

Conception de Bases de Données

Normalisation

Verónica Peralta

2021-2022

Overview

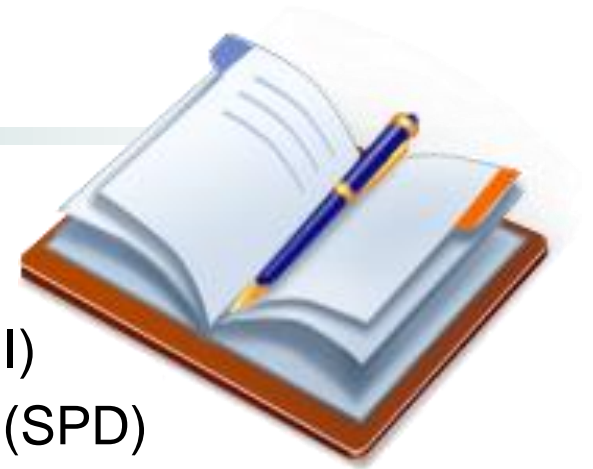
◆ Dans les séances précédentes nous avons vu :

- Dépendances fonctionnelles :
 - Comment identifier des DF ?
 - Comment savoir si une DF est dérivable ?
- Couverture minimale :
 - Comment calculer un ensemble minimal de DF à contrôler ?
- Clés
 - Comment trouver les clés d'une relation à partir des DF ?

Overview

- ◆ **Grande avantages des clés sur les DFs :**
 - Les SGDB fournissent le contrôle des clés alors qu'ils ne fournissent pas le contrôle des DFs.
- ◆ **Notre but est donc de transformer les DF en clés !**
- ◆ **Technique de base :**
 - Identifier les tables avec des DFs qui ne correspondent pas aux clés et les **décomposer** en tables plus petites
- ◆ **Dans ce séance nous verrons :**
 - Des techniques de décomposition (appelé *normalisation*) et des critères pour obtenir des bonnes décompositions

Plan de la séance



◆ Décomposition de relations

- Décomposition sans perte d'information (SPI)
- Décomposition sans perte de dépendances (SPD)

◆ Formes normales (FN)

◆ Algorithmes de normalisation

- Décomposition en 3NF SPI et SPD
- Décomposition en BCNF SPI

◆ Pour aller plus loin...

*séance
suivante*

Décomposition de relations

◆ Décomposition

- Une décomposition d'un schéma de relation R est un ensemble de schémas de relation $D=\{R_1...R_m\}$ où tous les attributs de R se retrouvent dans au moins un des schémas de D

$$\cup_{i=1}^m \text{sort}(R_i) = \text{sort}(R)$$

- Les instances $I_1...I_m$ de $R_1...R_m$ se obtiennent comme projection de l'instance I de R

$$I_i = \pi_{R_i} (I)$$

Décomposition de relations

◆ Exemple

- Soient un schéma de relation $R[ABC]$ et une instance I de R .
- Une possible décomposition est $\rho = \{ R1[AB], R2[BC] \}$
- La décomposition nous donne deux instances de relation:
 - L'instance $I1$, sur $\text{sort}(R1) = \{A,B\}$
 - L'instance $I2$, sur $\text{sort}(R2) = \{B,C\}$.

A	B	C
a1	b1	c1
a2	b2	c2

A	B
a1	b1
a2	b2

B	C
b1	c1
b2	c2

Décomposition de relations

◆ Questions :

- Comment savoir si une décomposition est bonne ?
- Comment trouver une bonne décomposition ?

◆ Il nous faut :

- Des techniques de décomposition
- Des critères pour obtenir des bonnes décompositions

Décomposition sans perte d'information

♦ Motivation :

- Soient un schéma de relation $R[ABC]$ et une instance $I(R)$.
- Soit une décomposition $\rho = \{ R_1[AB], R_2[BC] \}$ de R
- Il serait souhaitable de joindre R_1 et R_2 et obtenir R , mais...

R			R ₁		R ₂		R ₁ ⋈ R ₂		
A	B	C	A	B	B	C	A	B	C
a1	b	c1	a1	b	b	c1	a1	b	c1
a2	b	c2	a2	b	b	c2	a1	b	c2
							a2	b	c1
							a2	b	c2

Décomposition sans perte d'information

◆ Définition : SPI

- Soient R un schéma de relation, $\rho = \{R_1, R_2, \dots, R_k\}$ une décomposition et F un ensemble de DF
- La décomposition ρ est **sans perte d'information (SPI)** par rapport à F si pour toute instance I sur R qui satisfait F , nous pouvons reconstruire R via des jointures naturelles

$$I = \pi_{R_1}(I) \bowtie \pi_{R_2}(I) \bowtie \dots \bowtie \pi_{R_k}(I)$$

Décomposition sans perte d'information

◆ Sous quelles conditions une décomposition ρ est SPI ?

◆ Idée intuitive :

- On a des problèmes quand les attributs en commun ne déterminent pas les autres (pas clé) dans au moins une des relations de ρ

R			R ₁		R ₂		R ₁ ⋈ R ₂		
A	B	C	A	B	B	C	A	B	C
a1	b	c1	a1	b	b	c1	a1	b	c1
a2	b	c2	a2	b	b	c2	a1	b	c2
							a2	b	c1
							a2	b	c2

Algorithme CHASE

- ♦ **Algorithme générale pour vérifier s'une décomposition est SPI**
- ♦ **Entrée**
 - Un schéma de relation $R(A_1 \dots A_n)$
 - Un ensemble de dépendances fonctionnelles F sur R
 - Une décomposition $\rho = \{R_1 \dots R_k\}$
- ♦ **Sortie:**
 - **Oui** ou **Non** la décomposition est SPI par rapport à F

Algorithme CHASE

- 1. Construire une matrice S avec n colonnes et k lignes**
 - La colonne j correspond à l'attribut A_j et la ligne i à la relation R_i
- 2. Sur la position $S[i,j]$:**
 - si A_j appartient à R_i faire $S[i,j] = a_j$ (une constante)
 - Sinon faire $S[i,j] = b_{ij}$ (une variable)
- 3. Répéter jusqu'à n'avoir aucun changement sur S :**
 - Pour chaque DF $X \rightarrow Y$ dans F
 - S'il existe des lignes avec les mêmes valeurs sur X , rendre égales les valeurs sur Y (en utilisant des a_j si possible)
- 4. La décomposition est SPI ssi il existe une ligne avec les valeurs $\langle a_1 \dots a_n \rangle$ dans S**

Algorithme CHASE

♦ Exemple :

- EMPLOYES-PROJETS (nss, nomE, numeroP, nomP, villeP, heures)
- $R_1(\text{nss}, \text{nomE})$
- $R_2(\text{numeroP}, \text{nomP}, \text{villeP})$
- $R_3(\text{nss}, \text{numeroP}, \text{heures})$
- $F = \{\text{nss} \rightarrow \text{nomE}, \text{numeroP} \rightarrow \text{nomP}, \text{villeP}, \text{nss}, \text{numeroP} \rightarrow \text{heures}\}$

Pas 1 : Construire une matrice S avec n colonnes (A_j) et k lignes (R_i)

	nss	nomE	numeroP	nomP	villeP	heures
R_1						
R_2						
R_3						

Algorithme CHASE

♦ Exemple :

- EMPLOYES-PROJETS (nss, nomE, numeroP, nomP, villeP, heures)
- $R_1(\text{nss}, \text{nomE})$
- $R_2(\text{numeroP}, \text{nomP}, \text{villeP})$
- $R_3(\text{nss}, \text{numeroP}, \text{heures})$
- $F = \{\text{nss} \rightarrow \text{nomE}, \text{numeroP} \rightarrow \text{nomP}, \text{villeP}, \text{nss}, \text{numeroP} \rightarrow \text{heures}\}$

