M1 · Bases de données avancées

TD3 Dépendances fonctionnelles

Un schéma relationnel $R(A_1, \ldots, A_n)$ est composé d'un nom de la relation R et d'une suite $(A_1, \ldots, A_n)$ d'attributs.

Un tuple de type R est une fonction

$$t : \{A_1, \ldots, A_n\} \rightarrow Val$$

de l'ensemble d'attributs $\{A_1, \ldots, A_n\}$ vers l'ensemble des valeurs Val,
nous notons par $t(A_i)$ la valeur de l'attribut A_i du tuple t.
Pour simplifier nous allons ignorer le fait qu'en SQL la valeur $t(A_i)$ d'un attribut A_i peut être *null*.

Une relation r de type R (ou une relation conforme au schéma R) est un ensemble de tuples de type R.

Un schéma d'une base de donnée est un ensemble de schémas relationnels.

D'habitude notre base de données doit satisfaire un certain nombre de contraintes et on considère que les relations sont valides si elles satisfont toutes les contraintes.

Nous allons étudier d'abord des contraintes données par des dépendances fonctionnelles.

Contraintes données par des dépendances fonctionnelles

❖ Pour un tuple t de type $R(A_1, \ldots, A_n)$ et un ensemble non vide $X \subset \{A_1, \ldots, A_n\}$ d'attributs on note $\pi_X(t)$ la projection de t sur X.

- Donc $t' = \pi_X(t)$ est un tuple avec les attributs X tel que, pour chaque $A_i \in X$, $t'(A_i) = t(A_i)$.

❖ Soit X, Y deux ensembles non vides d'attributs $, X, Y \subseteq \{A_1, \ldots, A_n\}$ de $R(A_1, \ldots, A_n)$.

- Une relation r de type R satisfait une dépendance fonctionnelle $X \rightarrow Y$ si, pour tous les tuples t_1, t_2 de r, si $\pi_X(t_1) = \pi_X(t_2)$ alors $\pi_Y(t_1) = \pi_Y(t_2)$

❖ Pour alléger la notation nous utiliserons souvent deux conventions:

- Si $\{A_{i_1}, \ldots, A_{i_k}\}$ et $\{B_{j_1}, \ldots, B_{j_m}\}$ deux ensembles d'attributs alors nous écrivons

$$A_{i_1} \ldots A_{i_k} \longrightarrow B_{j_1} \ldots B_{j_m}$$

pour noter la dépendance

$$\{A_{i_1}, \ldots, A_{i_k}\} \longrightarrow \{B_{j_1}, \ldots, B_{j_m}\}.$$

- De plus si X et Y sont deux ensembles d'attributs alors nous écrirons souvent XY pour désigner l'union $X \cup Y$ de X et Y.

Source :

Bases de données avancées - Normalisation / Wieslaw Zielonka

https://www.irif.fr/~zielonka/Enseignement/BDAvances/2012/NORMALISATION/normalisation.pdf

Dépendances fonctionnelles : Cours Normalisation diapos 8 - 13

Contraintes d'intégrité et dépendances

Vers une définition formelle de "qualité" d'un schéma relationnel

Contrainte d'intégrité sur un schéma

Une propriété que les instances du schéma sont censées satisfaire pour être valides

- e.g. contrainte de clef: NSS est une clef pour la relation Personne (NSS, nom, adresse)
- c'est la réalité qu'on modélise qui impose les contraintes

Le processus de modélisation doit identifier non-seulement les informations à représenter, mais également les contraintes qui existent sur celles-ci

⇒ Notre point de départ : un schéma relationnel (potentiellement à raffiner) avec un ensemble de contraintes identifiées

Dépendances fonctionnelles

Exemple

J

V#	Vnom	Vville	P#	Pnom	Pville	Qte
3	MagicV	Paris	322	manteau	Lille	2
1	StarV	Rome	546	veste	Rome	1
3	MagicV	Paris	322	manteau	Lille	5
2	IdealB	Lyon	145	jupe	Paris	7
2	IdealB	Lyon	234	jupe	Lille	1

J satisfait V# → Vnom Vville et P# → Pnom Pville

*A chaque fois que j'ai le même **V#**, j'ai le même **Vnom** et le même **Vville**.*

Contraintes d'intégrité et dépendances

- Dépendances fonctionnelles : Une forme particulière de contraintes d'intégrité
- Exemple

  Schéma : R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

  Un ensemble de dépendances fonctionnelles qu'on peut raisonnablement supposer :

  V# → Vnom Vville
  P# → Pnom Pville
  V# P# → Qte

- Sémantique (intuition) : pour qu'une instance J de la relation R soit valide, J doit satisfaire :
  - si deux tuples dans J ont la même valeur de V#
    alors ils ont la même valeurs de Vnom et de Vville
  - si deux tuples dans J ont la même valeur de P#
    alors ils ont la même valeurs de Pnom et de Pville
  - si deux tuples dans J ont la même valeur de V# et la même valeur de P#
    alors ils ont la même valeurs de Qte

Dépendances fonctionnelles

Exemple

J

V#	Vnom	Vville	P#	Pnom	Pville	Qte
3	MagicV	Paris	322	manteau	Lille	2
1	StarV	Rome	546	veste	Rome	1
3	MagicV	Paris	322	manteau	Lille	5
2	IdealB	Lyon	145	jupe	Paris	7
2	IdealB	Lyon	234	jupe	Lille	1

J viole V# P# → Qte

Cependant, il y a une dépendance qui est violé : le vendeur 3 va fournir le produit X en quantité a et quantité b → incohérence .

