

# Bases de données

## Cours 6

### Dépendances fonctionnelles et formes normales

Marie Pelleau & Laurent Tichit

[marie.pelleau@univ-cotedazur.fr](mailto:marie.pelleau@univ-cotedazur.fr),

[laurent.tichit@univ-cotedazur.fr](mailto:laurent.tichit@univ-cotedazur.fr)

6 décembre 2022

- 1 Conception : dépendances fonctionnelles
  - Motivation
  - Définition
  - Propriétés formelles
  - Notion de clé
- 2 Conception : formes normales
- 3 Autres SGBD, programmer avec un SGBD

# Analyse des problèmes d'un schéma relationnel

Commande(numprod, quantité, numfour, adressefour)

numprod	quantité	numfour	adressefour
101	300	901	Cours de l'Intendance, Bordeaux
104	280	902	Rue du Port, Clermont-Ferrand
112	170	904	Boulevard Joseph Garnier, Nice
103	250	901	Cours de l'Intendance, Bordeaux

# Analyse des problèmes d'un schéma relationnel

Commande(numprod, quantité, numfour, adressefour)

numprod	quantité	numfour	adressefour
101	300	901	Cours de l'Intendance, Bordeaux
104	280	902	Rue du Port, Clermont-Ferrand
112	170	904	Boulevard Joseph Garnier, Nice
103	250	901	Cours de l'Intendance, Bordeaux

- Quelles anomalies pour cette relation ?

# Analyse des problèmes d'un schéma relationnel

Commande(numprod, quantité, numfour, adressefour)

numprod	quantité	numfour	adressefour
101	300	901	Cours de l'Intendance, Bordeaux
104	280	902	Rue du Port, Clermont-Ferrand
112	170	904	Boulevard Joseph Garnier, Nice
103	250	901	Cours de l'Intendance, Bordeaux

- Quelles anomalies pour cette relation ?
  - Redondances

# Analyse des problèmes d'un schéma relationnel

Commande(numprod, quantité, numfour, adressefour)

numprod	quantité	numfour	adressefour
101	300	901	Cours de l'Intendance, Bordeaux
104	280	902	Rue du Port, Clermont-Ferrand
112	170	904	Boulevard Joseph Garnier, Nice
103	250	901	Cours de l'Intendance, Bordeaux

- Quelles anomalies pour cette relation ?
  - Redondances
    - Anomalies de modification
    - Anomalies d'insertion
    - Anomalies de suppression

# Dépendance fonctionnelle

- $R(A_1, A_2, \dots, A_n)$  un schéma relationnel,  $X, Y$  des sous-ensembles de  $\{A_1, A_2, \dots, A_n\}$ .
- On dit que  $X$  détermine  $Y$ , ou que  $Y$  dépend fonctionnellement de  $X$  s'il existe une fonction qui à partir de toute valeur de  $X$  détermine une valeur unique de  $Y$ .

# Dépendance fonctionnelle

- $R(A_1, A_2, \dots, A_n)$  un schéma relationnel,  $X, Y$  des sous-ensembles de  $\{A_1, A_2, \dots, A_n\}$ .
- On dit que  $X$  détermine  $Y$ , ou que  $Y$  dépend fonctionnellement de  $X$  s'il existe une fonction qui à partir de toute valeur de  $X$  détermine une valeur unique de  $Y$ .
- Plus formellement,  $X$  détermine  $Y$  si pour tous  $n$ -uplets  $u_1, u_2$  de  $R$ ,  
$$\pi_X(u_1) = \pi_X(u_2) \implies \pi_Y(u_1) = \pi_Y(u_2).$$



# Dépendance fonctionnelle

- $R(A_1, A_2, \dots, A_n)$  un schéma relationnel,  $X, Y$  des sous-ensembles de  $\{A_1, A_2, \dots, A_n\}$ .
- On dit que  $X$  détermine  $Y$ , ou que  $Y$  dépend fonctionnellement de  $X$  s'il existe une fonction qui à partir de toute valeur de  $X$  détermine une valeur unique de  $Y$ .
- Plus formellement,  $X$  détermine  $Y$  si pour tous  $n$ -uplets  $u_1, u_2$  de  $R$ ,  
$$\pi_X(u_1) = \pi_X(u_2) \implies \pi_Y(u_1) = \pi_Y(u_2).$$
- On note  $X \longrightarrow Y$ .

# Dépendance fonctionnelle

- $R(A_1, A_2, \dots, A_n)$  un schéma relationnel,  $X, Y$  des sous-ensembles de  $\{A_1, A_2, \dots, A_n\}$ .
- On dit que  $X$  détermine  $Y$ , ou que  $Y$  dépend fonctionnellement de  $X$  s'il existe une fonction qui à partir de toute valeur de  $X$  détermine une valeur unique de  $Y$ .
- Plus formellement,  $X$  détermine  $Y$  si pour tous  $n$ -uplets  $u_1, u_2$  de  $R$ ,  
$$\pi_X(u_1) = \pi_X(u_2) \implies \pi_Y(u_1) = \pi_Y(u_2).$$
- On note  $X \longrightarrow Y$ .

## Exemple

Le numéro d'étudiant permet de retrouver

- le nom d'un étudiant :  $\text{numétudiant} \longrightarrow \text{nom}$
- et même le prénom, la date de naissance :

$\text{numétudiant} \longrightarrow \text{nom, prénom, datenaissance}$

# Dépendances fonctionnelles composées

- Pour une dépendance fonctionnelle  $X \longrightarrow Y$ ,  $X$  peut être formé de plusieurs attributs.

# Dépendances fonctionnelles composées

- Pour une dépendance fonctionnelle  $X \longrightarrow Y$ ,  $X$  peut être formé de plusieurs attributs.

## Exemple

- Un cours est suivi par un ou plusieurs étudiants; un étudiant suit un ou plusieurs cours; il a une note pour chacun.

