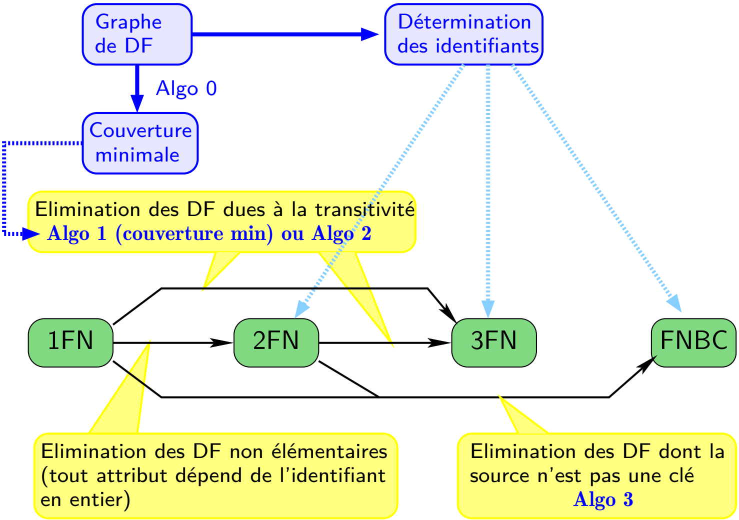
En Bref



Elimination des DF dues à la transitivité
Algo 1 (couverture min) ou Algo 2

Elimination des DF non élémentaires (tout attribut dépend de l'identifiant en entier)

En Bref



Graphe de DF

Détermination des identifiants

Algo 0

Couverture minimale

Elimination des DF dues à la transitivité
Algo 1 (couverture min) ou Algo 2

1FN

2FN

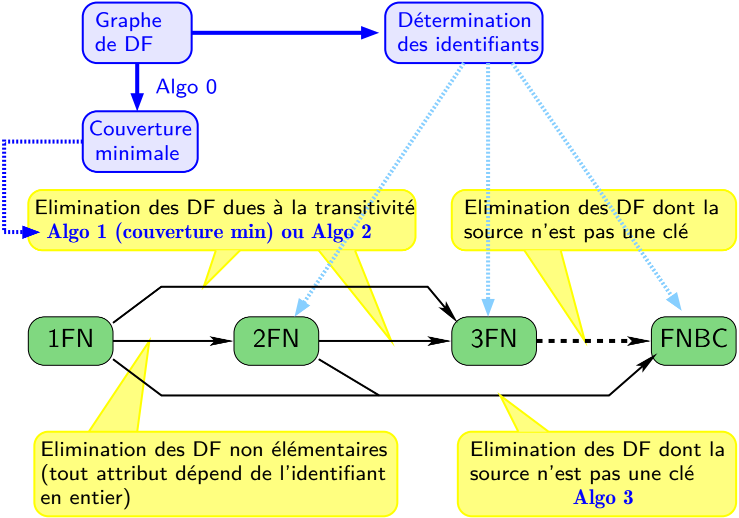
3FN

FNBC

Elimination des DF non élémentaires (tout attribut dépend de l'identifiant en entier)

Elimination des DF dont la source n'est pas une clé
Algo 3

En Bref



Graphe de DF

Détermination des identifiants

Algo 0

Couverture minimale

Elimination des DF dues à la transitivité **Algo 1 (couverture min) ou Algo 2**

Elimination des DF dont la source n'est pas une clé

1FN

2FN

3FN

FNBC

Elimination des DF non élémentaires (tout attribut dépend de l'identifiant en entier)

Elimination des DF dont la source n'est pas une clé **Algo 3**