V# P# ⟶ quantité

ca va satisfaire une forme normale Boyce-Codd.

Ces dépendances peuvent impliquer d'autres qu'on connait pas... On va voir un algorithme pour calculer les dépendances projeter...

Un schéma relationnel $R(A_1, \ldots, A_n)$ est composé d'un nom de la relation R et d'une suite $(A_1, \ldots, A_n)$ d'attributs.

$R(U)$

Schéma de relation

Univers
Je prends tous les attributs

Dépendances fonctionnelles (DF)

Soit R(U) un schéma de relation avec U : ensemble d'attributs

Une dépendance fonctionnelle est une expression : $X \to Y$, avec $X, Y \subseteq U$

Une instance J de R(U) satisfait $X \to Y$ si pour toute paire de tuples t, u dans J

$t[X] = u[X] \Rightarrow t[Y] = u[Y]$

(si t et u sont en accord sur X alors t et u sont en accord sur Y)

Soit X, Y deux ensembles non vides d'attributs , $X, Y \subseteq \{A_1, \ldots, A_n\}$ de $R(A_1, \ldots, A_n)$.

Pour un tuple t de type $R(A_1, \ldots, A_n)$ et un ensemble non vide $X \subset \{A_1, \ldots, A_n\}$ d'attributs on note $\pi_X(t)$ la projection de t sur X.

		X				Y				
U :		A_1	…	A_m		B_1	…	B_n		
t										
		a_1	…	a_m		b_1	…	b_1		
u										
		a_1	…	a_m		b_1	…	b_1		

J satisfait un ensemble F de DF, si J satisfait chaque DF dans F

$X \to Y$ si, pour tous les tuples t, u , si $\pi_X(t) = \pi_X(u)$ alors $\pi_Y(t) = \pi_Y(u)$

Remarque sur la notation.

Par la suite un ensemble d'attributs {A1,…,An} sera dénoté par A1…An

Pour alléger la notation

Donc A1…An → B1..Bn dénotera la DF {A1, …,An} → {B1, …, Bn}

Remarque sur le vocabulaire

X → Y en mots :"X determine Y"

De plus si X et Y sont deux ensembles d'attributs , alors nous écrirons souvent XY pour désigner l'union $X \cup Y$ de X et Y .

Si $\{A_{i_1}, \ldots, A_{i_k}\}$ et $\{B_{j_1}, \ldots, B_{j_m}\}$ deux ensembles d'attributs alors nous écrivons $A_{i_1} \ldots A_{i_k} \longrightarrow B_{j_1} \ldots B_{j_m}$ pour noter la dépendance $\{A_{i_1}, \ldots, A_{i_k}\} \longrightarrow \{B_{j_1}, \ldots, B_{j_m}\}$.

Source des commentaires : *Bases de données avancées - Normalisation / Wieslaw Zielonka*

https://www.irif.fr/~zielonka/Enseignement/BDAvances/2012/NORMALISATION/normalisation.pdf

Exercice 1 : Compléter des tuples au vu des dépendances fonctionnelles

On considère une relation $R(A, B, C, D, E)$ qui vérifie l'ensemble des dépendances fonctionnelles $\mathcal{F} = \{AB \rightarrow D, C \rightarrow D, D \rightarrow E\}$

Complétez les tuples suivants de la relation R avec des valeurs non nulles au vu de ces dépendances fonctionnelles.

A	B	C	D	E
X	x	A	a	1
X	x	B	a	1
Y	y	B	a	1
Y	t	B	a	1
Z	t	C	b	2
Z	x	A	a	1

On commence par ça.

Nos DF :

- AB -> D c-à-d : On aura pas 2 lignes avec le même AB et avec D différent.
- C -> D
- D -> E

Note : On peut avoir des tableaux dans lesquelles il y a des cases qui ne peuvent pas être rempli même si on connait les DF..

Remarque sur la notation :

$$T_1 \ldots . T_k \rightarrow B_1 \ldots . B_m$$

c'est pareil que

$$T_1 \ldots . T_k \rightarrow B_1$$
$$T_1 \ldots . T_k \rightarrow B_2$$
$$\ldots$$
$$T_1 \ldots . T_k \rightarrow B_m$$

Armstrong a proposé trois règles de déductions (**axiomes d'Armstrong**)

Axiomes de Armstrong : Cours Normalisation, diapos 31, 32

Implication de DF : Axiomes de Armstrong

Trois règles d'inférence (dont la correction est facile à vérifier) :

Pour un schéma de relation R(U) , et X,Y, Z ⊆ U

1) *Transitivité :* { X→Y, Y→Z } ⊨ X→Z

2) *Augmentation :* X→Y ⊨ X Z→ Y Z

3) *Réflexivité :* ⊨ XY → X (appelée DF triviale)

Ces règles ne sont pas seulement correctes , il s'agit d'axiomes , i.e

F ⊨ X→Y ssi

X→Y peut être dérivé de F par applications successives des trois règles ci-dessus

La transitivité :

Si X → Y et Y → Z alors X → Z.

la réflexivité :

Si X ⊇ Y alors X → Y

Tout ensemble d'attributs détermine lui-même ou une partie de lui-même.

L'augmentation:

Si X → Y alors XZ → Y Z .

Si X détermine Y, les deux ensembles d'attributs peuvent être enrichis par un même troisième.

Si

$$A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$$

Alors

$$A_1 A_2 \ldots A_n C_1 C_2 \ldots C_k \rightarrow B_1 B_2 \ldots B_m C_1 C_2 \ldots C_k$$

Pour tout ensemble d'attributs $C_1 C_2 \ldots C_k$.