Pas 2 : si A_j appartient à R_i faire $S[i,j] = a_j$ sinon faire $S[i,j] = b_{ij}$

	nss	nomE	numeroP	nomP	villeP	heures
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

Algorithme CHASE

◆ Exemple :

– $F = \{ \text{nss} \rightarrow \text{nomE}, \text{numeroP} \rightarrow \text{nomP villeP}, \text{nss numeroP} \rightarrow \text{heures} \}$

Pas 3 : Répéter jusqu'à n'avoir aucun changement sur S :

- Pour chaque DF $X \rightarrow Y$ dans F
 - S'il existe des lignes avec les mêmes valeurs sur X, rendre égales les valeurs sur Y

Pas 4 : La décomposition est SPI ssi il existe une ligne avec les valeurs $\langle a_1 \dots a_n \rangle$ dans S

	nss	nomE	numeroP	nomP	villeP	heures
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	a_2	a_3	a_4	a_5	a_6

Projection de dépendances

♦ Motivation :

- Soit $R[ABC]$ et un ensemble $F = \{A \rightarrow C\}$ sur R .
- Soit une décomposition $\rho = \{ R_1[AB], R_2[BC] \}$ de R

R	A	B	C
	a1	b1	c1
	a2	b2	c2

R ₁	A	B
	a1	b1
	a2	b2

R ₂	B	C
	b1	c1
	b2	c2

♦ Lors d'une màj, il faut vérifier la validation des DF

- Il faut reconstituer la relation ($R_1 \bowtie R_2$) et vérifier les DF ($A \rightarrow C$).
- Cette reconstruction est très coûteuse !

♦ Il nous faut :

- Pouvoir exprimer les DF sur les relations de la décomposition.

Projection de dépendances

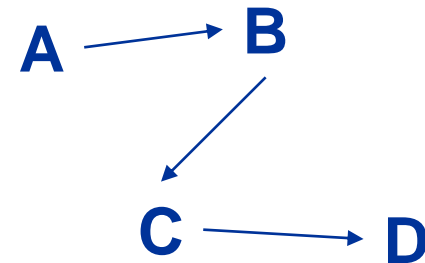
◆ Projection d'un ensemble de DF F sur une relation R

- C'est l'ensemble de toutes les DF impliquées par F et qui sont applicables sur R

$$F_R = \{X \rightarrow Y \mid X \rightarrow Y \in F^+ \wedge XY \subseteq \text{sort}(R)\}$$

◆ Exemple :

- $R(A,B,C,D)$; $\rho = \{R1(AB), R2(BCD)\}$
- $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$
- $F_{R1} = \{A \rightarrow B\}$
- $F_{R2} = \{B \rightarrow C, C \rightarrow D\}$



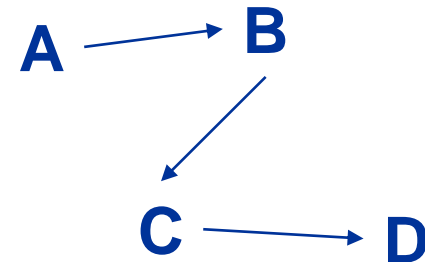
Projection de dépendances

◆ Attention :

- Il ne suffit pas de projeter les DF de F, **il faut considérer F+**

◆ Exemple :

- $R(A,B,C,D)$; $\rho = \{R1(\text{AC}), R2(BCD)\}$
- $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$
- $F_{R1} = \{A \rightarrow C\}$
- $F_{R2} = \{B \rightarrow C, C \rightarrow D\}$



◆ Deux stratégies :

- Obtenir une couverture maximale de F
- Vérifier les DF candidates sur les attributs de R1 et R2

Décomposition sans perte de dépendances

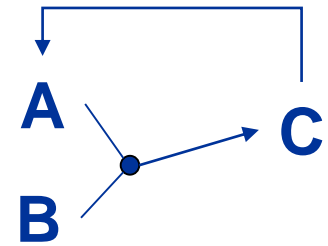
◆ Définition : SPD

- Une décomposition $\rho = \{R_1 \dots R_n\}$ est **sans perte de dépendances (SPD)** par rapport à un ensemble de DF F si l'union de toutes les DF projetés en $R_1 \dots R_n$ est équivalente à F

$$F_{R_1} \cup F_{R_2} \cup \dots \cup F_{R_n} \equiv F$$

◆ Exemple :

- $R(A,B,C)$; $\rho = \{R_1(AC), R_2(BC)\}$
- $F = \{AB \rightarrow C, C \rightarrow A\}$
- $F_{R_1} = \{C \rightarrow A\}$
- $F_{R_2} = \{\}$
- $G = F_{R_1} \cup F_{R_2} = \{C \rightarrow A\}$
- **ρ n'est pas SPD !**



Exercice: Projeter des DF

R(ABCDEFG)

Indiquer les DF que se projettent sur :

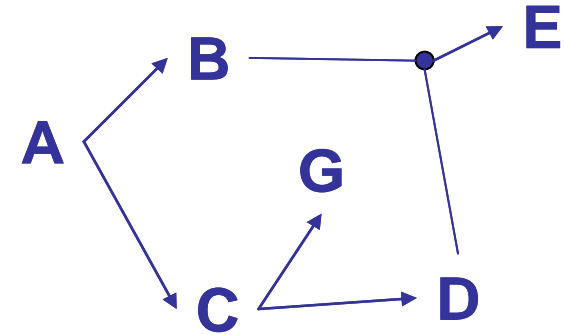
R1(ABC)

R2(CDG)

R3(BDE)

R4(ADG)

R5(ADE)



La décomposition $D = (R1, R2, R3)$ est-elle SPD ?

Et la décomposition $D2 = (R1, R3, R4)$?

Point sur la conception BD

◆ Nous avons vu :

- Ce qui est une décomposition
- Des critères de qualité d'une décomposition :
 - SPI
 - SPD

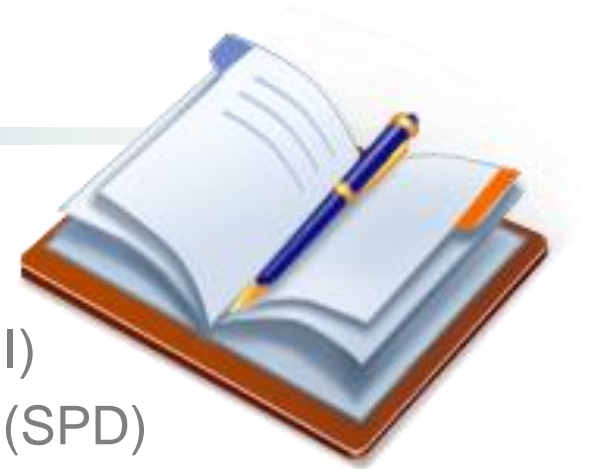
◆ Notre but est encore réduire la redondance

- Transformer les DF en clés

◆ Il nous faut :

- Produire des décompositions, SPI et SPD, basés sur les DF

Plan de la séance



- ◆ **Décomposition de relations**
 - Décomposition sans perte d'information (SPI)
 - Décomposition sans perte de dépendances (SPD)

- ◆ **Formes normales (FN)**

- ◆ **Algorithmes de normalisation**
 - Décomposition en 3NF SPI et SPD
 - Décomposition en BCNF SPI

- ◆ **Pour aller plus loin...**

Normalisation

- ♦ Le **processus de normalisation** consiste à transformer des schémas dans une forme structurale qui satisfait une collection des règles
 - Un schéma de relation est soumis à une série de tests pour “certifier” s’il est ou pas dans une certaine **forme normale**
 - Les schémas pas satisfaisants sont décomposés afin d’attendre une forme normale plus élevé
- ♦ Un schéma dans une **forme normale** possède certaines caractéristiques de qualité
 - Il existe plusieurs formes normales : **1NF, 2NF, 3NF, BCNF, 4NF...**
 - Les formes normales les plus élevés assurent un meilleur niveau de qualité (en particulier moins de redondance)

Première forme normale (1NF)

◆ Définition : 1NF

- Un schéma de relation est en 1NF si les domaines des attributs incluent seulement des valeurs atomiques (pas des listes de valeurs, ni valeurs composés)

◆ Exemple :

DEPARTMENTS

DeptName	<u>DeptCode</u>	ManagerSSN	Offices
Investigation	5	333445555	(Blois, Tours, Paris)
Administration	4	987654321	Bordeaux
Direction	1	888665555	Madrid, Spain

- Des valeurs de l'attribut *Offices* violent 1NF

Deuxième forme normale (2NF)

◆ Attribut de clé :

- Un attribut d'un schéma de relation R est un **attribut de clé** s'il appartient à une clé de R. Sinon, il est dit un **attribut non-clé**.