`numétudiant, numcours  $\longrightarrow$  note`

# Dépendances fonctionnelles composées

- Pour une dépendance fonctionnelle  $X \longrightarrow Y$ ,  $X$  peut être formé de plusieurs attributs.

## Exemple

- Un cours est suivi par un ou plusieurs étudiants; un étudiant suit un ou plusieurs cours; il a une note pour chacun.

`numétudiant, numcours  $\longrightarrow$  note`

- Lorsqu'un abonné emprunte un livre, on mémorise la date d'emprunt et la date de retour.

`isbn, numabo  $\longrightarrow$  dateemp, dateret`

# Propriétés des dépendances fonctionnelles : axiomes d'Armstrong

- Les dépendances fonctionnelles obéissent à des propriétés élémentaires (appelées axiomes d'Armstrong).

# Propriétés des dépendances fonctionnelles : axiomes d'Armstrong

- Les dépendances fonctionnelles obéissent à des propriétés élémentaires (appelées axiomes d'Armstrong).
- **Réflexivité** :
  - Si  $X$  est un ensemble d'attributs de  $R$  et  $Y \subseteq X$ , alors  $X \longrightarrow Y$ .
  - En particulier, pour tout  $X$ ,  $X \longrightarrow X$ .

# Propriétés des dépendances fonctionnelles : axiomes d'Armstrong

- Les dépendances fonctionnelles obéissent à des propriétés élémentaires (appelées axiomes d'Armstrong).
- **Réflexivité** :
  - Si  $X$  est un ensemble d'attributs de  $R$  et  $Y \subseteq X$ , alors  $X \longrightarrow Y$ .
  - En particulier, pour tout  $X$ ,  $X \longrightarrow X$ .
- **Augmentation** :
  - Si  $X$  détermine  $Y$ , alors  $(X, Z)$  détermine  $(Y, Z)$  pour tout ensemble d'attributs  $Z$ .
  - Autrement dit, si  $X \longrightarrow Y$ , alors pour tout  $Z$ , on a  $(X, Z) \longrightarrow (Y, Z)$ .



# Propriétés des dépendances fonctionnelles : axiomes d'Armstrong

- Les dépendances fonctionnelles obéissent à des propriétés élémentaires (appelées axiomes d'Armstrong).
- **Réflexivité** :
  - Si  $X$  est un ensemble d'attributs de  $R$  et  $Y \subseteq X$ , alors  $X \longrightarrow Y$ .
  - En particulier, pour tout  $X$ ,  $X \longrightarrow X$ .
- **Augmentation** :
  - Si  $X$  détermine  $Y$ , alors  $(X, Z)$  détermine  $(Y, Z)$  pour tout ensemble d'attributs  $Z$ .
  - Autrement dit, si  $X \longrightarrow Y$ , alors pour tout  $Z$ , on a  $(X, Z) \longrightarrow (Y, Z)$ .
- **Transitivité** :
  - Si  $X$  détermine  $Y$  et  $Y$  détermine  $Z$ , alors  $X$  détermine  $Z$ .
  - Autrement dit, si  $X \longrightarrow Y$  et  $Y \longrightarrow Z$ , alors  $X \longrightarrow Z$ .

# Conséquences des axiomes d'Armstrong

- On peut déduire des propriétés supplémentaires à partir des axiomes d'Armstrong.
  - Si  $X \longrightarrow Y$  et  $X \longrightarrow Z$ , alors  $X \longrightarrow Y, Z$ .
  - Si  $X \longrightarrow Y, Z$ , alors  $X \longrightarrow Y$  et  $X \longrightarrow Z$ .
  - Si  $X \longrightarrow Y$  et  $Z \longrightarrow T$ , alors  $X, Z \longrightarrow Y, T$ .
  - Si  $X \longrightarrow Y, Z$  et  $Z \longrightarrow T$ , alors  $X \longrightarrow Y, T$ .

# Conséquences des axiomes d'Armstrong

- On peut déduire des propriétés supplémentaires à partir des axiomes d'Armstrong.
  - Si  $X \longrightarrow Y$  et  $X \longrightarrow Z$ , alors  $X \longrightarrow Y, Z$ .
  - Si  $X \longrightarrow Y, Z$ , alors  $X \longrightarrow Y$  et  $X \longrightarrow Z$ .
  - Si  $X \longrightarrow Y$  et  $Z \longrightarrow T$ , alors  $X, Z \longrightarrow Y, T$ .
  - Si  $X \longrightarrow Y, Z$  et  $Z \longrightarrow T$ , alors  $X \longrightarrow Y, T$ .
- Les règles déduites des axiomes d'Armstrong sont des règles de dépendance fonctionnelle correctes.

## Conséquences des axiomes d'Armstrong

- On peut déduire des propriétés supplémentaires à partir des axiomes d'Armstrong.
  - Si  $X \longrightarrow Y$  et  $X \longrightarrow Z$ , alors  $X \longrightarrow Y, Z$ .
  - Si  $X \longrightarrow Y, Z$ , alors  $X \longrightarrow Y$  et  $X \longrightarrow Z$ .
  - Si  $X \longrightarrow Y$  et  $Z \longrightarrow T$ , alors  $X, Z \longrightarrow Y, T$ .
  - Si  $X \longrightarrow Y, Z$  et  $Z \longrightarrow T$ , alors  $X \longrightarrow Y, T$ .
- Les règles déduites des axiomes d'Armstrong sont des règles de dépendance fonctionnelle correctes.
- L'application répétée des axiomes d'Armstrong à partir d'un ensemble de dépendances fonctionnelles bien choisi va nous permettre de trouver toutes les dépendances fonctionnelles qui nous intéressent.