Étant donné que certains des C *peuvent aussi être des* A *ou des* B *ou les deux, nous devrions éliminer les attributs en double du côté gauche et faire de même pour le côté droit.*

Implication de DF : d'autres règles

Plusieurs autres règles correctes, mais pas nécessaires pour former des axiomes (dérivables des axiomes) :

Union : { X→Y, X→Z } ⊨ X→YZ

Séparation : X→YZ ⊨ X→Y

et d'autres encore ...

Comment dériver l'union à partir des 3 axiomes ?

$$\left\{ \underbrace{\underbrace{X \rightarrow Y}_{\substack{Augmentation \\ X \rightarrow XY}}}_{X \rightarrow ZY} \; , \; \underbrace{\underbrace{X \rightarrow Z}_{\substack{Augmentation \\ XY \rightarrow ZY}}}_{X \rightarrow ZY} \right\} \models X \rightarrow YZ$$

Comment dériver la séparation à partir des 3 axiomes ?

$$\underbrace{\underbrace{X \rightarrow YZ}_{\substack{Augmentation \\ X \rightarrow YZX}} \; , \underbrace{X \rightarrow YZX}_{\substack{Réflexivité \\ YZX \rightarrow Y}}}_{\substack{Transitivité \\ X \rightarrow Y}}$$

Source de certains commentaires :

Bases de données / Georges Gardarin / EYROLLES

Database Systems: the Complete Book par H. Garcia-Molina, J.Ullman and J.Widom, Prentice Hall.

Règles qui permettes d'ajouter des DF

Exercice 2 : Les axiomes d'Armstrong

Si on connait ABC, on connait forcément BC

Rappels : Les axiomes d'Armstrong sur les dépendances fonctionnelles sont les suivants :

- *Réflexivité (si $Y \subseteq X$ alors $X \rightarrow Y$)*
- *Augmentation (si $X \rightarrow Y$ alors $XZ \rightarrow YZ$ quel que soit Z),*
- *Transitivité (si $X \rightarrow Y$ et $Y \rightarrow Z$ alors $X \rightarrow Z$)*

« La connaissance de X implique la connaissance de Z »

Soit un ensemble d'attributs ***U*** et ***X***, ***Y***, ***Z*** et ***W*** des sous-ensembles d'attributs de ***U***.

A-t-on les implications logiques suivantes ?

Si oui, démontrez-le à partir des axiomes d'Armstrong ;
si non donnez un contre-exemple sous forme d'un ensemble de tuples.

1. {X → Y ;Z → W} implique XZ → Y W ?

- *L'augmentation :* Si A → B alors AC → BC

  $X \rightarrow Y \models XZ \rightarrow YZ$

  $Z \rightarrow W \models ZY \rightarrow WY$

- *La transitivité :* Si A → B et B → C alors A → C.

  $XZ \rightarrow \mathbf{YZ}$ et $\mathbf{ZY} \rightarrow WY \models \mathbf{XZ \rightarrow Y\,W}$

il est important dans l'examen d'utiliser le signe ⊨ et pas le signe ⇒ car ils ont pas la même signification

YZ et ZY c' est pareil !

2. {XY → Z;Z → X} implique Z → Y ?

Ça a l'air raisonnable ? Intuitivement ca semble correct ? NON.
(Indice : on nous demande une implication inverse..)

Donc : faut donner un contre-exemple (un tableau du même type de tableau vu en exercice1) c-à-d : remplier le tableau de façon à satisfaire les 2 premiers implications, mais pas la 3ème (donc, on a besoin d'au moins une ligne avec le même Z et avec un Y diffèrent). **Méthode pour faire un contre-exemple :** Commencer par écrire d'abord ce qu'on veut que soit faux (ici on veux 2 fois le même Z et 2 Y différents), et ensuite remplir le reste du tableau comme dans l'exercice 1.

X	Y	Z
b	1	a
b	2	a

Implique...

3. {X → Y ; Y → Z} implique X → Y Z ?

Ça a l'air raisonnable, donc je cherche comment le prouver.

Pourquoi ça a l'air raisonnable ?
Si je connait X, je connait Y
Si je connait Y, je connait Z
Donc, c'est raisonable qu'alors X connait Y et Z.

Méthode alternative

$(A)\ Y \rightarrow \mathrm{Z} \ \vDash\ Y \rightarrow \mathrm{ZY}$

$(T)\ \{X \rightarrow \mathrm{Y}\ ,\mathrm{Y}\ \rightarrow \mathrm{ZY}\}$
$\vDash\ \mathbf{X}\ \rightarrow\ \mathbf{Y\ Z}$

X → Y Y→ Z

Transitivité

X → Z et on a déjà X → Y donc par UNION X → YZ

Attention ! l'utilisation de l'UNION nécessite de montrer comment on dérive l'union a partir des 3 axiomes.

4. {X → Y ; W → Z} implique W Y X → Z ?

W Y X → W par réflexivité
on a W → Z
et donc par transition : W Y X → Z

Méthode alternative

$(A)_{*2}\ W \rightarrow \mathrm{Z} \ \vDash\ WYX \rightarrow \mathrm{ZYX}$

$(R)\ Z\ \subseteq \mathrm{ZYX} \ \vDash\ ZYX\ \rightarrow \mathrm{Z}$

$(T)\ \{WYX \rightarrow \mathrm{ZYX}\ , ZYX\ \rightarrow \mathrm{Z}\}$
$\vDash\ \mathbf{W\ Y\ X}\ \rightarrow\ \mathbf{Z}$

5. {W → Y, X → Z} implique W X → Y ?

