

Base de données

Chapitre 7 : Normalisation et formes normales

Stéphane Malandain – sbd – 2024

Formes normales

Objectifs :

- Normaliser une base de données
- Etudier les formes normales
- Approches pour l'élaboration d'un schéma de BD

Formes normales

Thèmes abordés :

- Dépendances fonctionnelles (DF)
- Règles de dérivation d'Armstrong
- Fermeture et couvertures minimales
- Perte d'information et perte de dépendances fonctionnelles
- Formes normales et dépendances d'inclusion
- Normalisation par décomposition et par synthèse

Vers la normalisation

Vers la normalisation

Approche descendante (top-down)

Représenter les concepts importants (principales entités, associations. . .) puis, affiner pour obtenir un plus bas niveau de détails (attributs. . .)

- Typiquement: Modèle EA ou procédé par décomposition (normalisation)

Approche ascendante (bottom-up)

A partir des attributs, définir et/ou suivre une systématique qui permet d'établir leurs associations et dépendances jusqu'à dégager les concepts importants

- Typiquement: Normalisation, dépendances fonctionnelles et formes normales

Approche ascendante (bottom-up)

Exemple d'une commande

R (no-di, no-comm, no-article, date, qté, prix-unit, nom, adresse)

Approche ascendante (bottom-up)

Exemple d'une commande



Approche ascendante (bottom-up)

Exemple d'une commande

R1(no-cl, nom, adresse)

R2(no-article, prix-unit)

R3(no-comm, no-cl, date)

R4(no-comm, no-article, qte)



Client(no-cl, nom, adresse)

Article(no-article, prix-unit)

Commande(no-comm, no-cl, date)

Composition(no-comm, no-article, qte)

Procédure par décomposition

Relation

Com(no_cli, no_comm, no_article, date, qté, prix_unit, nom, adresse)

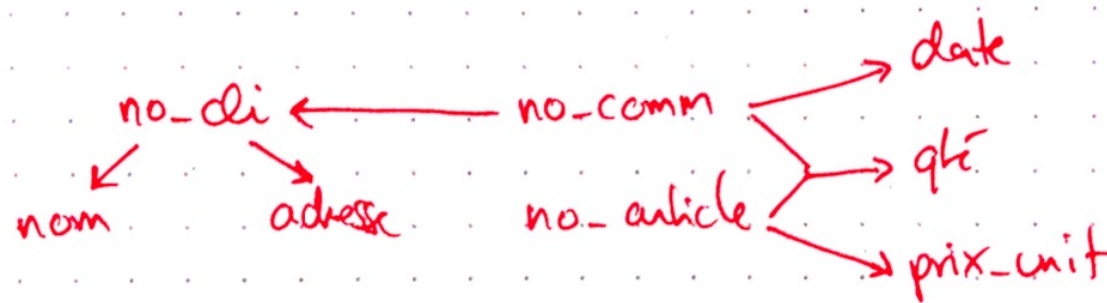
DF

- | | |
|--------------------------------|--|
| 1. no_comm → date | => <i>toute commande a une date</i> |
| 2. no_comm → no_cli | => <i>une commande est émise par un client</i> |
| 3. no_cli → nom | => <i>un client a un nom</i> |
| 4. no_cli → adresse | => <i>un client a une adresse</i> |
| 5. {no_comm, no_article} → qté | => <i>chaque commande a une quantité par article</i> |
| 6. no_article → prix_unit | => <i>un article a un prix</i> |

Graphe des DF

Graphes des attributs et des DF (1/2)

Un **graphe des DF** permet de représenter graphiquement les DF entre attributs.

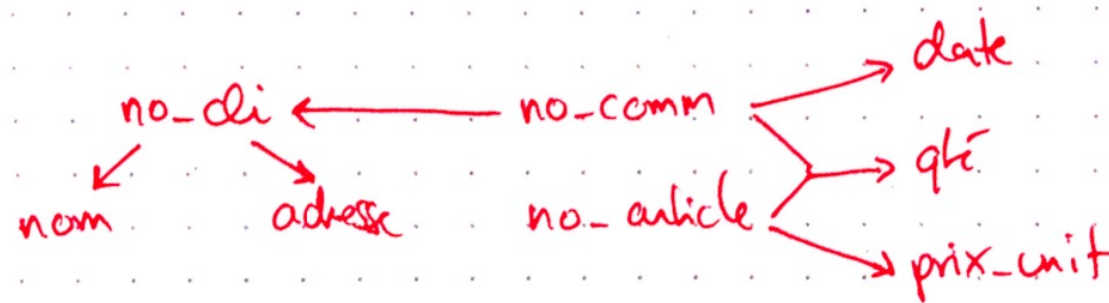


Dépendance fonctionnelle

- Une DF **externe** possède un déterminé qui ne constitue pas un déterminant
- Les autres DF sont dites **internes**.

Graphes des attributs et des DF (2/2)

Un **graphe des DF** permet de représenter graphiquement les DF entre attributs.