# Dépendances fonctionnelles élémentaires

## Définition

On dit qu'une dépendance fonctionnelle  $X \longrightarrow A$  est **élémentaire** (notée DFE) si

- $A$  est un attribut non inclus dans  $X$
- et s'il n'existe pas de  $Y \subsetneq X$  tel que  $Y \longrightarrow A$ .

# Dépendances fonctionnelles élémentaires

## Définition

On dit qu'une dépendance fonctionnelle  $X \longrightarrow A$  est **élémentaire** (notée DFE) si

- $A$  est un attribut non inclus dans  $X$
- et s'il n'existe pas de  $Y \subsetneq X$  tel que  $Y \longrightarrow A$ .

## Exemple

- $A_1, A_2 \longrightarrow A_3$  est élémentaire si ni  $A_1$  ne détermine  $A_3$ , ni  $A_2$  ne détermine  $A_3$ .
- $\text{ISBN} \longrightarrow \text{Titre et numétudiant}$ ,  $\text{cours} \longrightarrow \text{date\_validation\_cours}$  sont élémentaires.
- $\text{ISBN} \longrightarrow \text{Titre, Éditeur}$  n'est pas élémentaire.

# Dépendances fonctionnelles élémentaires

## Remarque

On peut toujours décomposer une DF non élémentaire en DF élémentaires.

## Exemple

ISBN  $\longrightarrow$  Titre, Éditeur

décomposée en

ISBN  $\longrightarrow$  Titre et ISBN  $\longrightarrow$  Éditeur.

# Fermeture transitive des DFE

## Définition

Pour un ensemble  $\mathcal{F}$  de dépendances fonctionnelles élémentaires, on appelle **fermeture transitive** de  $\mathcal{F}$ , notée  $\mathcal{F}^+$ , l'ensemble de toutes les DFE qui peuvent être composées par transitivité à partir des DFE de  $\mathcal{F}$ .



# Fermeture transitive des DFE

## Définition

Pour un ensemble  $\mathcal{F}$  de dépendances fonctionnelles élémentaires, on appelle **fermeture transitive** de  $\mathcal{F}$ , notée  $\mathcal{F}^+$ , l'ensemble de toutes les DFE qui peuvent être composées par transitivité à partir des DFE de  $\mathcal{F}$ .

## Exemple

$$\mathcal{F} = \{A \rightarrow B, B \rightarrow C, B \rightarrow D, B \rightarrow E\}.$$

# Fermeture transitive des DFE

## Définition

Pour un ensemble  $\mathcal{F}$  de dépendances fonctionnelles élémentaires, on appelle **fermeture transitive** de  $\mathcal{F}$ , notée  $\mathcal{F}^+$ , l'ensemble de toutes les DFE qui peuvent être composées par transitivité à partir des DFE de  $\mathcal{F}$ .

## Exemple

$\mathcal{F} = \{A \rightarrow B, B \rightarrow C, B \rightarrow D, B \rightarrow E\}$ . La fermeture transitive de  $\mathcal{F}$  est

$$\mathcal{F}^+ = \{A \rightarrow B, B \rightarrow C, B \rightarrow D, B \rightarrow E, A \rightarrow C, A \rightarrow D, A \rightarrow E\}.$$

# Famille génératrice des DFE

## Définition

Une **famille génératrice** d'un ensemble  $\mathcal{E}$  de DFE est un sous-ensemble minimal  $\mathcal{F}$  (pour l'inclusion) de  $\mathcal{E}$  dont la fermeture transitive  $\mathcal{F}^+$  contient  $\mathcal{E}$ .

# Famille génératrice des DFE

## Définition

Une **famille génératrice** d'un ensemble  $\mathcal{E}$  de DFE est un sous-ensemble minimal  $\mathcal{F}$  (pour l'inclusion) de  $\mathcal{E}$  dont la fermeture transitive  $\mathcal{F}^+$  contient  $\mathcal{E}$ .

## Remarque

Tout ensemble de DFE (donc tout ensemble de DF) admet au moins une famille génératrice.

# Famille génératrice des DFE

## Définition

Une **famille génératrice** d'un ensemble  $\mathcal{E}$  de DFE est un sous-ensemble minimal  $\mathcal{F}$  (pour l'inclusion) de  $\mathcal{E}$  dont la fermeture transitive  $\mathcal{F}^+$  contient  $\mathcal{E}$ .

## Remarque

Tout ensemble de DFE (donc tout ensemble de DF) admet au moins une famille génératrice.

## Exemple

L'ensemble  $\mathcal{E} = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$  admet deux familles génératrices

# Famille génératrice des DFE

## Définition

Une **famille génératrice** d'un ensemble  $\mathcal{E}$  de DFE est un sous-ensemble minimal  $\mathcal{F}$  (pour l'inclusion) de  $\mathcal{E}$  dont la fermeture transitive  $\mathcal{F}^+$  contient  $\mathcal{E}$ .

## Remarque

Tout ensemble de DFE (donc tout ensemble de DF) admet au moins une famille génératrice.

## Exemple

L'ensemble  $\mathcal{E} = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, C \rightarrow B\}$  admet deux familles génératrices

$$\mathcal{F}_1 = \{A \rightarrow C, B \rightarrow C, C \rightarrow B\} \text{ et } \mathcal{F}_2 = \{A \rightarrow B, B \rightarrow C, C \rightarrow B\}.$$

# Exemple et représentation de DF

## Exemple

Voiture(Immat, Constructeur, Modèle, Puissance, Couleur).

- Dépendances fonctionnelles :

$$\mathcal{F} = \{ \text{Immat} \longrightarrow \text{Modèle}, \text{Modèle} \longrightarrow \text{Marque}, \\ \text{Modèle} \longrightarrow \text{Puissance}, \text{Immat} \longrightarrow \text{Couleur} \}.$$

# Exemple et représentation de DF

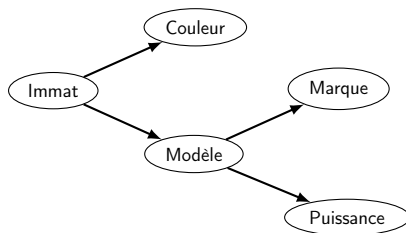
## Exemple

Voiture(Immat, Constructeur, Modèle, Puissance, Couleur).