<u>Méthode 1</u> : WX → YX par augmentation
et on a donc WX → Y par séparation.

<u>Méthode 2</u> : WX → W par réflexivité
on a déjà W → Y et donc par transitivité : W X → Y

6. {X → Y } implique Y ZX → Z ?

Y ZX → Z simplement par réflexivité. Pas besoin d'utiliser X → Y pour prouver ça.

Méthode alternative

$(A)\ \{X \rightarrow Y\} \vDash XY \rightarrow Y$

$(A)\ \{XY \rightarrow Y\} \vDash XYZ \rightarrow YZ$

$(S)\ \{XYZ \rightarrow YZ\} \vDash \mathbf{XYZ \rightarrow Z}$

7. {X → Y, X → W, W Y → Z} implique X → Z ?

X –> Y ⊨ X –> XY *augmentation*

X –> W ⊨ XY –> WY *augmentation*

X –> XY and XY –> WY ⊨ X –> WY *transitivité*

X –> WY and WY –> Z ⊨ X –> Z *transitivité*

Note : On aurai pu écrire que par UNION vu que X → Y et X → W alors **X → YW** et avec le fait qu'on a **W Y → Z** montrer que X → Z par transitivité.

8. {XY → Z, Y → W} implique XW → Z ?

Ça a l'air raisonnable ? NON.
(Indice : w qui change de côté dans l'implication..)

Donc : faut donner un contre-exemple, c-à-d : remplier le tableau de façon à satisfaire les 2 premiers implications, mais pas la 3ème

X	Y	W	Z
x	6	w	1
x	4	w	2

9. {X → Y, XY → Z} implique X → Z ?

- *L'augmentation :* Si A → B alors AC → BC

X –> XY

- *Donnée de l'exercice*

XY –> Z

- *La transitivité :* Si A → B et B → C alors A → C.

X –> Z

Qualité d'un schéma relationnel, diapos 4 - 7

Qualité d'un schéma relationnel

Quelles sont ces "bonnes propriétés" d'un schéma relationnel ?

Exemple:

Attributs relatifs à des vendeurs, produits, et fournitures

V#: numéro de vendeur

Vnom: nom du vendeur

Vville: ville du vendeur

P#: numéro du produit

Pnom: nom du produit

Pville: ville où le produit est stocké

Qte: quantité de produit fournie au vendeur

Qualité d'un schéma relationnel

- Un schéma relationnel possible : une seule relation "fourniture" avec tous les attributs

R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

- C'est une mauvaise modélisation! Pourquoi?

1) Redondance

V#	Vnom	Vville	P#	Pnom	Pville	Qte
3	MagicV	Paris	...	...	...	...
3	MagicV	Paris	...	...	...	...
2	IdealB	Lyon	...	...	...	...
2	IdealB	Lyon	...	...	...	...

Ex: Vnom et Vville sont **déterminés** par V#, i.e.

si deux fournitures ont le même V# , elles ont aussi le même Vville et le même Vnom

On représente l'information que le vendeur 3 est MagicV et qu'il est à Paris, une fois pour chaque fourniture : **redondant**

Le schéma maintenant est mauvais car on a mis tous les attributs dans la même table.

Qualité d'un schéma relationnel

R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

C'est une mauvaise modélisation ! Pourquoi ?

1) Redondance

2) Anomalies de mise à jour

Vnom ou Vville pourrait être mis à jour dans une fourniture et pas dans une autre, ce qui donnerait une incohérence. Pour éviter cela : mise à jour plus coûteuse.

3) Anomalies d'insertion

On ne peut pas stocker un vendeur s'il ne reçoit pas de fourniture

4) Anomalies de suppression

Si on supprime toutes les fournitures d'un vendeur, on perd toute l'info sur ce vendeur

on a « Paris » sur le 1er et 2ème tuple, il pouvait avoir une anomalie de mis à jour si un déménagement sera enregistrer que dans un endroit (On risque d'oublier de mettre à jour a plusieurs places).

Dans ce cas, Ca n'a pas de sens d'insérer un tuple dans la table… On n'appelle ça anomalie d'insertion car il va manquer des informations au niveau des tuples.

Comment voir que il y a des anomalie avec une instance qui n'a pas encore des données ? faut imaginer des données.

Problèmes causés par la redondance

Stocker les mêmes informations de manière ***redondante****, c'est-à-dire à plusieurs endroits dans une base de données, peut entraîner plusieurs problèmes :*

- ***Stockage redondant*** *: certaines informations sont stockées à plusieurs reprises.*
- ***Anomalies de mise à jour*** *: si une copie de ces données répétées est mise à jour, une incohérence est créée à moins que toutes les copies ne soient mises à jour de la même manière.*
- ***Anomalies d'insertion*** *: il peut s'avérer impossible de stocker certaines informations à moins que d'autres informations ne soient également stockées.*
- ***Anomalies de suppression*** *: il peut s'avérer impossible de supprimer certaines informations sans perdre également d'autres informations.*

Source : ***Database Management Systems*** *par Raghu Ramakrishnan and Johannes Gehrke. McGraw-Hill.*

Qualité d'un schéma relationnel

Solution : un "bon" schéma

Vendeur (V#, Vnom, Vville) Clef: V#
Produit (P#, Pnom, Pville) Clef: P#
Fourniture(V#, P#, Qte) Clef: V# P#

Plus d'anomalie! Comment y arriver?