Attributs

- Un attribut **externe** est un déterminé strict (nom, adresse, date, qté et prix_unit)
- Un attribut **interne** est un déterminant et un déterminé (no_cli) strict.
- Un attribut **source** est un déterminant mais n'est pas un déterminé (no_comm, no_article)

Règles de dérivation d'Armstrong

Règles de dérivation (règles de Armstrong)

Elles permettent de produire de nouvelles DF à partir de DF connues

1. Réflexivité (DF triviale)

- $A \rightarrow A', A' \subseteq A$
- $A \rightarrow A$ (Tout attribut se détermine lui-même)
- Exemple: $\{\text{no_comm}, \text{no_article}\} \rightarrow \text{no_article}$

2. Augmentation

- $A \rightarrow B$, alors $AX \rightarrow BX$

3. Transitivité

- $A \rightarrow B$ et $B \rightarrow C$, alors $A \rightarrow C$
- Exemple: $\text{no_comm} \rightarrow \text{nom}$

Règles de dérivation

D'autres règles peuvent être déduites à partir des 3 règles de base

4. Décomposition

- $A \rightarrow BC$, alors $A \rightarrow B$ (et $A \rightarrow C$)

- Raisonnement

1. $A \rightarrow BC$

(Donné)

2. $BC \rightarrow B$

(Réflexivité)

3. $A \rightarrow B$

(Transitivité de 1. et 2.)

Règles de dérivation

5. Union

- $A \rightarrow B$ et $A \rightarrow C$, alors $A \rightarrow BC$

- Raisonnement

| | |
|------------------------|----------------------------|
| 1. $A \rightarrow B$ | (Donné) |
| 2. $A \rightarrow C$ | (Donné) |
| 3. $AA \rightarrow AB$ | (Augmentation de 1.) |
| 4. $A \rightarrow AB$ | (Simplification de 3.) |
| 5. $AB \rightarrow CB$ | (Augmentation de 2.) |
| 6. $A \rightarrow CB$ | (Transitivité de 4. et 5.) |

Règles de dérivation

6. Pseudo-transitivité

- $A \rightarrow B$ et $BC \rightarrow D$, alors $AC \rightarrow D$

- Raisonnement

- | | |
|------------------------|-------------------------------|
| 1. $A \rightarrow B$ | (Donné) |
| 2. $BC \rightarrow D$ | (Donné) |
| 3. $AC \rightarrow BC$ | (Augmentation de 1.) |
| 4. $AC \rightarrow D$ | (Transitivité de 3. et de 2.) |

DF élémentaire (DFE)

Soit G un groupe d'attribut et A un attribut, une DF $G \rightarrow A$ est élémentaire si aucun sous-ensemble du déterminant G ne peut déterminer A .

- $AB \rightarrow C$ est élémentaire si ni A , ni B détermine individuellement C
- $\{no_comm, no_prod\} \rightarrow qté$ est élémentaire
- $\{no_comm, no_prod\} \rightarrow no_cli$ n'est pas élémentaire
- $AB \rightarrow CD$ n'est pas élémentaire (CD est un groupe d'attributs)

Fermeture

Fermeture de DF

Une fermeture F^+ d'un ensemble F de DFE est l'ensemble de toutes les DFE qui peuvent être déduites de F en appliquant les règles de dérivation.

Exemple

- $R(A, B, C, D, E)$

- $F = \{ A \rightarrow B, B \rightarrow C, CD \rightarrow E \}$

- DF Dérivées :

- $A \rightarrow C$

- $AD \rightarrow E$

- $F^+ = \{ A \rightarrow B, B \rightarrow C, CD \rightarrow E, A \rightarrow C, AD \rightarrow E \}$

Fermeture d'un ensemble d'attributs

Une fermeture K^+ d'un ensemble K d'attributs d'une relation R est l'ensemble des attributs déterminés par K en appliquant chaque DF de R . C'est utile pour déterminer une clé minimale.

Exemple

- $R(A, B, C, D, E)$
- $F = \{A \rightarrow B, B \rightarrow C, CD \rightarrow E\}$
- $K = \{A, D\}$

$$K^+ = \{A, D\}$$

$$K^+ = \{A, B, D\} \quad (DF1)$$

$$K^+ = \{A, B, C, D\} \quad (DF2)$$

$$K^+ = \{A, B, C, D, E\} \quad (DF3)$$

Preuve de clé minimale : Pour prouver que $\{A, D\}$ est minimal, il faut calculer le K^+ pour $\{A\}$ et K^+ pour $\{B\}$ et ensuite montrer qu'ils sont différents de R

Couverture et clés minimales

Couverture et clés minimales

La couverture minimale F_{min} de F est le plus petit ensemble de DF à partir duquel il est possible de reconstituer F tel que $F_{min}^+ = F^+$.

Cette couverture est utile pour **prouver** qu'une décomposition est **sans perte de dépendances fonctionnelles**. Il peut exister plusieurs couvertures minimales.