- Dépendances fonctionnelles :

$$\mathcal{F} = \{ \text{Immat} \rightarrow \text{Modèle}, \text{Modèle} \rightarrow \text{Marque}, \\ \text{Modèle} \rightarrow \text{Puissance}, \text{Immat} \rightarrow \text{Couleur} \}.$$

- On peut représenter  $\mathcal{F}$  par un graphe orienté dont les nœuds sont les attributs et les arcs sont les DFE (avec un seul attribut en destination et éventuellement plusieurs en sources).





# Clé

- On considère une relation  $R(A_1, A_2, \dots, A_n)$ ,  $\mathcal{C}$  un sous-ensemble de  $A_1, A_2, \dots, A_n$ . Alors  $\mathcal{C}$  est une **clé** de  $R$  si
  - $\mathcal{C} \longrightarrow A_1, A_2, \dots, A_n$ ,
  - et il n'existe pas  $\mathcal{D} \subsetneq \mathcal{C}$  tel que  $\mathcal{D} \longrightarrow A_1, A_2, \dots, A_n$ .

# Clé

- On considère une relation  $R(A_1, A_2, \dots, A_n)$ ,  $\mathcal{C}$  un sous-ensemble de  $A_1, A_2, \dots, A_n$ . Alors  $\mathcal{C}$  est une **clé** de  $R$  si
  - $\mathcal{C} \longrightarrow A_1, A_2, \dots, A_n$ ,
  - et il n'existe pas  $\mathcal{D} \subsetneq \mathcal{C}$  tel que  $\mathcal{D} \longrightarrow A_1, A_2, \dots, A_n$ .
- Une **clé** est un ensemble minimal (pour l'inclusion) d'attributs qui déterminent les autres.
- Si une relation comporte plusieurs clés, chacune est dite **clé candidate** et on choisit une qu'on appelle **clé primaire**

# Clé

- On considère une relation  $R(A_1, A_2, \dots, A_n)$ ,  $\mathcal{C}$  un sous-ensemble de  $A_1, A_2, \dots, A_n$ . Alors  $\mathcal{C}$  est une **clé** de  $R$  si
  - $\mathcal{C} \longrightarrow A_1, A_2, \dots, A_n$ ,
  - et il n'existe pas  $\mathcal{D} \subsetneq \mathcal{C}$  tel que  $\mathcal{D} \longrightarrow A_1, A_2, \dots, A_n$ .
- Une **clé** est un ensemble minimal (pour l'inclusion) d'attributs qui déterminent les autres.
- Si une relation comporte plusieurs clés, chacune est dite **clé candidate** et on choisit une qu'on appelle **clé primaire**
- Par définition du modèle relationnel, une relation admet toujours une clé (les n-uplets sont différents deux à deux).

## Exercice : clés

Supposons deux relations  $R(A, B, C)$  et  $S(A, B, C)$  ayant exactement le même schéma. La seule clé de  $R$  est  $A$ , la seule clé de  $S$  est  $B$ . Soient les trois relations suivantes :

- $T = R \cup S$
- $U = R \cap S$
- $V = R - S$

Quelles sont la ou les clés des relations  $T$ ,  $U$  et  $V$ ? Expliquer en quelques lignes et/ou à l'aide d'un petit exemple.

- 1 Conception : dépendances fonctionnelles
- 2 Conception : formes normales
  - Introduction
  - Formes Normales
  - Exemple
  - Exercice
- 3 Autres SGBD, programmer avec un SGBD

# Introduction

- Normaliser un schéma relationnel : le remplacer par un schéma équivalent dont le schéma ne présente pas d'anomalie.
- Outils : analyse des dépendances fonctionnelles.
- Objectifs :
  - éviter les redondances,
  - éviter les problèmes de mises à jours.
- Formes normales (de plus en plus contraignantes) :
  - première forme normale (1FN)
  - deuxième forme normale (2FN)
  - troisième forme normale (3FN)
  - forme normale de Boyce-Codd (FNBC)
  - quatrième forme normale (4FN)
  - cinquième forme normale (5FN).

# Décomposition

## Exemple

Voiture(Immat, Constructeur, Modèle, Puissance, Couleur).

- Dépendances fonctionnelles :

$$\mathcal{F} = \{ \text{Immat} \longrightarrow \text{Modèle}, \text{Modèle} \longrightarrow \text{Marque}, \\ \text{Modèle} \longrightarrow \text{Puissance}, \text{Immat} \longrightarrow \text{Couleur} \}.$$

- Fermeture transitive :

$$\mathcal{F}^+ = \{ \text{Immat} \longrightarrow \text{Modèle}, \text{Modèle} \longrightarrow \text{Marque}, \\ \text{Modèle} \longrightarrow \text{Puissance}, \text{Immat} \longrightarrow \text{Couleur}, \\ \text{Immat} \longrightarrow \text{Marque}, \text{Immat} \longrightarrow \text{Puissance} \}.$$

Décomposition préservant les DF :

Voiture(Immat, Modèle, Couleur)

Modèles(Modèle, Constructeur, Puissance)

# Première Forme Normale

## Définition

Une relation est en **première forme normale (1FN)** si elle ne possède pas d'attribut multi-valué.

## Exemple (Relation qui n'est pas en 1FN)

Personne(numpers, nom, prénom, immat1, immat2)

$$\mathcal{F} = \{ \text{numpers} \longrightarrow \text{nom}, \text{numpers} \longrightarrow \text{prénom}, \\ \text{numpers} \longrightarrow \text{immat1}, \text{numpers} \longrightarrow \text{immat2} \}.$$



# Première Forme Normale

## Définition

Une relation est en **première forme normale (1FN)** si elle ne possède pas d'attribut multi-valué.