La théorie de la normalisation des bd relationnelles nous donne:

- Des formes normales :
  - propriétés d'un schéma qui garantissent absence (ou réduction) de redondance, et des anomalies qui en dérivent
  - définies par rapport à un ensemble de contraintes (appelés dépendances)

- Des techniques de normalisation : passage d'un schéma arbitraire (mauvais) à un schéma en forme normale (typiquement par décomposition)

Exercice 3 : *Qualité d'un schéma relationnel*

Plusieurs anomalies peuvent se produire lorsque la modélisation n'est pas bien conçue.

Pour chaque modélisation ci-dessous,

- **Identifiez les problèmes de modélisation.**
- **Donnez des exemples de données incohérentes, ou requêtes inefficaces, qui peuvent résulter de la modélisation.**
- **Proposez une nouvelle modélisation qui permet d'éviter ces problèmes.**

1. Les étudiants s'inscrivent et peuvent choisir plusieurs options. On veut limiter le nombre d'inscriptions de façon à bloquer les inscriptions si plus de 40 étudiants s'y inscrivent.
   - (CodeOption, LibelleOption, Descriptif, NombreInscrits)
   - (NumEtudiant, NomEtudiant, PrenomEtudiant)
   - (NumEtudiant, CodeOption)

Contraint externe...

CodeOption	LibelleOption	Descriptif	NombreInscrits
1	Algo	abcde...	3
2	Math	abcd...	1
3	Prog	abc...	1
4	BDD	a...	0

NumEtudiant	NomEtudiant	PrenomEtudiant
1	Abbott	William
2	Costello	Lou
3	Laurel	Stanley
4	Hardy	Oliver
5	Lloyd	Harold
6	Marx	Groucho

NumEtudiant	CodeOption
1	1
2	2
3	3
4	1
5	1

- **Identifiez les problèmes de modélisation.**
  - ***Redondance :*** *le nombre d'inscrits pour chaque option peut être calculé via une requête SQL sur la table « NumEtudiant, CodeOption »*
  - ***Anomalies de mise à jour :*** *si la table « NumEtudiant, CodeOption » est mise à jour, une incohérence est créée à moins que la table d'Options ne soit mis à jour*

de la même manière.

- **Donnez des exemples de données incohérentes, ou requêtes inefficaces, qui peuvent résulter de la modélisation.**

  Anomalies de mise à jour : si la table « NumEtudiant, CodeOption » est mise à jour, une incohérence est créée car la table d'Options n'est pas mis à jour.

CodeOption	LibelleOption	Descriptif	NombreInscrits
1	Algo	abcde...	3
2	Math	abcd...	1
3	Prog	abc...	1
4	BDD	a...	0

NumEtudiant	NomEtudiant	PrenomEtudiant
1	Abbott	William
2	Costello	Lou
3	Laurel	Stanley
4	Hardy	Oliver
5	Lloyd	Harold
6	Marx	Groucho

NumEtudiant	CodeOption
1	1
2	2
3	3
4	1
5	1
1	4

- **Proposez une nouvelle modélisation qui permet d'éviter ces problèmes.**

- (CodeOption, LibelleOption, Descriptif, NombreInscrits)
- (NumEtudiant, NomEtudiant, PrenomEtudiant)
- (NumEtudiant, CodeOption)

2. Chaque étudiant s'inscrit à une seule option.
   - (CodeOption, LibelleOption, Descriptif)
   - (NumEtudiant, NomEtudiant, PrenomEtudiant)
   - (NumEtudiant, CodeOption)

CodeOption	LibelleOption	Descriptif
1	Algo	abcde...
2	Math	abcd...
3	Prog	abc...
4	BDD	a...

NumEtudiant	NomEtudiant	PrenomEtudiant
1	Abbott	William
2	Costello	Lou
3	Laurel	Stanley
4	Hardy	Oliver
5	Lloyd	Harold
6	Marx	Groucho

NumEtudiant	CodeOption
1	1
2	2
3	3
4	1
5	1

- **Identifiez les problèmes de modélisation.**
  - *La table « NumEtudiant, CodeOption » est inutile car chaque étudiant choisie 1 option, donc il suffit d'ajouter une colonne CodeOption pour la table étudiant*
- **Donnez des exemples de données incohérentes, ou requêtes inefficaces, qui peuvent résulter de la modélisation.**

  Requêtes inefficaces *: Pour faire une jointure par exemple y a 3 tables qui interviennes, alors que si on ajouter une colonne CodeOption pour la table étudiant le nombre de tables pour cet requête est 2.*

  On risque aussi d'insérer plusieurs options pour un étudiant. Dans ce cas on satisfait pas les contraintes et on a quelque chose qui est pas consistant.
- **Proposez une nouvelle modélisation qui permet d'éviter ces problèmes.**

  – (CodeOption, LibelleOption, Descriptif)

  – (NumEtudiant, NomEtudiant, PrenomEtudiant**, CodeOption***)

3. Chaque étudiant peut s'inscrire à plusieurs options.
   - (CodeOption, LibelleOption, Descriptif)
   - (NumEtudiant, NomEtudiant, PrenomEtudiant)
   - (NumEtudiant, ListeCodeOption)

CodeOption	LibelleOption	Descriptif
1	Algo	abcde...
2	Math	abcd...
3	Prog	abc...
4	BDD	a...