◆ Observation :

- Beaucoup de redondances apparaissent quand un attribut non-clé dépend d'une partie de la clé.

◆ Exemple :

EMPLOYEE-PROJECT

<u>SSN</u>	<u>ProjCode</u>	Hours	EmpName	ProjName	ProjCity
17809...	1	21	José Silva	Automation	Blois
17809...	2	14	José Silva	Credit	Tours
26512...	2	35	Ann Martin	Credit	Tours

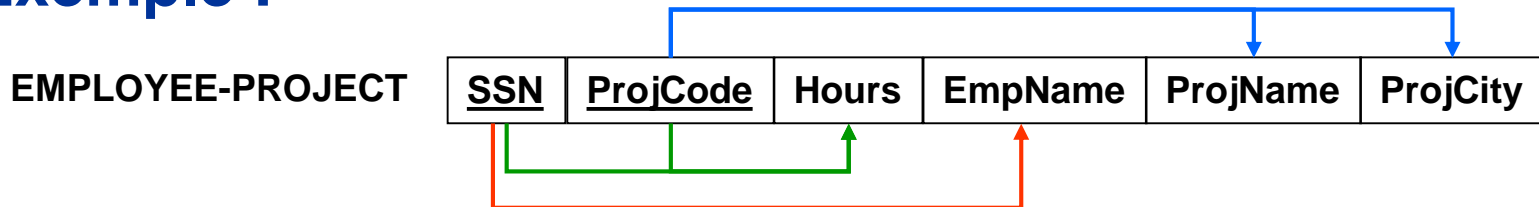
- Clé : {SSN, ProjCode}
- L'attribut non-clé *EmpName* dépend de *SSN*
- Les attributs non-clé *ProjName* et *ProjCity* dépendent de *ProjCode*

Deuxième forme normale (2NF)

◆ Définition : 2NF

- Un schéma de relation R est en 2NF si :
 - il est en 1NF et
 - aucun attribut non clé dépend partiellement d'une clé
- Dit autrement : **il n'y a pas** de DF $X \rightarrow A$ où :
 - X est un sous-ensemble stricte d'une clé de R, et
 - A est un attribut non-clé

◆ Exemple :



- Le schéma est en 1NF
- **SSN** → **EmpName** et **ProjCode** → **ProjName ProjCity** violent 2NF

Troisième forme normale (3NF)


♦ Observation :

- D'autres redondances apparaissent quand un attribut non-clé dépend d'attributs non-clé.

♦ Exemple :

EMPLOYEE-DEPARTMENT

<u>SSN</u>	EmpName	Birthdate	Address	DeptCode	DeptName	MngerSSN
17809...	José Silva	150978	Blois	1	Diffusion	18002...
18002...	John Smith	140280	Paris	1	Diffusion	18002...
26512...	Ann Martin	061265	Tours	2	Direction	26704...



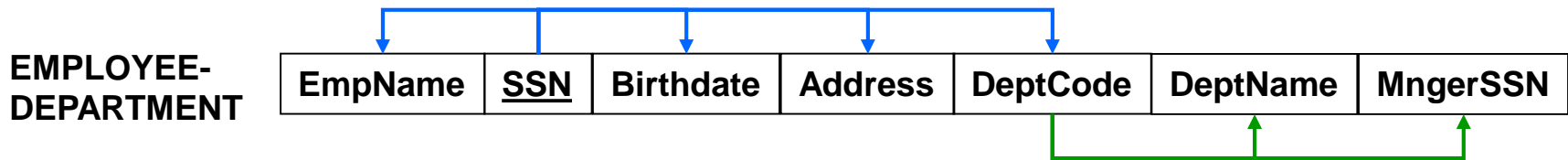
- Clé : {SSN}
- L'attribut non-clé *DepCode* détermine les attributs non-clé *DeptName* et *MngerSSN*

Troisième forme normale (3NF)

◆ Définition : 3NF

- Un schéma de relation R est en 3NF si :
 - il est en 2NF, et
 - aucun attribut non clé dépend transitivement d'une clé
- Dit autrement : **il n'y a pas** de DF non triviale $X \rightarrow A$ où :
 - X n'est pas une surclé de R, et
 - A est un attribut non-clé

◆ Exemple :



- Le schéma est en 2NF
- **DeptCode \rightarrow DeptName MngerSSN** viole 3NF

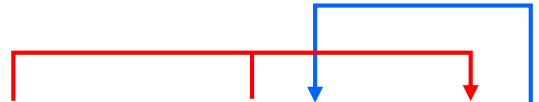
Forme normale de Boyce-Codd (BCNF)

◆ Observation :

- D'autres redondances apparaissent en présence de plusieurs clés, quand une DF ne correspond pas à une clé.

◆ Exemple :

ALLOCATIONS



The diagram shows functional dependencies between the attributes of the table. A red arrow points from 'Student' to 'Course', and another red arrow points from 'Student' to 'Professor'. A blue arrow points from 'Professor' to 'Course'.

Student	Course	Professor
Paul	Math	Martin
Ann	Math	Martin
Ann	Algorithms	Louise

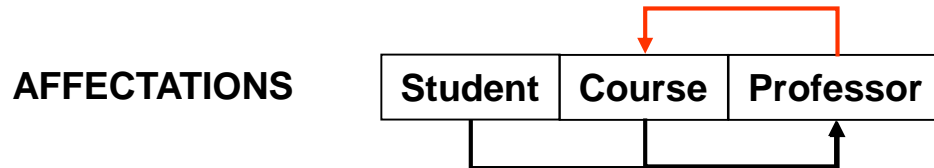
- Clés : (Student, Course), (Student, Professor)
- L'attribut *Professor* n'est pas clé et détermine *Course*

Forme normale de Boyce-Codd (BCNF)

◆ Définition : BCNF

- Un schéma de relation R est en BCNF si :
 - pour toute DF non triviale $X \rightarrow A$, X est une surclé de R
- Dit autrement : il n'y a pas de DF non triviale $X \rightarrow A$ où :
 - X n'est pas une surclé de R

◆ Exemple :



- Le schéma est en 3NF
- $Professor \rightarrow Course$ viole BCNF

Exercice: Test de forme normale

Indiquer en quelle forme normale sont les schémas suivantes :

1. $R1(ABCD)$, $F1 = \{AB \rightarrow C, AB \rightarrow D\}$
2. $R2(ABCG)$, $F2 = \{AB \rightarrow CG, B \rightarrow G\}$
3. $R3(ABDE)$, $F3 = \{A \rightarrow BDE, BD \rightarrow E\}$
4. $R4(BCG)$, $F4 = \{C \rightarrow G\}$
5. $R5(ABCD)$, $F5 = \{AB \rightarrow CD, D \rightarrow B\}$
6. $R6(ABCDEFG)$, $F6 = \{A \rightarrow BC, C \rightarrow DG, BD \rightarrow E, AB \rightarrow D, BC \rightarrow G\}$

Point sur la conception BD

◆ Nous avons vu :