## Exemple (Relation qui n'est pas en 1FN)

Personne(numpers, nom, prénom, immat1, immat2)

- Problème 1 : pas plus de deux véhicules,

# Première Forme Normale

## Définition

Une relation est en **première forme normale (1FN)** si elle ne possède pas d'attribut multi-valué.

## Exemple (Relation qui n'est pas en 1FN)

Personne(numpers, nom, prénom, immat1, immat2)

- Problème 1 : pas plus de deux véhicules,
- Problème 2 : immat2 peut prendre de la place inutilement.

# Première Forme Normale

## Définition

Une relation est en **première forme normale (1FN)** si elle ne possède pas d'attribut multi-valué.

## Exemple (Relation qui n'est pas en 1FN)

Personne(numpers, nom, prénom, immat1, immat2)

- Problème 1 : pas plus de deux véhicules,
- Problème 2 : immat2 peut prendre de la place inutilement.

## Solution

Créer une nouvelle relation avec un attribut qui correspond à une composante de l'attribut multi-valué. Ajouter comme clé étrangère la clé primaire de la première relation.

Personne(numpers, nom, prénom)

Véhicule(numpers, immatriculation)

# Deuxième Forme Normale

## Définition

Une relation est en **deuxième forme normale (2FN)** si elle est en 1FN et aucun attribut non-clé ne dépend que d'une partie de la clé.


# Deuxième Forme Normale

## Définition

Une relation est en **deuxième forme normale (2FN)** si elle est en 1FN et aucun attribut non-clé ne dépend que d'une partie de la clé.

## Exemple (Relation 1FN non 2FN)

Projet (numprojet, numemployé, fonction, nomemployé)



$\mathcal{F} = \{ \text{numprojet}, \text{numemployé} \rightarrow \text{fonction},$   
 $\text{numemployé} \rightarrow \text{nomemployé} \}.$


# Deuxième Forme Normale

## Définition

Une relation est en **deuxième forme normale (2FN)** si elle est en 1FN et aucun attribut non-clé ne dépend que d'une partie de la clé.

## Exemple (Relation 1FN non 2FN)

Projet (numprojet, numemployé, fonction, nomemployé)



- Problème 1 : employé enregistré seulement si dans un projet.


# Deuxième Forme Normale

## Définition

Une relation est en **deuxième forme normale (2FN)** si elle est en 1FN et aucun attribut non-clé ne dépend que d'une partie de la clé.

## Exemple (Relation 1FN non 2FN)

Projet (numprojet, numemployé, fonction, nomemployé)



- Problème 1 : employé enregistré seulement si dans un projet.
- Problème 2 : redondance si employé dans plusieurs projets.


# Deuxième Forme Normale

## Définition

Une relation est en **deuxième forme normale (2FN)** si elle est en 1FN et aucun attribut non-clé ne dépend que d'une partie de la clé.

## Exemple (Relation 1FN non 2FN)

Projet (numprojet, numemployé, fonction, nomemployé)



- Problème 1 : employé enregistré seulement si dans un projet.
- Problème 2 : redondance si employé dans plusieurs projets.

## Solution

On extrait la dépendance fonctionnelle : la clé primaire de la nouvelle relation est la partie de clé primaire utile de départ.

Employé(numemployé, nomemployé)

Projet(numprojet, numemployé, fonction)



# Troisième Forme Normale

## Définition

Une relation est en **troisième forme normale (3FN)** si elle est en 2FN et si tout attribut n'appartenant à aucune clé (candidate) ne dépend que des clés candidates.


# Troisième Forme Normale

## Définition

Une relation est en **troisième forme normale (3FN)** si elle est en 2FN et si tout attribut n'appartenant à aucune clé (candidate) ne dépend que des clés candidates.

## Exemple (Relation 2FN non 3FN)

Employé (numemployé, nomemp, numservice, nomservice)



$\mathcal{F} = \{ \text{numemployé} \rightarrow \text{nomemp}, \text{numemployé} \rightarrow \text{numservice}, \text{numservice} \rightarrow \text{nomservice} \}.$


# Troisième Forme Normale

## Définition

Une relation est en **troisième forme normale (3FN)** si elle est en 2FN et si tout attribut n'appartenant à aucune clé (candidate) ne dépend que des clés candidates.

## Exemple (Relation 2FN non 3FN)

Employé (numemployé, nomemp, numservice, nomservice)



- Problème 1 : redondance (nom de service).


# Troisième Forme Normale

## Définition

Une relation est en **troisième forme normale (3FN)** si elle est en 2FN et si tout attribut n'appartenant à aucune clé (candidate) ne dépend que des clés candidates.

## Exemple (Relation 2FN non 3FN)

Employé (numemployé, nomemp, numservice, nomservice)



- Problème 1 : redondance (nom de service).
- Problème 2 : ajout d'info sur service : traiter tous les n-uplets.


# Troisième Forme Normale

## Définition

Une relation est en **troisième forme normale (3FN)** si elle est en 2FN et si tout attribut n'appartenant à aucune clé (candidate) ne dépend que des clés candidates.

## Exemple (Relation 2FN non 3FN)

Employé (numemployé, nomemp, numservice, nomservice)



- Problème 1 : redondance (nom de service).
- Problème 2 : ajout d'info sur service : traiter tous les n-uplets.

## Solution

Créer une nouvelle relation contenant l'attribut de la partie droite de la DF.  
La clé primaire est la partie gauche de la DF.