NumEtudiant	NomEtudiant	PrenomEtudiant
1	Abbott	William
2	Costello	Lou
3	Laurel	Stanley
4	Hardy	Oliver
5	Lloyd	Harold
6	Marx	Groucho

NumEtudiant	ListeCodeOption
1	1, 2
2	2, 4
3	3
4	1, 2, 3, 4
5	1

- **Identifiez les problèmes de modélisation.**
  - ***Requêtes inefficaces:*** *car les options sont stocké dans une liste· il faut donc à chaque fois parcourir tous ces listes...*

- **Donnez des exemples de données incohérentes, ou requêtes inefficaces, qui peuvent résulter de la modélisation.**

  Requêtes inefficaces *: Si je cherche le liste des étudiants inscrit a l'option4 je doit parcourir la ListeCodeOption de chaque etudiant·*

- **Proposez une nouvelle modélisation qui permet d'éviter ces problèmes.**

  – (CodeOption, LibelleOption, Descriptif)
  – (NumEtudiant, NomEtudiant, PrenomEtudiant)
  – (NumEtudiant, codeOption) – **un certain étudiant pourra apparaitre dans plusieurs tuples.**

Normalement, c'est interdit moralement d'utiliser des listes dans une BDD car ca correspond a une table·

4. Les personnels d'une société implantée sur plusieurs sites sont affectés à un service d'une entreprise et à un site.
   - (codeSite, nomSite, adresseSite),
   - (codeService, nomService),
   - (codeSite, codeEmploye, NomEmploye)
   - (codeService, codeEmploye, NomEmploye, TelephoneEmploye)

codeSite	nomSite	adresseSite
1	Nouveau-site	Lyon
2	Ancien-site	Paris
3	Site-centrale	Marseille
4	Nouveau-site-centrale	Havre

codeService	nomService
1	technique
2	logistique
3	administratif
4	sécurité
5	autres

codeSite	codeEmploye	NomEmploye
1	3211	William
2	4211	Lou
3	5211	Stanley
4	698	Oliver
1	123	Harold
2	77	Groucho

codeService	codeEmploye	NomEmploye	TelephoneEmploye
1	3211	William	89657
2	4211	Lou	45678
3	5211	Stanley	123664
4	698	Oliver	456698
5	123	Harold	89956
1	77	Groucho	123657

- **Identifiez les problèmes de modélisation.**
  - ***Redondance :*** *NomEmployee sur 2 tables différentes*
  - ***Anomalies de mise à jour*** *: si le nom de l'employée est mise à jour, une incohérence est créée à moins que les 2 tables ou il y a le nom ne soit mis à jour de la même manière.*
  - ***On risque d'avoir un employé affecter a plusieurs sites et services car CodeEmployee peut apparaitre plusieurs fois dans la 3ème et 4ème table avec des codeSite ou codeService différents***
- **Donnez des exemples de données incohérentes, ou requêtes inefficaces, qui peuvent résulter de la modélisation.**

  Données incohérentes *: si le nom de l'employée est mise à jour suite à un mariage, divorce etc*
- **Proposez une nouvelle modélisation qui permet d'éviter ces problèmes.**

– (**codeSite**, nomSite, adresseSite),
– (**codeService**, nomService),
– (codeSite , codeService, **codeEmploye**, NomEmploye, TelephoneEmploye)

Décomposition d'un schéma de relation, diapos 48 - 49

Décomposition d'un schéma de relation

L'outil indispensable pour arriver à une forme normale

- Soit R(U) un schéma de relation
- Une décomposition de R(U) est un ensemble { $R_1(S_1), \ldots, R_k(S_k)$ } de schémas de relation tels que:

$$U = \bigcup_{i=1}^{k} Si$$

- Exemple

{Vendeur (V#,Vnom,Vville),
Produit (P#, Pnom, Pville),
Fourniture(V#, P#, Qte) }

est une décomposition de
R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

Propriétés d'une décomposition

- On ne peut pas décomposer arbitrairement
- Conditions pour une décomposition "raisonnable" :
  - Décomposition sans perte d'information
  - Décomposition sans perte de dépendances fonctionnelles

décomposition sans perte d'informations : la mise en forme normale de Boyce-Codd elle nous protège de ca mais pas de la perte de dépendances fonctionnelles

Une relation r de type **R** (ou une relation conforme au schéma **R**) est un ensemble de tuples de type **R**.

Notre problème: **comment décomposer la table pour éviter les problèmes**.

Mais d'abord qu'est-ce que c'est une décomposition d'une relation r ?

Si r une relation dans le schéma $R(A_1, \ldots, A_n)$ et $X_1, \ldots, X_k \subset \{A_1, \ldots, A_n\}$ des ensembles d'attributs tels que $X_1 \cup \ldots \cup X_k = \{A_1, \ldots, A_n\}$ alors la décomposition de r c'est l'ensemble de tables obtenues par projections:

$\pi_{X_1}(r), \ldots, \pi_{X_k}(r)$.

Décompositions sans perte d'information, diapos 48 - 49

Décomposition sans perte d'information

Idée : Si on remplace R (V#, Vnom, Vville, P#, Pnom, Pville, Qte) par {Vendeur, Produit, Fournitures}

notre BD, au lieu de stocker une instance J de R stockera ses projections

$\Pi_{V\#,Vnom,Vville}(J)$ $\Pi_{P\#,Pnom,Pville}(J)$ $\Pi_{V\#,P\#,Qte}(J)$

les dépendances sont pas expliquer la, mais chaque membre gauche d'une dépendance est une clé pour la relation pour laquelle elle s'applique

J

V#	Vnom	Vville	P#	Pnom	Pville	Qte
3	MagicV	Paris	5	jupe	Paris	5
3	MagicV	Paris	6	veste	Lille	2
2	IdealB	Lyon	12	manteau	Lyon	1
2	IdealB	Lyon	13	jupe	Paris	1