- Ce qui est une décomposition
- Des critères de qualité d'une décomposition :
 - SPI
 - SPD
 - Formes normales

◆ Il nous faut :

- Des algorithmes de normalisation qui nous produisent des décompositions SPI, SPD, dans une forme normale élevée

Plan de la séance



- ◆ **Décomposition de relations**
 - Décomposition sans perte d'information (SPI)
 - Décomposition sans perte de dépendances (SPD)

- ◆ **Formes normales (FN)**

- ◆ **Algorithmes de normalisation**
 - Décomposition en 3NF SPI et SPD
 - Décomposition en BCNF SPI

- ◆ **Pour aller plus loin...**

Décomposition en 3NF SPI et SPD

◆ Entrée :

- Un schéma de relation $R(A_1 \dots A_n)$ qui n'est pas en 3NF
- Un ensemble de DF F sur R

◆ Sortie :

- Une décomposition $\rho = \{R_1 \dots R_k\}$ en 3NF et qui assure SPI et SPD

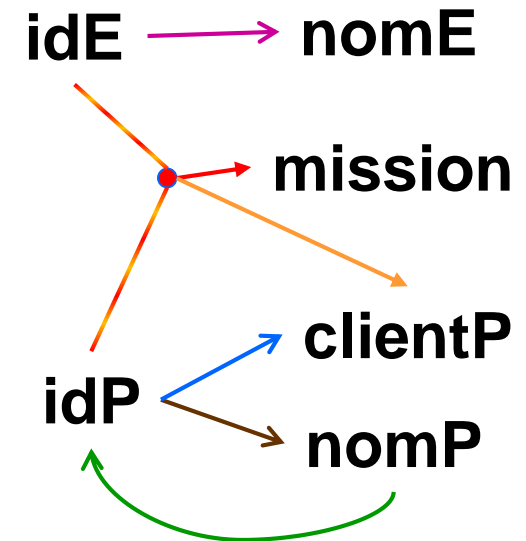
◆ Algorithme :

1. trouver une couverture minimale G de F
2. pour chaque partie gauche X d'une DF
soit $Y = \{B_1 \dots B_j\}$ l'ensemble d'attributs tel que $X \rightarrow B_i \in G$
créer un schéma (XY)
3. si aucun schéma contient une clé de R , en créer un

Exemple : Décomposition en 3NF SPI SPD

- AFFECTATION (idE, nomE, idP, nomP, clientP, mission)
 - $F = \{ \text{idE} \rightarrow \text{nomE}, \text{idP} \rightarrow \text{nomP}, \text{idP} \rightarrow \text{clientP}, \text{nomP} \rightarrow \text{idP}, \text{idE idP} \rightarrow \text{mission}, \text{idE idP} \rightarrow \text{clientP} \}$

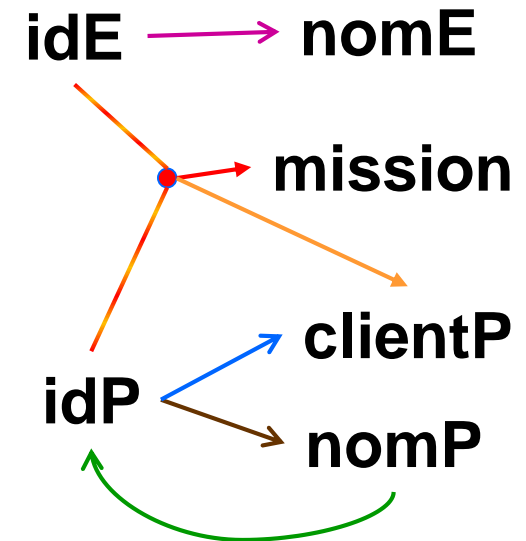
idE	nomE	idP	nomP	clientP	mission
1	Dupond	1	Web-23	EDF	Chef projet
1	Dupond	2	Web-24	CAF	Chef projet
2	Smith	1	Web-23	EDF	Développeur
2	Smith	3	BD-35	Dupond	Intégrateur
3	Dupond	1	Web-23	EDF	Graphiste
4	Li	3	BD-35	Dupond	Technicien



- Dans quelle forme normale est AFFECTATION ?
- Quelles sont ses clés ?

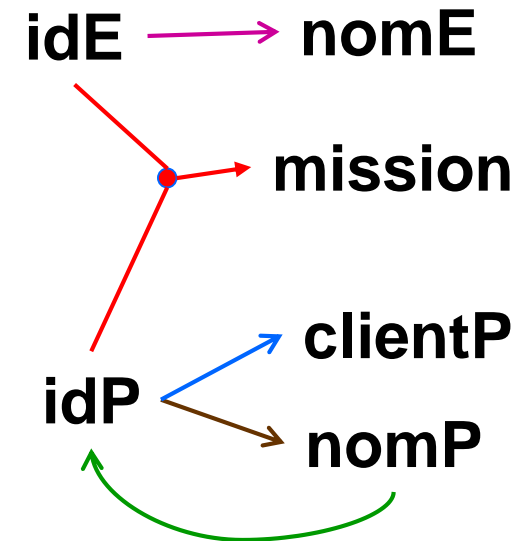
Exemple : Décomposition en 3NF SPI SPD

- AFFECTATION (idE, nomE, idP, nomP, clientP, mission)
 - $F = \{idE \rightarrow nomE, idP \rightarrow nomP, idP \rightarrow clientP, nomP \rightarrow idP, idE \ idP \rightarrow mission, idE \ idP \rightarrow clientP\}$
 - Clés : (idE, idP), (idE, nomP)
 - 1NF
- Pas 1: couverture minimale
 - $F' = \{idE \rightarrow nomE, idP \rightarrow nomP, idP \rightarrow clientP, nomP \rightarrow idP, idE \ idP \rightarrow mission\}$



Exemple : Décomposition en 3NF SPI SPD

- AFFECTATION (idE, nomE, idP, nomP, clientP, mission)
 - $F' = \{ \text{idE} \rightarrow \text{nomE}, \text{idP} \rightarrow \text{nomP}, \text{idP} \rightarrow \text{clientP}, \text{nomP} \rightarrow \text{idP}, \text{idE idP} \rightarrow \text{mission} \}$
 - Clés : (idE, idP), (idE, nomP)
- Pas 2: créer les relations
 - EMPLOYES (idE, nomE)
 - $F_{\text{emp}} = \{ \text{idE} \rightarrow \text{nomE} \}$
 - Clé : idE
 - PROJETS (idP, nomP, clientP)
 - $F_{\text{proj}} = \{ \text{idP} \rightarrow \text{nomP}, \text{idP} \rightarrow \text{clientP}, \text{nomP} \rightarrow \text{idP} \}$
 - Clés : (idP), (nomP)
 - AFFECTATION (idE, idP, mission)
 - $F_{\text{aff}} = \{ \text{idE idP} \rightarrow \text{mission} \}$
 - Clé : (idE, idP)
- Observation : on ne crée pas R (nomP, idP). Pourquoi ?