Service(numservice, nomservice)

Employé(numemployé, nomemp, numservice)

# Forme Normale de Boyce-Codd

## Définition

Une relation est en **forme normale de Boyce-Codd (FNBC)** si elle est en 3FN et s'il n'existe pas de dépendance fonctionnelle dont la partie gauche n'est pas clé de la relation.


# Forme Normale de Boyce-Codd

## Définition

Une relation est en **forme normale de Boyce-Codd (FNBC)** si elle est en 3FN et s'il n'existe pas de dépendance fonctionnelle dont la partie gauche n'est pas clé de la relation.

## Exemple (Relation 3FN non FNBC)

Personne (numSS, Pays, Nom, Région)



$\mathcal{F} = \{ \text{numSS, Pays} \longrightarrow \text{nom}, \text{numSS, Pays} \longrightarrow \text{Région}, \\ \text{Région} \longrightarrow \text{Pays} \}.$

Clés candidates : (numSS, Pays) et (numSS, Région); la partie gauche de Région  $\longrightarrow$  Pays n'est pas clé.


# Forme Normale de Boyce-Codd

## Définition

Une relation est en **forme normale de Boyce-Codd (FNBC)** si elle est en 3FN et s'il n'existe pas de dépendance fonctionnelle dont la partie gauche n'est pas clé de la relation.

## Exemple (Relation 3FN non FNBC)

Personne (numSS, Pays, Nom, Région)



Clés candidates : (numSS, Pays) et (numSS, Région) ; la partie gauche de Région  $\rightarrow$  Pays n'est pas clé.

- Problème : redondance.




# Forme Normale de Boyce-Codd

## Définition

Une relation est en **forme normale de Boyce-Codd (FNBC)** si elle est en 3FN et s'il n'existe pas de dépendance fonctionnelle dont la partie gauche n'est pas clé de la relation.

## Exemple (Relation 3FN non FNBC)

Personne (numSS, Pays, Nom, Région)



Clés candidates : (numSS, Pays) et (numSS, Région) ; la partie gauche de Région  $\rightarrow$  Pays n'est pas clé.

- Problème : redondance.

## Solution

Créer une nouvelle relation avec les deux attributs de la DF et dont la clé primaire sera l'attribut en partie gauche de la DF. L'attribut à droite de la DF est supprimé de l'ancienne relation.

Région(Région, Pays)

Personne(numSS, Région, Nom)

# Bilan

- L'objectif de la décomposition d'une relation  $R$  est de la « découper » en relations plus petites  $R_1, R_2, \dots, R_m$  pour éviter la redondance.
- Une décomposition préserve les dépendances fonctionnelles si la fermeture transitive  $\mathcal{F}^+$  des DF de  $R$  est la même que la fermeture transitive  $\mathcal{F}'^+$  de l'union des DF de  $R_1, R_2, \dots, R_n$ .
- La mise en troisième forme normale 3FN préserve les dépendances fonctionnelles. Elle évite des anomalies provoquées par les redondances.
- La mise en forme normale de Boyce-Codd (FNBC) ne préserve pas forcément les dépendances fonctionnelles, mais elle peut éviter des redondances supplémentaires.
- 4FN et 5FN : pas vues dans ce cours.

# Exemple

- Analyse de l'existant :

- Chaque abonné a un numéro d'abonné, un nom, un prénom, une adresse et une date d'abonnement.

Les auteurs qui écrivent les livres ont un numéro d'auteur, un nom et un prénom.

Les livres ont un numéro isbn, un titre, un nombre de pages et un éditeur.

Chaque livre appartient à un genre qui est répertorié par son code (ex : SF) et son nom (ex : Science-Fiction).

Quand un abonné emprunte un livre, on mémorise la date d'emprunt ; lorsqu'il le restitue, on mémorise la date de restitution.

# Exemple

- Analyse de l'existant :

- Chaque abonné a un numéro d'abonné, un nom, un prénom, une adresse et une date d'abonnement.

Les auteurs qui écrivent les livres ont un numéro d'auteur, un nom et un prénom.

Les livres ont un numéro isbn, un titre, un nombre de pages et un éditeur.

Chaque livre appartient à un genre qui est répertorié par son code (ex : SF) et son nom (ex : Science-Fiction).

Quand un abonné emprunte un livre, on mémorise la date d'emprunt ; lorsqu'il le restitue, on mémorise la date de restitution.

- Schéma relationnel :

Abonné(numabo, nomabo, prénomabo, dateabo)

Livre(isbn, titre, éditeur, nbrpages, codegen)

Emprunt(numabo, isbn, dateemp, dateret)

Genre(codegen, nomgen)

Auteur(numaut, nomaut, prénomaut)

Écrit(numaut, isbn)

## Exemple (2)

- Dépendances fonctionnelles

$\text{numabo} \longrightarrow \text{nomabo}, \text{prénomabo}, \text{dateabo}$

$\text{isbn} \longrightarrow \text{titre}, \text{éditeur}, \text{nbrpages}, \text{codegen}$

$\text{numabo}, \text{isbn}, \text{dateemp} \longrightarrow \text{dateret}$

$\text{codegen} \longrightarrow \text{nomgen}$

$\text{numaut} \longrightarrow \text{nomaut}, \text{prénomaut}$

$\text{numaut}, \text{isbn} \longrightarrow \emptyset$

## Exemple (2)

- Dépendances fonctionnelles

$\text{numabo} \longrightarrow \text{nomabo}, \text{prénomabo}, \text{dateabo}$

$\text{isbn} \longrightarrow \text{titre}, \text{éditeur}, \text{nbrpages}, \text{codegen}$

$\text{numabo}, \text{isbn}, \text{dateemp} \longrightarrow \text{dateret}$

$\text{codegen} \longrightarrow \text{nomgen}$

$\text{numaut} \longrightarrow \text{nomaut}, \text{prénomaut}$

$\text{numaut}, \text{isbn} \longrightarrow \emptyset$

- On vérifie la correspondance des relations et des dépendances fonctionnelles réelles.