Couverture minimale

Déterminer une couverture minimale:

- En **supprimant** toutes les DF **non élémentaires** et **déduites** tout en gardant le même F^+
- Regroupement des DF selon le même déterminant
- F_{min} de Com :
 - `no_comm` → `date`, `no_cli`
 - `no_cli` → `nom`, `adresse`
 - `{no_comm, no_article}` → `qté`
 - `no_article` → `prix_unit`

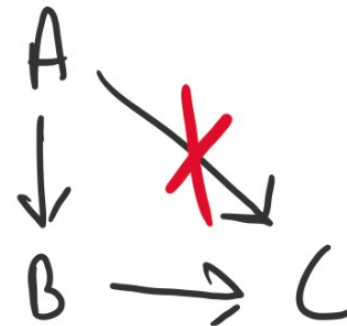
Couverture minimale

A l'aide d'un graphe des DF

- Suppression des DF non élémentaires



- Suppression des DF déduites



Clés minimales

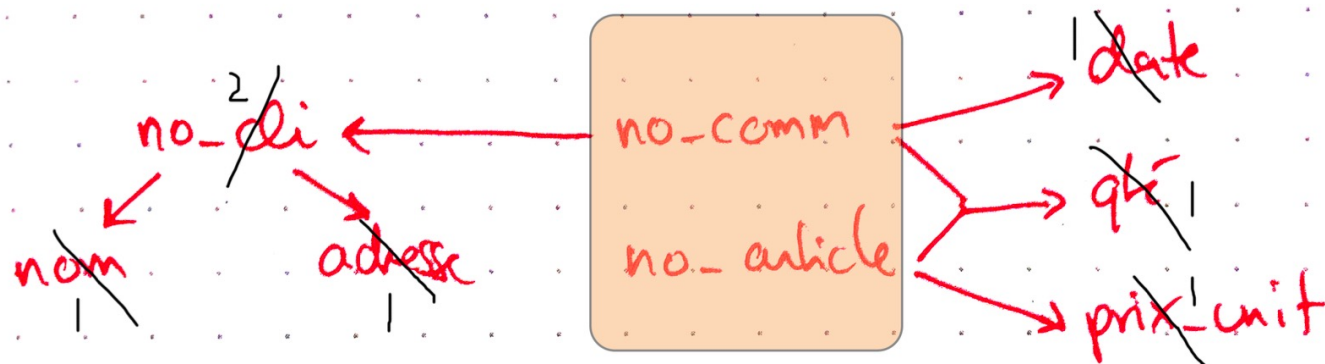
- Tout identifiant d'une relation détermine chaque attribut de cette relation.
- Il est important de déterminer la clé minimale K_{min} qui définit tous les autres attributs.
- K_{min}^+ doit retourner l'ensemble des attributs. *min*
- Une relation $R(A, B, C, D)$ a comme clé (non minimale) $K = (A, B, C, D)$.
On dit que R est une "super clé".

Clés minimales

- Définir la clé minimale d'un **schéma de base de données** à l'aide du graphe AF sans circuit:
 - On retire de K les attributs externes
 - On répète jusqu'à ce qu'il ne soit plus possible de retirer les

Attributs

- Il reste généralement les attributs sources



Clés minimales

Si la procédure précédente abouti à un circuit (appelé noyau irréductible), la relation comporte plusieurs clés

- Pour chaque attribut du circuit, le retirer du noyau avec les DF associées
- On obtient des sous-graphes signifiant les clés
- On répète la procédure tant qu'il y a un circuit

Clés minimales : Exemple (1/3)

Exemple

Attribution(no_etu, instrument, no_prof, nom_etu, nom_prof)

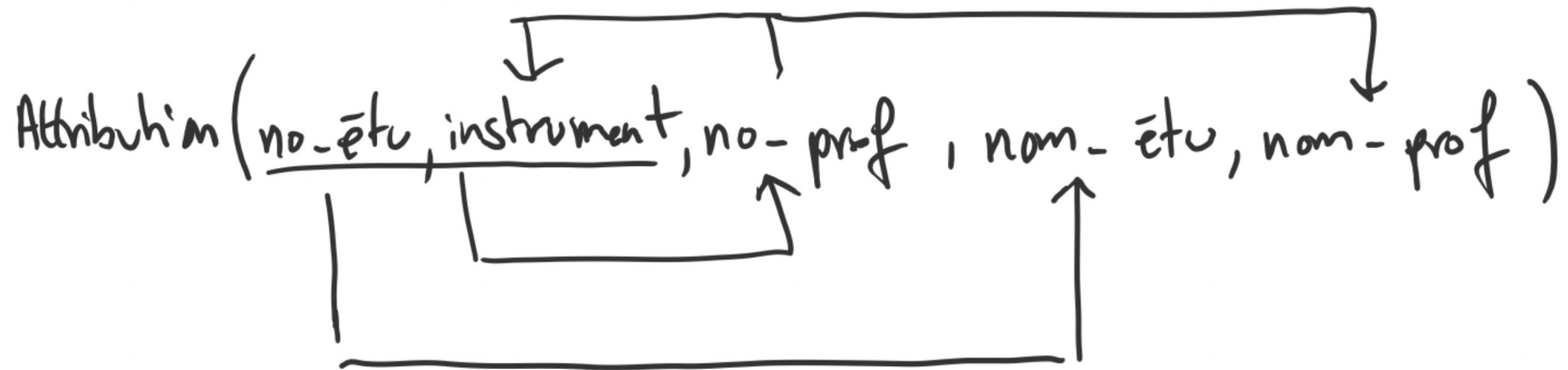


Figure 1: Schéma avec DF

Clés minimales : Exemple (2/3)