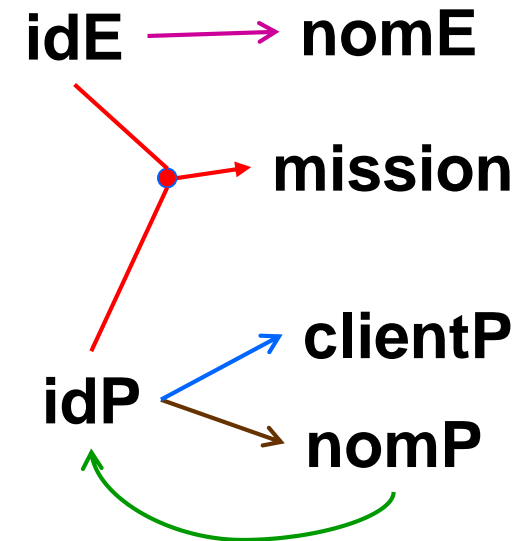


Exemple : Décomposition en 3NF SPI SPD

- AFFECTATION (idE, nomE, idP, nomP, clientP, mission)
 - $F' = \{idE \rightarrow nomE, idP \rightarrow nomP, idP \rightarrow clientP, nomP \rightarrow idP, idE \ idP \rightarrow mission\}$
 - Clés : (idE, idP), (idE, nomP)

- Décomposition

- EMPLOYES (idE, nomE)
 - $F_{emp} = \{idE \rightarrow nomE\}$
 - Clé : idE
- PROJETS (idP, nomP, clientP)
 - $F_{proj} = \{idP \rightarrow nomP, idP \rightarrow clientP, nomP \rightarrow idP\}$
 - Clés : (idP), (nomP)
- AFFECTATION (idE, idP, mission)
 - $F_{aff} = \{idE \ idP \rightarrow mission\}$
 - Clé : (idE, idP)

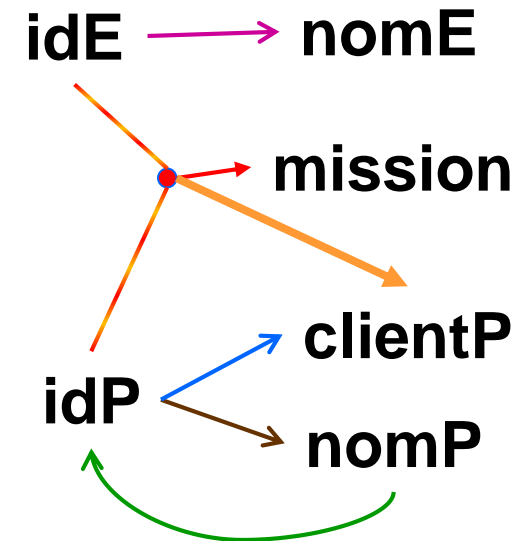


- Pas 3: vérifier que les relations contiennent une clé



Exemple : Décomposition en 3NF SPI SPD

- AFFECTATION (idE, nomE, idP, nomP, clientP, mission)
 - $F = \{\text{idE} \rightarrow \text{nomE}, \text{idP} \rightarrow \text{nomP}, \text{idP} \rightarrow \text{clientP}, \text{nomP} \rightarrow \text{idP}, \text{idE idP} \rightarrow \text{mission}, \text{idE idP} \rightarrow \text{clientP}\}$
 - Clés : (idE, idP), (idE, nomP)
- **Et si on ne calcule pas une couverture minimale ?**
- On obtient :
 - EMPLOYES (idE, nomE)
 - $F_{\text{emp}} = \{\text{idE} \rightarrow \text{nomE}\}$
 - Clé : idE
 - PROJETS (idP, nomP, clientP)
 - $F_{\text{proj}} = \{\text{idP} \rightarrow \text{nomP}, \text{idP} \rightarrow \text{clientP}, \text{nomP} \rightarrow \text{idP}\}$
 - Clés : (idP), (nomP)
 - AFFECTATION (idE, idP, mission, **clientP**)
 - $F_{\text{aff}} = \{\text{idE idP} \rightarrow \text{mission}, \text{idE idP} \rightarrow \text{clientP}, \text{idP} \rightarrow \text{clientP}\}$
 - Clé : (idE, idP)

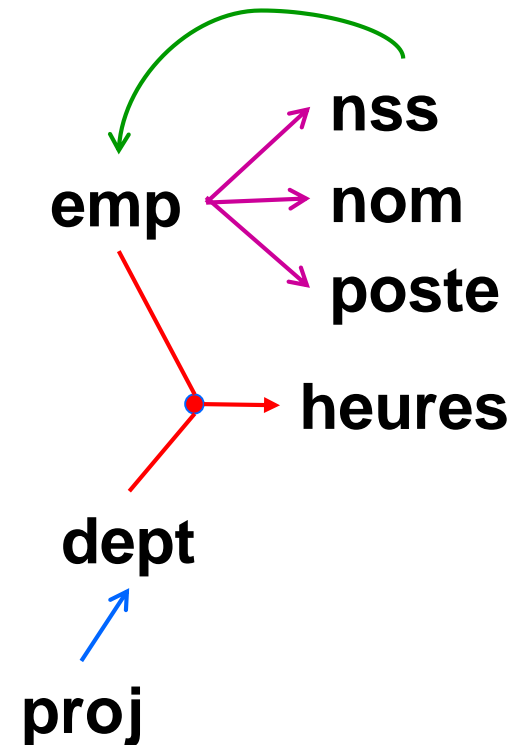


1NF !!!

Exercice : Mettre en 3NF SPI SPD

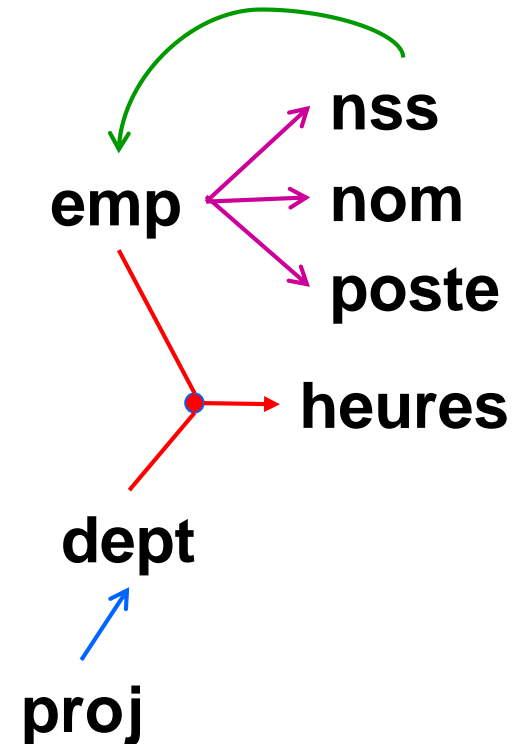
- AFFECTATION (emp, nss, nom, poste, proj, dept, heures)
 - $F = \{\text{emp} \rightarrow \text{nss nom poste}, \text{nss} \rightarrow \text{emp}, \text{proj} \rightarrow \text{dept}, \text{emp dept} \rightarrow \text{heures}\}$

emp	nss	nom	poste	proj	dept	heures
1	1750...	Dupond	Chef projet	1	web	35
2	2691...	Smith	Chef projet	2	bi	25
2	2691...	Smith	Chef projet	3	bi	10
3	2881...	Smith	Consultant	1	web	20
3	2881...	Smith	Consultant	2	bi	15
4	1990...	Lopés	Stagiaire	3	bi	20
5	2740...	Li	Directeur	0	dir	35



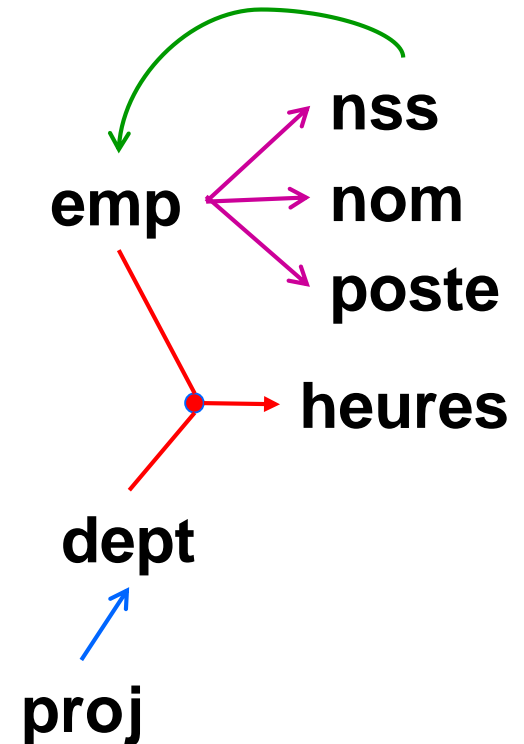
Exercice : Mettre en 3NF SPI SPD

- AFFECTATION (emp, nss, nom, poste, proj, dept, heures)
 - $F = \{\text{emp} \rightarrow \text{nss nom poste}, \text{nss} \rightarrow \text{emp}, \text{proj} \rightarrow \text{dept}, \text{emp dept} \rightarrow \text{heures}\}$
 - Clés : (emp, proj), (emp, nss) **1NF**
- Pas 1: couverture minimale
 - F est déjà minimal