## Exemple (2)

- Dépendances fonctionnelles

$\text{numabo} \longrightarrow \text{nomabo}, \text{prénomabo}, \text{dateabo}$

$\text{isbn} \longrightarrow \text{titre}, \text{éditeur}, \text{nbrpages}, \text{codegen}$

$\text{numabo}, \text{isbn}, \text{dateemp} \longrightarrow \text{dateret}$

$\text{codegen} \longrightarrow \text{nomgen}$

$\text{numaut} \longrightarrow \text{nomaut}, \text{prénomaut}$

$\text{numaut}, \text{isbn} \longrightarrow \emptyset$

- On vérifie la correspondance des relations et des dépendances fonctionnelles réelles.
- Sous quelle forme normale (maximale) est ce schéma relationnel ?

## Exemple (2)

- Dépendances fonctionnelles

$\text{numabo} \rightarrow \text{nomabo}, \text{prénomabo}, \text{dateabo}$

$\text{isbn} \rightarrow \text{titre}, \text{éditeur}, \text{nbrpages}, \text{codegen}$

$\text{numabo}, \text{isbn}, \text{dateemp} \rightarrow \text{dateret}$

$\text{codegen} \rightarrow \text{nomgen}$

$\text{numaut} \rightarrow \text{nomaut}, \text{prénomaut}$

$\text{numaut}, \text{isbn} \rightarrow \emptyset$

- On vérifie la correspondance des relations et des dépendances fonctionnelles réelles.
- Sous quelle forme normale (maximale) est ce schéma relationnel ?

### Solution

Ce schéma est en forme normale de Boyce-Codd.



## Exercice : les commandes

Soit la relation suivante :

Commande(NumCommande, NumProduit, QuantitéCommandée, NumClient, NumReprésentant)

dans laquelle les dépendances fonctionnelles valides sont les suivantes :

NumCommande, NumProduit  $\longrightarrow$  QuantitéCommandée, NumClient, NumReprésentant

NumCommande  $\longrightarrow$  NumClient, NumReprésentant

NumClient  $\longrightarrow$  NumReprésentant

- 1 Proposer une clé primaire valide pour cette relation.
- 2 Expliquer pourquoi cette relation n'est pas en 2FN.
- 3 Décomposer la relation Commande en deux relations distinctes pour obtenir un schéma relationnel en 2FN, tout en préservant les dépendances. Attention, on ne demande ici que la 2FN.
- 4 Les tables obtenues sont-elles en 3FN ? Expliquer votre réponse. Si ce n'est pas le cas, modifier le schéma afin d'obtenir un résultat en 3FN.

- 1 Conception : dépendances fonctionnelles
- 2 Conception : formes normales
- 3 Autres SGBD, programmer avec un SGBD
  - Autre SGBD : MySQL / MariaDB
  - Programmer avec une base de données

# MySQL / MariaDB

- Serveur de bases de données relationnelles.
- Principe : connexion.
- MySQL : double licence GPL et propriétaire.
- MariaDB : projet OpenSource dérivé de MySQL.
- L'outil phpMyAdmin est développé en PHP et offre une interface graphique pour l'administration des base de données. Il peut utiliser MySQL ou MariaDB.

# Intérêts de MySQL / MariaDB

- Par rapport à SQLite, présente des intérêts :

# Intérêts de MySQL / MariaDB

- Par rapport à SQLite, présente des intérêts :
  - gérer des utilisateurs différents de façon fine
    - on peut donner (ou révoquer) des privilèges à un utilisateur qui lui permet de créer des tables, de lire des fichiers...

# Intérêts de MySQL / MariaDB

- Par rapport à SQLite, présente des intérêts :
  - gérer des utilisateurs différents de façon fine
    - on peut donner (ou révoquer) des privilèges à un utilisateur qui lui permet de créer des tables, de lire des fichiers...
  - SQLite peut seulement utiliser les permissions « traditionnelles » sur les fichiers.

# Intérêts de MySQL / MariaDB

- Par rapport à SQLite, présente des intérêts :
  - gérer des utilisateurs différents de façon fine
    - on peut donner (ou révoquer) des privilèges à un utilisateur qui lui permet de créer des tables, de lire des fichiers...
  - SQLite peut seulement utiliser les permissions « traditionnelles » sur les fichiers.
- MySQL peut gérer beaucoup plus de types :
  - Tinyint, Smallint, Mediumint, Int, Bigint, Double, Float, Real, Decimal, Double precision, Numeric, Timestamp, Date, Datetime, Char, Varchar, Year, Tinytext, Tinyblob, Blob, Text, MediumBlob, MediumText, Enum, Set, Longblob, Longtext...

# Principes généraux

- Nous avons vu le langage SQL, standard pour les bases de données relationnelle.



# Principes généraux

- Nous avons vu le langage SQL, standard pour les bases de données relationnelle.
- En pratique, la plupart des accès aux bases de données se font à travers de logiciels qui implémentent des applications.
  - Ces logiciels sont écrits habituellement dans un langage généraliste (C, C++, Java, Python, etc).

# Principes généraux

- Nous avons vu le langage SQL, standard pour les bases de données relationnelle.
- En pratique, la plupart des accès aux bases de données se font à travers de logiciels qui implémentent des applications.
  - Ces logiciels sont écrits habituellement dans un langage généraliste (C, C++, Java, Python, etc).
  - Beaucoup de langages de scripts (comme PHP et JavaScript) sont aussi utilisés pour l'accès aux bases de données pour les applications Web.