$\Pi_{V\#,Vnom,Vville}(J)$

V#	Vnom	Vville
3	MagicV	Paris
2	IdealB	Lyon

$\Pi_{P\#,Pnom,Pville}(J)$

P#	Pnom	Pville
5	jupe	Paris
6	veste	Lille
12	manteau	Lyon
13	jupe	Paris

$\Pi_{V\#,P\#,Qte}(J)$

V#	P#	Qte
3	5	5
3	6	2
2	12	1
2	13	1

Décomposition sans perte d'information

Idée

- La décomposition doit garantir que pour toute instance J de R, les projections de J contiennent la "même information" que J
- C'est à dire on doit pouvoir reconstruire une instance J de R à partir de ses projections
- Comment tenter de reconstruire l'instance à partir de ses projections?
  Jointure naturelle

$\Pi_{V\#,Vnom,Vville}(J) \bowtie \Pi_{P\#,Pnom,Pville}(J) \bowtie \Pi_{V\#,P\#,Qte}(J)$

On va utiliser la jointure naturelle, on prend plusieurs ensembles de relation et on va construire une relation sur l'union de tt les attributs

Le problème: est-ce que les tables obtenues par projections contiennent la même information que la table r ?

Mais « la même information » cela veut dire quoi? Problème de « faux » tuples.

Il y a un seul moyen d'essayer de reconstruire la relation r à partir de relations $\pi_{X_1}(r), \ldots, \pi_{X_k}(r)$, c'est faire **la jointure naturelle**.

Si $\pi_{X_1}(r) \bowtie \ldots \bowtie \pi_{X_k}(r)$ donne r alors nous avons réussi à décomposer r en tables plus petites tout en gardant les mêmes données que dans r.

Décomposition sans perte d'information

Rappel. Jointure naturelle de deux instances de relation:
I avec ensemble d'attributs X, et J avec ensemble d'attributs Y

$I \bowtie J$

retourne l'ensemble des tuples t sur attributs $X \cup Y$ telles que $t[X] \in I$ et $t[Y] \in J$

X= {A, B}

Y= {B,C}

I

A	B
1	2
4	2
6	6
7	7

J

B	C
2	3
2	5
9	1
8	8

$I \bowtie J$

A	B	C
1	2	3
1	2	5
4	2	3
4	2	5

Décomposition sans perte d'information (*lossless join*)

Définition.

Soir R(U) une schéma de relation et F un ensemble de DFs sur R.

Une décomposition $\{ R_1(S_1), \ldots, R_k(S_k) \}$ de R est

sans perte d'information par rapport à F

ssi, pour toute instance J de R qui satisfait F,

$$J = \pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie \ldots \bowtie \pi_{Sk}(J)$$

Décomposition sans perte d'information

Dans l'exemple, propriété souhaitée pour notre décomposition :

$$J = \Pi_{V\#, Vnom, Vville}(J) \bowtie \Pi_{P\#, Pnom, Pville}(J) \bowtie \Pi_{V\#, P\#, Qte}(J)$$

pour toute instance valide J de R

Est-ce vrai ?

Dans l'exemple de J donné, oui. Mais pour d'autres J valides?

Intuitivement, Oui : puisque en partant de la fourniture (V#, P#, Qte)

- V# nous permet de récupérer toutes les infos sur un unique vendeur (grâce à la DF V# → Vnom Vville)
- P# nous permet de récupérer toutes les infos sur un unique produit

  (grâce à la DF P# → Pnom Pville)

(une procédure plus rigoureuse pour ce test plus loin)

la propriété de décompositions sans perte d'information dépend des dépendances fonctionnelles

Un exemple de décomposition avec perte d'information

R(A, B, C) décomposition : { R1(A, B), R2(B, C) }

F= {AB → C}

Il existe une instance J de R qui satisfait F, mais qu'on ne peut pas reconstruire à partir de ses projections:

J

A	B	C
1	2	3
4	2	5

$\Pi_{AB}(J)$

A	B
1	2
4	2

$\Pi_{BC}(J)$

B	C
2	3
2	5

$\Pi_{AB}(J) \bowtie \Pi_{BC}(J)$

A	B	C
1	2	3
4	2	5
1	2	5
4	2	3

là on va avoir la perte d'info, et ces pertes d'infos ces ses tuples parasites· A partir du moment où je fais la jointure avec les 2 composantes... je vais aussi joindre 2 5 et du coup je me retrouve avec ce 2 5 parasite qui était pas là au début, et ça c'est pas correcte, ... je me retrouve aussi avec 4 3 qui était pas là au début

Décomposition sans perte d'information (*lossless join*)

Pour une instance J arbitraire, quelle est la connexion entre

J et $\pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$?

- Pour tout J, $J \subseteq \pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$

  Par la définition de jointure naturelle et projection :

  $t \in J \Rightarrow t[Si] \in \pi_{Si}(J)$ pour tout i $\Leftrightarrow$ $t \in \pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$

- le seule problème est donc que les jointures peuvent générer des tuples en plus (voir exemple précédent)
- Mais J n'est pas arbitraire : J satisfait des DFs, cela peut garantir l'inclusions inverse dans certains cas
- Tester si un décomposition est sans perte d'information : un algorithme simple existe

Source des certains commentaires : *Bases de données avancées - Normalisation / Wieslaw Zielonka*
https://www.irif.fr/~zielonka/Enseignement/BDAvances/2012/NORMALISATION/normalisation.pdf

Décompositions sans perte d'information

Définition. Soit r une relation de type $R = (A_1, \ldots, A_n)$. Soit X, Y deux ensembles non vides d'attributs tels que $X \cup Y = \{A1, \ldots, An\}$. On prenant les projections $r1 = \pi_X(r)$ et $r2 = \pi_Y(r)$ de r nous obtenons une décompositions de r en deux relations $r1$ et $r2$.