Exercice : Mettre en 3NF SPI SPD

- AFFECTATION (emp, nss, nom, poste, proj, dept, heures)
 - $F = \{\text{emp} \rightarrow \text{nss nom poste}, \text{nss} \rightarrow \text{emp}, \text{proj} \rightarrow \text{dept}, \text{emp dept} \rightarrow \text{heures}\}$
 - Clés : (emp, proj), (emp, nss)
- Pas 2: créer les relations
 - EMPLOYES (emp, nss, nom, poste)
 - $F_{\text{emp}} = \{\text{emp} \rightarrow \text{nss nom poste}, \text{nss} \rightarrow \text{emp}\}$
 - Clés : (emp), (nss)
 - PROJETS (proj, dept)
 - $F_{\text{proj}} = \{\text{proj} \rightarrow \text{dept}\}$
 - Clé : (proj)
 - HEURES (emp, dept, heures)
 - $F_{\text{heu}} = \{\text{emp dept} \rightarrow \text{heures}\}$
 - Clé : (emp, dept)
- Pas 3: vérifier que les relations contiennent une clé
 - AFFECTATION (emp, proj)
 - $F_{\text{aff}} = \{\}$; Clé : (emp, proj)



Exercice : Mettre en 3NF SPI SPD

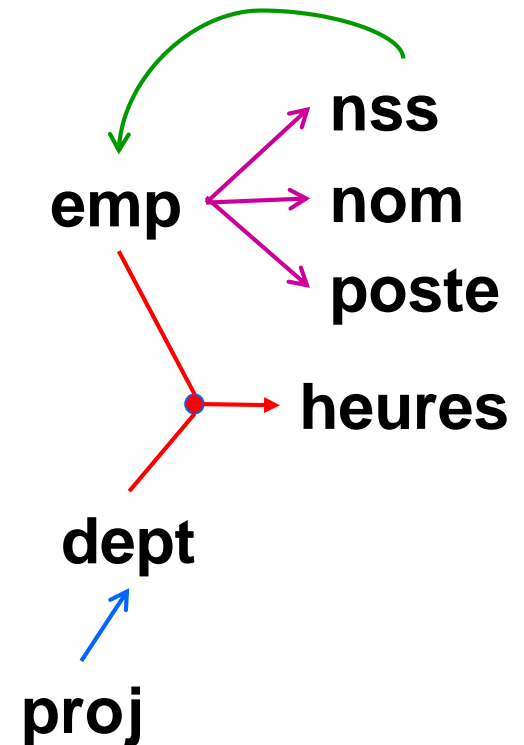
– AFFECTATION (emp, nss, nom, poste, proj, dept, heures)

- $F = \{\text{emp} \rightarrow \text{nss nom poste}, \text{nss} \rightarrow \text{emp}, \text{proj} \rightarrow \text{dept}, \text{emp dept} \rightarrow \text{heures}\}$

emp	nss	nom	poste	proj	dept	heures
1	1750...	Dupond	Chef projet	1	web	35
2	2691...	Smith	Chef projet	2	bi	25
2	2691...	Smith	Chef projet	3	bi	10
3	2881...	Smith	Consultant	1	web	20
3	2881...	Smith	Consultant	2	bi	15
4	1990...	Lopés	Stagiaire	3	bi	20
5	2740...	Li	Directeur	0	dir	35

– **Devoir :**

- Calculer l'instance d'EMPLOYES, PROJETS, HEURES
- Calculer les jointures pour montrer que ce n'est pas SPI
- Faire du même en incluant AFFECTATION



Décomposition en BCNF SPI

◆ Entrée :

- Un schéma de relation $R(A_1 \dots A_n)$ qui n'est pas en BCNF
- Un ensemble de DF F sur R

◆ Sortie :

- Une décomposition $\rho = \{R_1 \dots R_k\}$ en BCNF et qui assure SPI

◆ Algorithme :

$\rho = \{ R \}$

repeter

soit $Q \in \rho$ un schéma qui n'est pas en BCNF

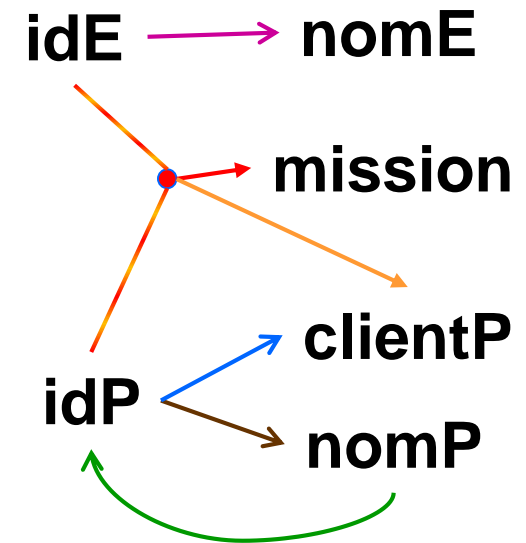
soit $X \rightarrow Y$ une DF en Q qui viole BCNF

remplacer Q en ρ par deux schémas : $(Q \setminus Y)$ et (XY)

jusque ρ soit en BCNF

Exemple : Décomposition en BCNF SPI

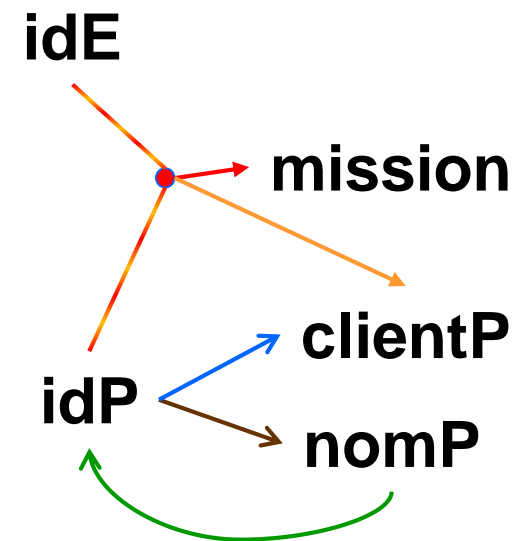
- AFFECTATION (idE, nomE, idP, nomP, clientP, mission)
 - $F = \{idE \rightarrow nomE, idP \rightarrow nomP, idP \rightarrow clientP, nomP \rightarrow idP, idE \ idP \rightarrow mission, idE \ idP \rightarrow clientP\}$
 - Clés : (idE, idP), (idE, nomP)
 - 1NF
- Observation : Pas besoin de couverture minimale
- $idE \rightarrow nomE$ viole BCNF
 - EMPLOYES (idE, nomE)
 - $F_{emp} = \{idE \rightarrow nomE\}$
 - Clé : idE
 - AFFECTATION (idE, idP, nomP, clientP, mission)
 - $F_{aff} = \{idP \rightarrow nomP, idP \rightarrow clientP, nomP \rightarrow idP, idE \ idP \rightarrow mission, idE \ idP \rightarrow clientP\}$
 - Clés : (idE, idP), (idE, nomP)