# Principes généraux

- Nous avons vu le langage SQL, standard pour les bases de données relationnelle.
- En pratique, la plupart des accès aux bases de données se font à travers de logiciels qui implémentent des applications.
  - Ces logiciels sont écrits habituellement dans un langage généraliste (C, C++, Java, Python, etc).
  - Beaucoup de langages de scripts (comme PHP et JavaScript) sont aussi utilisés pour l'accès aux bases de données pour les applications Web.
- C'est un sujet très large, il existe des livres entiers dédiés à une technique particulière de programmation sur les bases de données.

# Approches de la programmation de base de données

Plusieurs techniques existent pour inclure les interactions de bases de données dans les programmes. Les approches principales sont les suivantes :

- Incorporer les commandes de bases de données dans un langage généraliste. Les commandes SQL sont identifiées par un préfixe spécial, par exemple EXEC SQL.

# Approches de la programmation de base de données

Plusieurs techniques existent pour inclure les interactions de bases de données dans les programmes. Les approches principales sont les suivantes :

- Incorporer les commandes de bases de données dans un langage généraliste. Les commandes SQL sont identifiées par un préfixe spécial, par exemple EXEC SQL.
- Utiliser une bibliothèque de fonctions de bases de données.

# Approches de la programmation de base de données

Plusieurs techniques existent pour inclure les interactions de bases de données dans les programmes. Les approches principales sont les suivantes :

- Incorporer les commandes de bases de données dans un langage généraliste. Les commandes SQL sont identifiées par un préfixe spécial, par exemple EXEC SQL.
- Utiliser une bibliothèque de fonctions de bases de données.
- Inventer un tout nouveau langage de programmation pour qu'il soit compatible avec le modèle de base de données et le langage de requêtes. Exemple : PL/SQL créé par Oracle.

# Séquence typique d'interaction en programmation de base de données

- Le modèle commun d'accès est le modèle client-serveur où un **programme client** fait quelques appels à un ou plusieurs **serveurs de bases de données** pour accéder ou mettre à jour les données.

# Séquence typique d'interaction en programmation de base de données

- Le modèle commun d'accès est le modèle client-serveur où un **programme client** fait quelques appels à un ou plusieurs **serveurs de bases de données** pour accéder ou mettre à jour les données.
  - Le programme client ouvre une **connexion** au serveur de BD.



# Séquence typique d'interaction en programmation de base de données

- Le modèle commun d'accès est le modèle client-serveur où un **programme client** fait quelques appels à un ou plusieurs **serveurs de bases de données** pour accéder ou mettre à jour les données.
  - Le programme client ouvre une **connexion** au serveur de BD.
  - Une fois la connexion établie, le programme interagit avec la BD en envoyant des requêtes.

# Séquence typique d'interaction en programmation de base de données

- Le modèle commun d'accès est le modèle client-serveur où un **programme client** fait quelques appels à un ou plusieurs **serveurs de bases de données** pour accéder ou mettre à jour les données.
  - Le programme client ouvre une **connexion** au serveur de BD.
  - Une fois la connexion établie, le programme interagit avec la BD en envoyant des requêtes.
  - Quand le programme n'a plus besoin de la BD, il ferme la connexion.

# Notion de curseur

- Le SQL opère sur un ensemble de lignes, qui n'est pas forcément bien géré par le langage support.

# Notion de curseur

- Le SQL opère sur un ensemble de lignes, qui n'est pas forcément bien géré par le langage support.
- Un **curseur** peut être défini sur n'importe quelle relation ou n'importe quelle requête qui produit un ensemble de lignes. Une fois un curseur déclaré,

# Notion de curseur

- Le SQL opère sur un ensemble de lignes, qui n'est pas forcément bien géré par le langage support.
- Un **curseur** peut être défini sur n'importe quelle relation ou n'importe quelle requête qui produit un ensemble de lignes. Une fois un curseur déclaré,
  - On peut l'ouvrir, ce qui le positionne juste avant la première ligne

# Notion de curseur

- Le SQL opère sur un ensemble de lignes, qui n'est pas forcément bien géré par le langage support.
- Un **curseur** peut être défini sur n'importe quelle relation ou n'importe quelle requête qui produit un ensemble de lignes. Une fois un curseur déclaré,
  - On peut l'ouvrir, ce qui le positionne juste avant la première ligne
  - Aller chercher la première ligne

# Notion de curseur

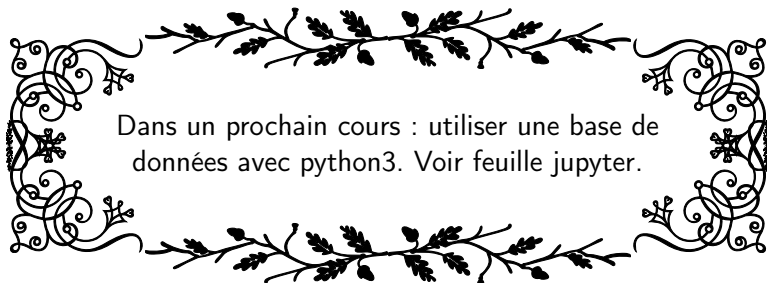
- Le SQL opère sur un ensemble de lignes, qui n'est pas forcément bien géré par le langage support.
- Un **curseur** peut être défini sur n'importe quelle relation ou n'importe quelle requête qui produit un ensemble de lignes. Une fois un curseur déclaré,
  - On peut l'ouvrir, ce qui le positionne juste avant la première ligne
  - Aller chercher la première ligne
  - Déplacer le curseur

# Notion de curseur

- Le SQL opère sur un ensemble de lignes, qui n'est pas forcément bien géré par le langage support.
- Un **curseur** peut être défini sur n'importe quelle relation ou n'importe quelle requête qui produit un ensemble de lignes. Une fois un curseur déclaré,
  - On peut l'ouvrir, ce qui le positionne juste avant la première ligne
  - Aller chercher la première ligne
  - Déplacer le curseur
  - Fermer le curseur



## À suivre



Dans un prochain cours : utiliser une base de données avec python3. Voir feuille jupyter.