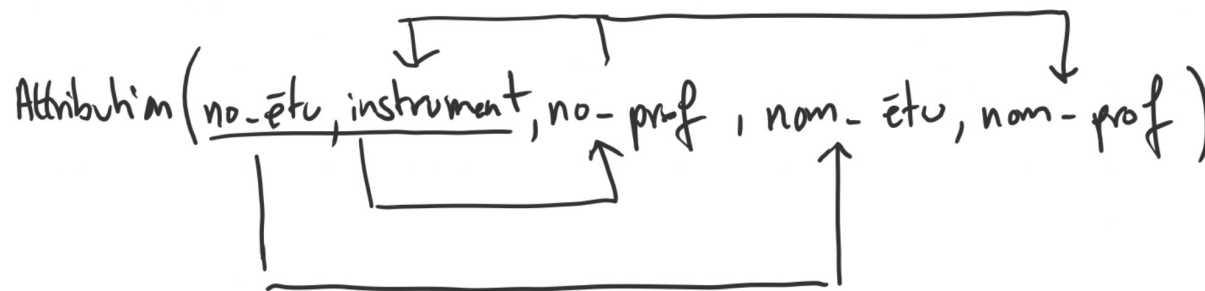


Figure 2: Schéma avec DF

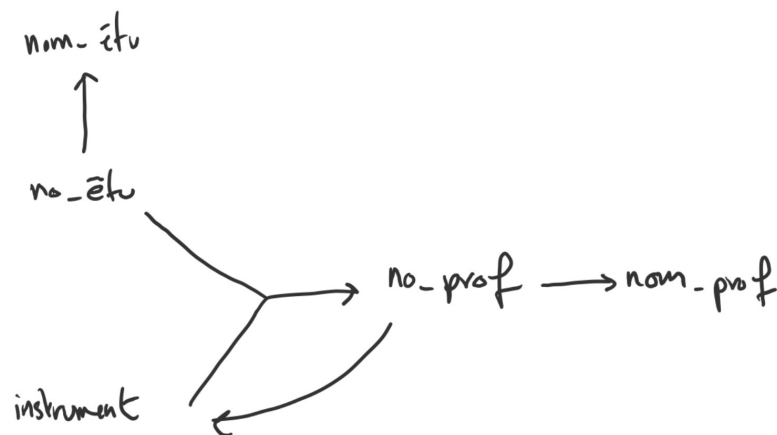
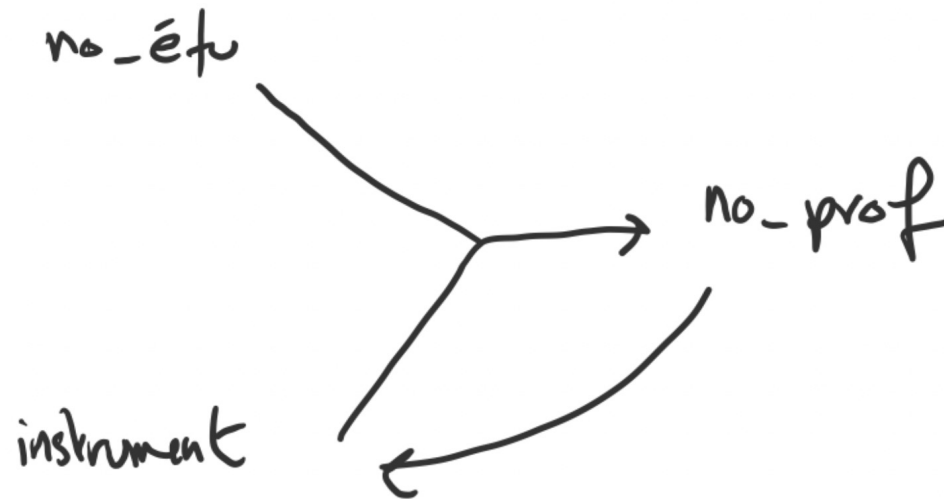


Figure 3: Schéma avec DF

Clés minimales : Exemple (1/3)

Graphe minimal des DF



Attributs du circuit: `no_instrument` et `no_prof`

- En enlevant `no_instrument` il reste la clé `{no_etu, no_prof}` (clé 1)
- En enlevant `no_prof` il reste la clé `{no_etu, no_instrument}` (clé 2)