Exemple : Décomposition en BCNF SPI

- EMPLOYES (idE, nomE)
 - $F_{\text{emp}} = \{\text{idE} \rightarrow \text{nomE}\}$ **BCNF**
 - Clé : idE
- AFFECTATION (idE, idP, nomP, clientP, mission)
 - $F = \{\text{idP} \rightarrow \text{nomP}, \text{idP} \rightarrow \text{clientP}, \text{nomP} \rightarrow \text{idP}, \text{idE idP} \rightarrow \text{mission}, \text{idE idP} \rightarrow \text{clientP}\}$ **1NF**
 - Clés : (idE, idP), (idE, nomP)
- $\text{idP} \rightarrow \text{nomP}$ viole BCNF
 - PROJETS (idP, nomP)
 - $F_{\text{proj}} = \{\text{idP} \rightarrow \text{nomP}, \text{nomP} \rightarrow \text{idP}\}$
 - Clés : (idP), (nomP)
 - AFFECTATION (idE, idP, clientP, mission)
 - $F_{\text{aff}} = \{\text{idP} \rightarrow \text{clientP}, \text{idE idP} \rightarrow \text{mission}, \text{idE idP} \rightarrow \text{clientP}\}$
 - Clé : (idE, idP)

$\text{idE} \rightarrow \text{nomE}$



Exemple : Décomposition en BCNF SPI

- EMPLOYES (idE, nomE)
 - $F_{\text{emp}} = \{\text{idE} \rightarrow \text{nomE}\}$
 - Clé : idE
- PROJETS (idP, nomP)
 - $F_{\text{proj}} = \{\text{idP} \rightarrow \text{nomP}, \text{nomP} \rightarrow \text{idP}\}$
 - Clés : (idP), (nomP)
- AFFECTATION (idE, idP, nomP, clientP, mission)
 - $F = \{\text{idP} \rightarrow \text{clientP}, \text{idE idP} \rightarrow \text{mission}, \text{idE idP} \rightarrow \text{clientP}\}$
 - Clé : (idE, idP)
- $\text{idP} \rightarrow \text{clientP}$ viole BCNF
 - CLIENTS-PROJ (idP, clientP)
 - $F_{\text{cli}} = \{\text{idP} \rightarrow \text{clientP}\}$
 - Clé : (idP)
 - AFFECTATION (idE, idP, mission)
 - $F_{\text{aff}} = \{\text{idE idP} \rightarrow \text{mission}\}$
 - Clé : (idE, idP)

BCNF

BCNF

1NF

$\text{idE} \rightarrow \text{nomE}$

$\text{idP} \rightarrow \text{nomP}$

$\text{idE idP} \rightarrow \text{mission}$
 $\text{idP} \rightarrow \text{clientP}$

Exemple : Décomposition en BCNF SPI

- EMPLOYES (idE, nomE)
 - $F_{\text{emp}} = \{\text{idE} \rightarrow \text{nomE}\}$
 - Clé : idE
- PROJETS (idP, nomP)
 - $F_{\text{proj}} = \{\text{idP} \rightarrow \text{nomP}, \text{nomP} \rightarrow \text{idP}\}$
 - Clés : (idP), (nomP)
- CLIENTS-PROJ (idP, clientP)
 - $F_{\text{cli}} = \{\text{idP} \rightarrow \text{clientP}\}$
 - Clé : (idP)
- AFFECTATION (idE, idP, mission)
 - $F = \{\text{idE idP} \rightarrow \text{mission}\}$
 - Clé : (idE, idP)

BCNF

idE \rightarrow **nomE**

BCNF

idP \rightarrow **nomP**

nomP \rightarrow **idP**

BCNF

idP \rightarrow **clientP**

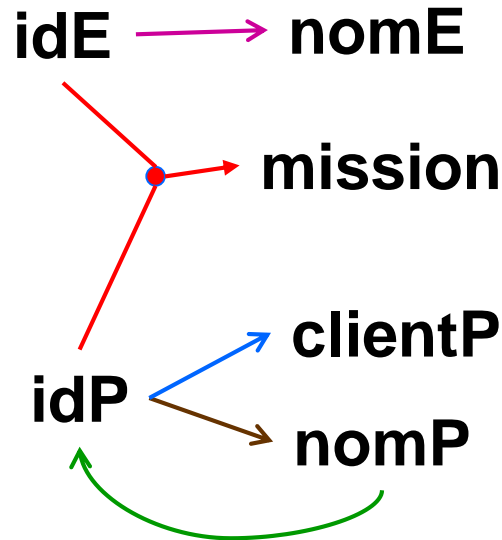
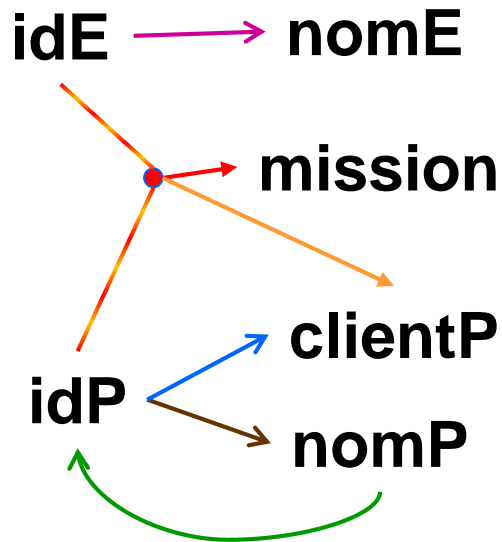
BCNF

idE \rightarrow **mission**

idP \rightarrow **mission**

Exemple : Décomposition en BCNF SPI

- La décomposition préserve les DF ?



$idE \rightarrow nomE$

$idP \rightarrow nomP$

$idP \rightarrow clientP$

$idE \rightarrow mission$

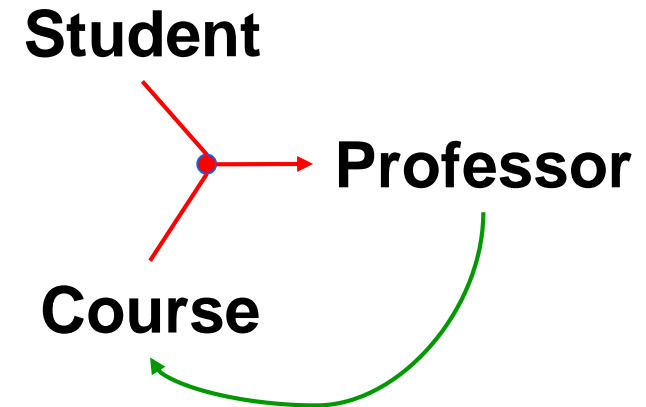
BCNF vs. 3NF

♦ Observation :

- L'algorithme de décomposition en BCNF est SPI mais on peut perdre des dépendances
- Pour assurer SPD, il faut sacrifier un niveau de normalisation et se contenter avec 3NF
- Dans 3NF il peut y avoir un peu de redondance, donc **il faudra vérifier certaines DF lors des mises à jour**
 - Il est convenable de bien les documenter, pour aider la tâche de l'administrateur BD lors de l'implémentation

Exemple : BCNF vs. 3NF

- AFFECTATION (Student, Course, Professor)
 - $F = \{\text{Student Course} \rightarrow \text{Professor}, \text{Professor} \rightarrow \text{Course}\}$
 - Clés : (Student, Course), (Student, Professor)
 - 3NF
- $\text{Professor} \rightarrow \text{Course}$ viole BCNF
 - PROF-COURSE (Professor, Course)
 - $F_{\text{cli}} = \{\text{Professor} \rightarrow \text{Course}\}$
 - Clé : (Professor) **BCNF**
 - PROF-STUDENT (Student, Professor)
 - $F_{\text{aff}} = \{\}$
 - Clé : (Student, Professor) **BCNF**
- On perd une dépendance fonctionnelle



Student	Course	Professor
Paul	Math	Martin
Ann	Math	Martin
Claire	Math	Jean
Claire	Algorithms	Louise