Perte d'informations et de dépendances fonctionnelles

Décomposition sans perte d'informations

Une décomposition d'un schéma de relation R en un ensemble de relations R_1, R_2, \dots, R_n est sans perte d'informations si

$$R = R_1 \bowtie R_2 \bowtie \dots \bowtie R_n$$

Décomposition sans perte d'informations

Exemple de décomposition SPI

- R

| no_vehicule | type | marque | couleur |
|-------------|---------|---------|---------|
| 1 | Model X | Tesla | noir |
| 2 | Model X | Tesla | rouge |
| 8 | 911 | Porsche | noir |

Décomposition sans perte d'informations

Exemple de décomposition SPI

R1

| no_vehicule | type | couleur |
|-------------|---------|---------|
| 1 | Model X | noir |
| 2 | Model X | rouge |
| 8 | 911 | noir |

R2

| type | marque |
|---------|---------|
| Model X | Tesla |
| 911 | Porsche |

Décomposition sans perte d'informations

Exemple de décomposition SPI

$$R_1 \bowtie R_2 = R$$

| no_vehicule | type | marque | couleur |
|-------------|---------|---------|---------|
| 1 | Model X | Tesla | noir |
| 2 | Model X | Tesla | rouge |
| 8 | 911 | Porsche | noir |

Décomposition sans perte d'informations

Exemple de décomposition avec perte d'information

R1

| no_vehicule | type |
|-------------|---------|
| 1 | Model X |
| 2 | Model X |
| 8 | 911 |

R2

| type | couleur |
|---------|---------|
| Model X | noir |
| Model X | rouge |
| 911 | noir |

R3

| type | marque |
|---------|---------|
| Model X | Tesla |
| 911 | Porsche |

Décomposition sans perte d'informations

Exemple de décomposition avec perte d'information

$$R_1 \bowtie R_2 \bowtie R_3 \neq R$$

| no_vehicule | type | marque | couleur |
|-------------|---------|---------|---------|
| 1 | Model X | Tesla | noir |
| 1 | Model X | Tesla | rouge |
| 2 | Model X | Tesla | noir |
| 2 | Model X | Tesla | rouge |
| 8 | 911 | Porsche | noir |

Décomposition sans perte de DF

Une décomposition d'un schéma de relation R est **sans perte de dépendances fonctionnelles** si la fermeture F^+ de R est égale à l'union des fermetures de R_1, R_2, \dots, R_n :

$$F^+(R) = F^+(R_1) \cup F^+(R_2) \cup \dots \cup F^+(R_n)$$

La normalisation

Formes normales

Normaliser une relation est un processus qui consiste à décomposer une relation en relations plus petites.

- Sans redondances
- Sans pertes d'informations (SPI)
- Sans pertes de dépendances fonctionnelles (SPDF)

Formes normales

Les formes normales permettent une mesure qualitative du niveau de normalisation d'une relation.

Il existe 6 principales FN:

- la première forme normale (1FN)
- la deuxième forme normale (2FN)
- la troisième forme normale (3FN)
- la forme normale de Boyce-Codd (FNBC)
- la quatrième forme normale (4FN)
- la cinquième forme normale (5FN)

Formes normales

- La troisième (éventuellement FNBC) est généralement l'objectif minimal
- Les quatre premières reposent sur les DF monovaluées
- 4FN et 5FN reposent respectivement sur les dépendances multivaluées et sur les jointures (hors cours)

Relation dénormalisée

Une relation qui ne respecte pas la première forme normale est **dénormalisée**.

Typiquement:

- Bases de données NoSQL
- Structures récursives ou possédant des listes (JSON, classes. . .)

Première forme normale (1FN)

Une relation est en première forme normale si tous ses attributs sont atomiques (non multivalués)

Conséquence :

- Pas de listes

Première forme normale (1FN)

Exemple non 1FN

| no_étudiant | nom | prenom | tags |
|-------------|--------|-----------|------------------|
| 1 | Dupont | Marc | sympa, paresseux |
| 2 | Burrix | Steven | assidu |
| 3 | Racloz | Stéphanie | assidu, sympa |

Première forme normale (1FN)

Décomposition 1FN

| no_étudiant | nom | prenom |
|-------------|--------|-----------|
| 1 | Dupont | Marc |
| 2 | Burrix | Steven |
| 3 | Racloz | Stéphanie |

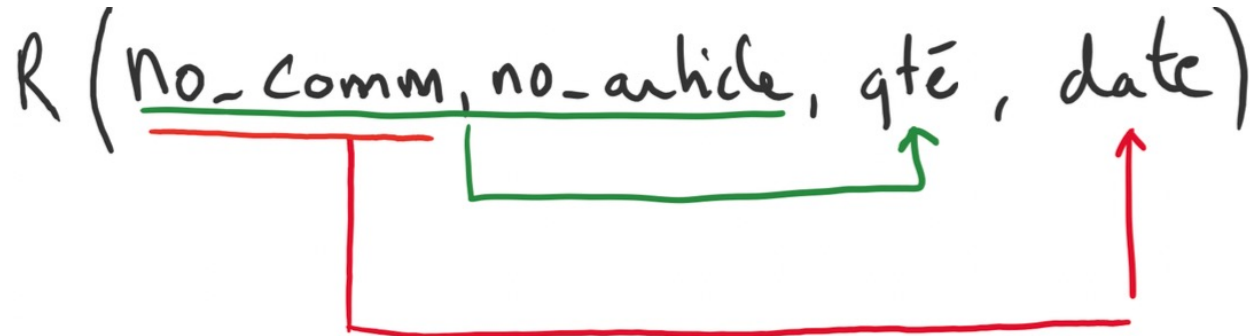
| no_étudiant | tags |
|-------------|-----------|
| 1 | sympa |
| 1 | paresseux |
| 2 | assidu |
| 3 | assidu |
| 3 | sympa |

Deuxième forme normale (2FN)

Une relation est en deuxième forme normale si :

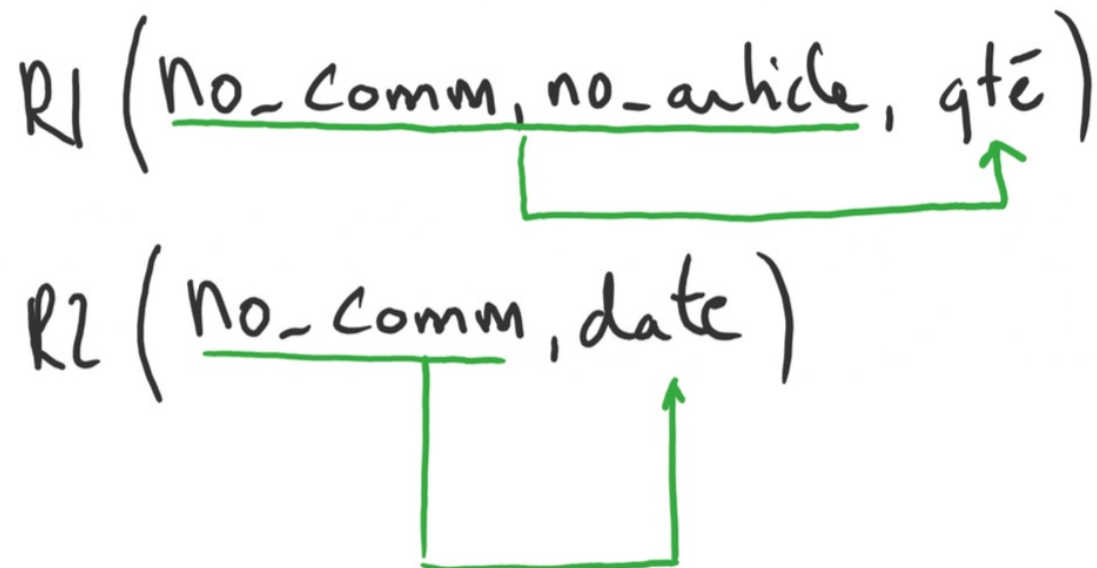
- Elle est en 1FN
- Aucun attribut non-clé ne dépend d'une partie de la clé

Exemple non 2FN



Deuxième forme normale (2FN)

Décomposition 2FN

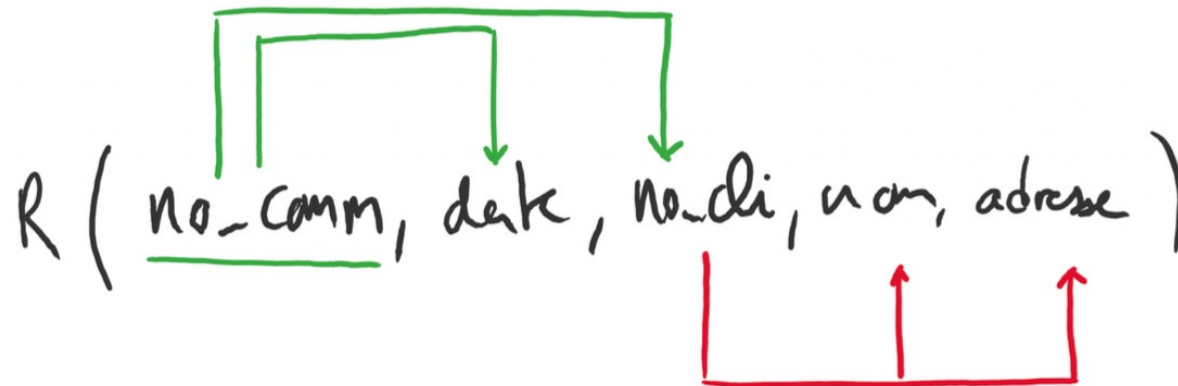


Troisième forme normale (3FN)

Une relation est en troisième forme normale si :