Exemple : BCNF vs. 3NF

3NF

- AFFECTATION (Student, Course, Professor)
 - $F = \{\text{Student Course} \rightarrow \text{Professor}, \text{Professor} \rightarrow \text{Course}\}$
 - Clés : (Student, Course), (Student, Professor)
 - 3NF

Student	Course	Professor
Paul	Math	Martin
Ann	Math	Martin
Claire	Math	Jean
Claire	Algorithms	Louise

BCNF

- PROF-COURSE (Professor, Course)
 - $F_{\text{cli}} = \{\text{Professor} \rightarrow \text{Course}\}$
 - Clé : (Professor)
- PROF-STUDENT (Student, Professor)
 - $F_{\text{aff}} = \{\}$
 - Clé : (Student, Professor)

Student	Professor
Paul	Martin
Ann	Martin
Claire	Jean
Claire	Louise

Professor	Course
Martin	Math
Jean	Math
Louise	Algorithms

Ça se joue à quelle DF est plus importante

Plan de la séance



- ◆ **Décomposition de relations**
 - Décomposition sans perte d'information (SPI)
 - Décomposition sans perte de dépendances (SPD)
- ◆ **Formes normales (FN)**
- ◆ **Algorithmes de normalisation**
 - Décomposition en 3NF SPI et SPD
 - Décomposition en BCNF SPI
- ◆ **Pour aller plus loin...**

Attributs multivalués

♦ Motivation :

- ETUDIANTS (idE, tel, adresse)
 - Chaque étudiant peut avoir **plusieurs** téléphones et **plusieurs** adresses
 - Chaque téléphone (fixe) **est associé** à une adresse,
 - pas les portables
- Modélisation classique :

IdE	tel	adresse
1	02 54 02 54 02	8 r Papin, Blois
1	02 36 12 34 56	17 r Breton, Tours
2	02 54 99 99 99	3 pl Jaurès, Blois
3	02 02 02 02 02	1 av Europe, Blois
3	02 04 06 08 10	55 av Paris, Blois

Attributs multivalués

♦ Motivation :

- ETUDIANTS (idE, tel, email)
 - Chaque étudiant peut avoir **plusieurs** téléphones et **plusieurs** emails.
 - Il n'y a **pas de relation** entre les téléphones et les emails.
- Laquelle de ces modélisations vous semble la mieux ?

Ex. Quels sont les emails de l'étudiant qui a le tel. t1 ?

```
SELECT email FROM Etudiants  
where tel = t1;
```

IdE	tel	email
1	t1	e1
1	t2	e2
2	t3	e3
2	t4	

Màj plus difficile
Sémantique trompeuse
Requêtes plus difficiles

IdE	tel	email
1	t1	e1
1	t1	e2
1	t2	e1
1	t2	e2
2	t3	e3
2	t4	e3

Màj plus difficile
Redondance

Et le tel. t4 ?

Attributs multivalués

♦ Motivation :

- ETUDIANTS (idE, tel, email)
 - Chaque étudiant peut avoir **plusieurs** téléphones et **plusieurs** emails.
 - Il n'y a **pas de relation** entre les téléphones et les emails.
- **Solution intuitive :**

IdE	tel
1	t1
1	t2
2	t3
2	t4

IdE	email
1	e1
1	e2
2	e3

Ex. Quels sont les emails de l'étudiant qui a le tel. t1 ?

```
SELECT email FROM TEL-ETUD, EMAIL-ETUD  
where TelEtud.IdE = EmailEtud.IdE  
and tel = t1;
```


Dépendances multivaluées (DMV)

♦ Idée intuitive

- Une DF $X \rightarrow A$ impose que pour chaque valeur de X il doit y avoir une seule valeur de A
- Une DMV $X \twoheadrightarrow Y$ indique que pour chaque valeur de X il peut avoir plusieurs valeurs de Y, **indépendamment** des valeurs des autres attributs

♦ Il y a une théorie similaire à celle des DF

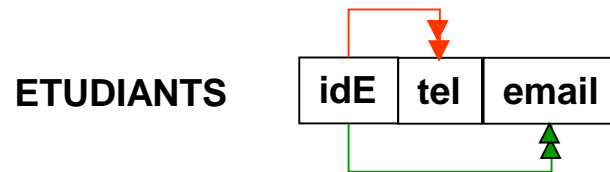
- Définitions, axiomes, dérivations, projection de dépendances, algorithmes...

Quatrième forme normale (4NF)

◆ Définition : 4NF

- Un schéma de relation R est en 4NF s'il n'y a pas DMV non triviales
 - Une DMV $X \twoheadrightarrow Y$ est triviale si $Y \subseteq X$ ou $X \cup Y = \text{sort}(R)$

◆ Exemple :



- Clé : (idE, tel, email)
- **idE** \twoheadrightarrow **tel**, **idE** \twoheadrightarrow **email** violent 4NF

Décomposition en 4NF SPI

◆ Entrée :

- Un schéma de relation $R(A_1 \dots A_n)$ qui n'est pas en 4NF
- Un ensemble de DMV D sur R

◆ Sortie :

- Une décomposition $\rho = \{R_1 \dots R_k\}$ en 4NF et qui assure SPI

◆ Algorithme :

$\rho = \{ R \}$

repeter

soit $Q \in \rho$ un schéma qui n'est pas en 4NF

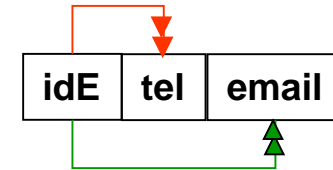
soit $X \twoheadrightarrow Y$ une DMV en Q qui viole 4NF

remplacer Q en ρ par deux schémas : $(Q \setminus Y)$ et (XY)

jusque ρ soit en 4NF

Exemple : Décomposition en 4NF SPI

- ETUDIANTS (idE, tel, email)
 - $F = \{ \}$
 - $M = \{ \text{idE} \twoheadrightarrow \text{tel}, \text{idE} \twoheadrightarrow \text{email} \}$
 - Clé : (idE, tel, email)
 - BCNF



- $\text{idE} \twoheadrightarrow \text{tel}$ viole 4NF
 - TEL-ETUD (idE, tel)
 - $M_{\text{tel}} = \{ \text{idE} \twoheadrightarrow \text{tel} \}$
 - Clé : (idE, tel)
 - EMAIL-ETUD (idE, email)
 - $M_{\text{email}} = \{ \text{idE} \twoheadrightarrow \text{email} \}$
 - Clé : (idE, email)

4NF

4NF

Pour aller encore plus loin...

◆ D'autres formes normales existent : 5NF, 6NF, ...

- Elles contrôlent des cas très particuliers de redondances
- Elles restent théoriques

◆ En pratique :

- Les méthodologies de modélisation des SI tiennent compte jusqu'à la 4FN

Résumé

◆ Nous avons vu :

- L'importance des contraintes d'intégrité
- Des techniques pour les définir (DF) et les contrôler (normalisation)
- Des algorithmes largement utilisés pour normaliser

◆ La normalisation est un thème très important pour la conception BD

- Evite la redondance et les anomalies de mis à jour
- Permet de contrôler automatiquement la satisfaction des contraintes (grâce aux clés)

Réflexion sur les contraintes

♦ Les contraintes sont vérifiées automatiquement

- A chaque ajout, modification ou suppression de données dans la base.
- En cas de non respect de la contrainte, l'action demandée est refusée.

♦ En tant que DBA :

- Votre rôle est de concevoir, d'implanter et de faire respecter des règles de bonne gestion.
- Il est normal que des besoins de maintenance conduisent à désactiver momentanément des contraintes.
- Il est possible que des besoins d'optimisation des performances amènent à dé-normaliser et/ou retirer des contraintes :
 - Cela se fait après analyse des utilisations, pas a priori.
 - Les contraintes retirés doivent être contrôlés manuellement (ex. via triggers) ou assurés (ex. via des mis à jours contrôlés) pour éviter des anomalies