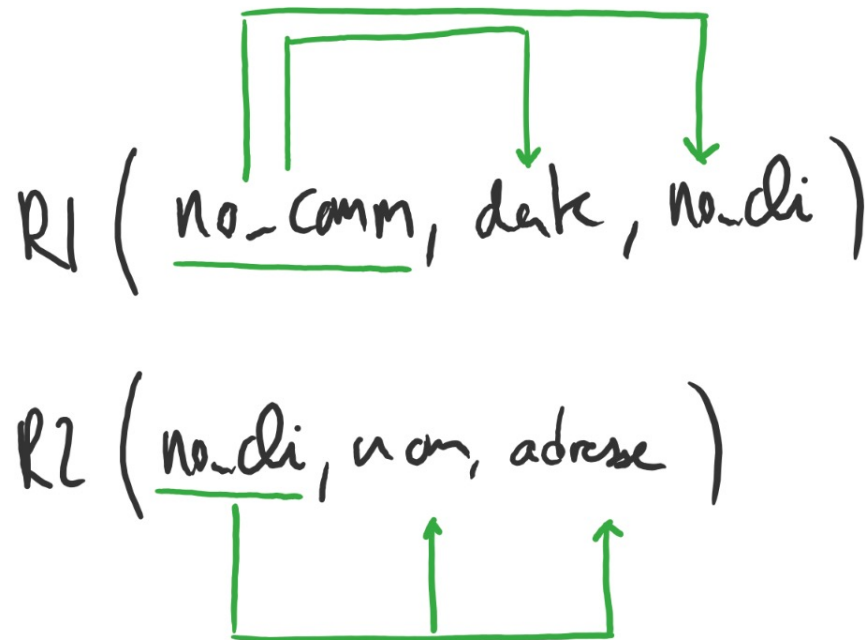
- Elle est en 2FN
- Aucun attribut non-clé ne dépend d'un attribut non-clé

Exemple non 2FN



Troisième forme normale (2FN)

Décomposition 3FN

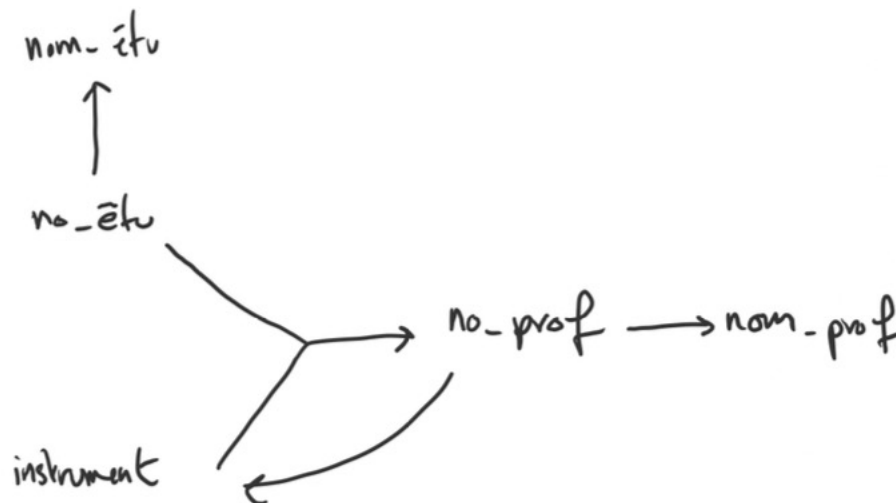


Forme normale de Boyce-Codd (FNBC)

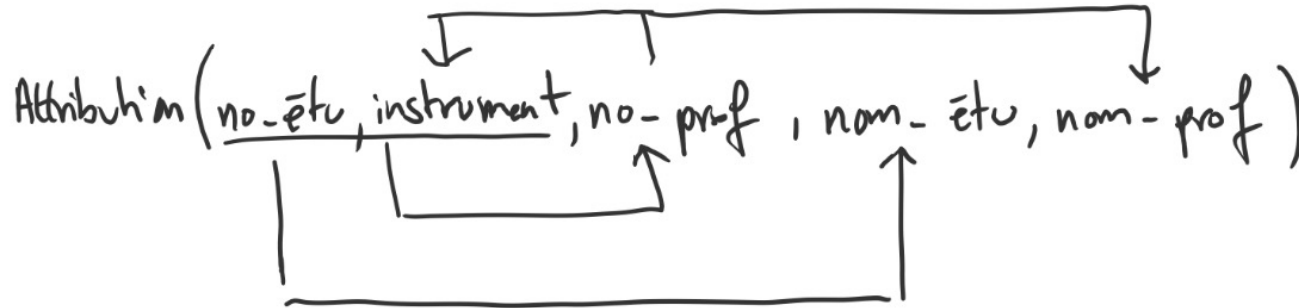
La forme normale de Boyce-Codd (FNBC) intervient lors de DF cycliques

Une relation est en forme normale de Boyce-Codd si :

- Elle est en 3FN
- Aucun attribut non-clé n'est la source d'une dépendance fonctionnelle vers une partie de la clé.



Forme normale de Boyce-Codd (FNBC)



```

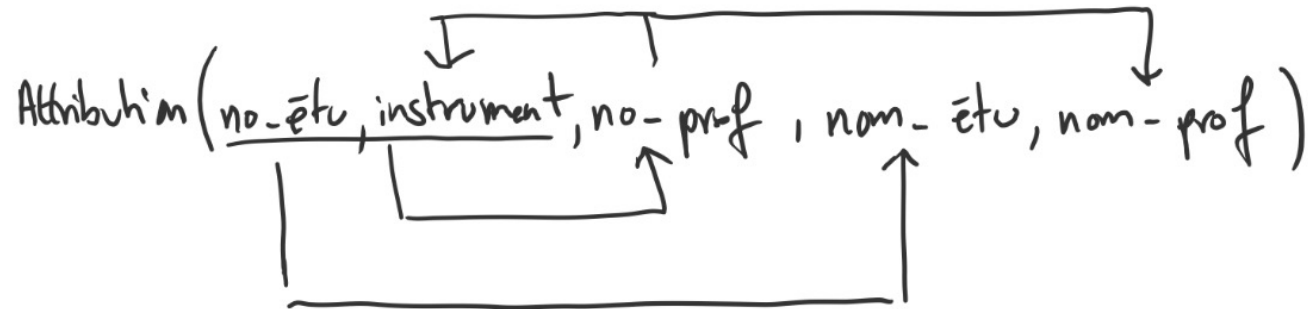
Etudiant(no_etu, nom_etu)
Prof(no_prof, nom_prof)
Attribution(no_etu, instrument,
no_prof),
  no_etu ⊆ Etudiant.no_etu
  no_prof ⊆ Prof.no_prof
DF: no_prof → instrument
  
```

Solution en 3FN "classique"

- La DF `no_prof → instrument` n'est pas modélisable à l'aide d'une relation
- Solution acceptable à condition que la DF soit vérifiée à l'aide d'un trigger

Forme normale de Boyce-Codd (FNBC)

Décomposition FNBC



Etudiant(no_etu, nom_etu)

Prof(no_prof, nom_prof, instrument)

Attribution(no_etu, no_prof), $\text{no_prof} \subseteq \text{Prof.no_prof}$

DF : $\{\text{no_etu}, \text{instrument}\} \rightarrow \text{no_prof}$

- Cette fois la DF $\{\text{no_etu}, \text{instrument}\} \rightarrow \text{no_prof}$ doit être vérifiée sur Attribution \bowtie no Prof

La dénormalisation

La dénormalisation

Il s'agit du procédé inverse. Il implique une redondance qui permet une efficacité des requêtes (éviter des jointures par ex.), d'être plus proche d'une sérialisation (JSON, XML. . .) ou d'une logique de POO.

- Ce processus est maîtrisé et intervient après une normalisation
- Il est nécessaire d'identifier où se trouve les besoins en performance (généralement connu en milieu ou en fin de développement)

Normalisation par décomposition

Technique de décomposition

1. Partir d'un schéma incluant **tous les attributs**
2. Identifier une DF qui ne respecte pas une FN (3FN)
3. Décomposer la DF problématique en créant un nouveau schéma
4. Répétez les points 2 et 3 tant que le schéma n'est pas normalisé

Technique de décomposition

1FN

Regrouper les répétitions dans une table

Technique de décomposition

2FN

$$R(\underbrace{K_1, K_2}_{\uparrow}, X, Y) \Rightarrow R_1(\underbrace{K_1, K_2, X}_{\downarrow}, K_2 \subseteq R_2 \cdot K_2) \\ R_2(\underbrace{K_2, Y}_{\downarrow})$$

Technique de décomposition

3FN

$$R(\underline{K}, X, Y, Z) \Rightarrow \begin{matrix} R_1(\underline{K}, X, Y) \\ R_2(\underline{Y}, Z) \end{matrix}, Y \subseteq R_2.Y$$

$$R(\underline{K_1}, K_2, X, Y) \Rightarrow R(\underline{K_1, K_2}, X, Y)$$

avec DF: $X \rightarrow K_2$

Technique de décomposition

FNBC

$$R \left(\underbrace{K_1, K_2}_{\text{DF}}, X, Y \right) \Rightarrow \begin{array}{l} R_1(\underline{K_1, X}, Y) \\ R_2(\underline{X}, K_2) \end{array}, X \subseteq R_2.X$$

avec DF: $\{K_1, K_2\} \rightarrow X, Y$ sur R_1 et R_2

Normalisation par synthèse

Normalisation par synthèse

- Déterminer les schémas à l'aide des DF
- Regroupement sémantique des DF selon le même déterminant
 $\text{no_client} \rightarrow \text{nom}$ et $\text{no_client} \rightarrow \text{prenom}$ devient
 $\text{no_client} \rightarrow \{\text{nom}, \text{prenom}\}$
- Chaque DF devient un schéma de relation
- Cette technique produit toujours un schéma en 3FN

Récapitulation

Récapitulation

- Nouvelles approches pour élaborer un schéma de BD
 - approche descendante : décomposition basée sur les FN
 - approche ascendante : méthode par synthèse
 - outils précieux pour concevoir une BD
- Normalisation permet d'éviter toutes anomalies