Cette décomposition est sans perte[1] si $r1 \bowtie r2 = r$, ou $\bowtie$ est la jointure naturelle de $r1$ et $r2$.

Si la décomposition est sans perte alors nous pouvons oublier la relation r et garder uniquement $\pi_X(r)$ et $\pi_Y(r)$ parce que r est reconstructible (par la jointure naturelle) à partir de $\pi_X(r)$ et $\pi_Y(r)$.

Notons que toujours $\pi_X(r) \bowtie \pi_Y(r)$ contient tous les tuples de r, c'est-`a-dire toujours r est inclus dans $\pi_X(r) \bowtie \pi_Y(r)$. Mais nous avons un problème si $\pi_X(r) \bowtie \pi_Y(r)$ contient des nouveaux tuples qui n'existaient pas dans la relation r. Dans ce cas la décomposition de r introduira de tuples parasites.

Theorem (Heath). *Soit r une relation de type $R(X)$ une relation qui satisfait toutes les dépendances de F.*
Soit $X = X_1 \cup X_2$. Alors la décomposition $\pi_{X_1}(r)$, $\pi_{X_2}(r)$ est sans perte si

- *soit $X_1 \cap X_2 \longrightarrow X_1$,*
- *soit $X_1 \cap X_2 \longrightarrow X_2$.*

En pratique on utilise le résultat suivant qui découle immédiatement du **théorème de Heath** :

Theorem (Théorème de décomposition). *Soit $R(Z)$ un schéma relationnel avec F comme l'ensemble de DF.*
Soit $X \longrightarrow Y$ une DF de F.
Alors la décomposition $R_1(XY) = \pi_{X \cup Y}(R)$, $R_2(Z \setminus Y) = \pi_{Z \setminus Y}(R)$ est sans perte.

Les applications successives de décompositions sans perte donne une décompositions sans perte.

Il ne faut jamais faire de décomposition si elle peut engendrer une perte d'information

Source : Bases de données avancées - Normalisation / Wieslaw Zielonka
https://www.irif.fr/~zielonka/Enseignement/BDAvances/2012/NORMALISATION/normalisation.pdf

[1] Dans certains livres on utilise le terme : **décomposition a jonction conservative**.

Exercice 4 : *Dépendances fonctionnelles, proposer une bonne décomposition*

On veut décrire les séances de travaux dirigés (TD) des unités de valeurs (UV) d'un département d'une université par la relation suivante :

FAC (NoTD, Salle, Horaire, Noenseignant, NomEnseignant, PrénomEnseignant, NoUV, NomUV, Noétudiant, NomEtudiant, PrénomEtudiant, AdresseEtudiant, DateInscription)

- *L'enseignement, dans ce département est divisé en unités de valeurs, chacune étant identifiée par un numéro ou par son nom.*
- *Un étudiant s'inscrit a une ou plusieurs UV (six au maximum), et pour chaque UV a un groupe de TD (NoTD).*
  - *Les inscriptions dans les différentes UV sont indépendantes les unes des autres.*
  - *On mémorise la date d'inscription de chaque étudiant à chaque UV (DateInscription).*
- *Il y a une séance de TD par semaine pour chaque UV.*
  - *Chaque TD a lieu dans une salle donnée et a un horaire donné.*
- *Les groupes de TD sont numérotés 1, 2, 3, … pour chaque UV.*
  - *Un enseignant assure un ou plusieurs groupes de TD d'une ou plusieurs UV.*
  - *Un groupe de TD d'une UV est assuré toute l'année par le même enseignant, plusieurs enseignants pouvant se partager les différents groupes de TD d'une même UV.*
- *On ne conserve que le prénom usuel de chaque personne.*

NoTD	Salle	Horaire	No enseignant	Nom Enseignant	Prénom Enseignant	NoUV	NomUV	No étudiant	Nom Etudiant	Prénom Etudiant	Adresse Etudiant	Date Inscription
1	567	Lundi 16h-17h	1	Abbott	William	1	Math	1	Hardy	Oliver	Paris	1.9
2	568	Mardi 16h-17h	2	Costello	Lou	1	Math	2	Lloyd	Harold	Pau	1.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	3	Marx	Groucho	Havre	3.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	1	Hardy	Oliver	Paris	4.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	2	Lloyd	Harold	Pau	6.9
2	568	Mardi 16h-17h	2	Costello	Lou	1	Math	3	Marx	Groucho	Havre	10.9

➢ Identifiez les dépendances fonctionnelles de FAC·

1. NoTD, NoUV -> Salle, Horaire, NoEnseignant
2. NoEnseignant -> NomEnseignant, PrenomEnseignant
3. NoUV -> NomUV
4. NomUV -> NoUV
5. NoEtudiant -> NomEtudiant, PrenomEtudiant, AdresseEtudiant
6. NoUV, NoEtudiant -> DateInscription, NoTD
7. Salle, Horaire -> NoUV, NoTD
8. NoEtudiant, Horaire -> Salle
9. NoEnseignant, Horaire -> Salle

Triviales !!

Car il est écrit : « unités de valeurs, chacune étant identifiée par un numéro ou par son nom »

On peut ajouter NoEnseignant *a droite ou a gauche*

➢ Proposez une bonne décomposition de FAC· Donnez les dépendances fonctionnelles qui proviennent de cette décomposition··

Il y a un algo pour le décomposition, mais comme on va le voir au prochain TP, la on le fait de façon informel :

NoTD	Salle	Horaire	No enseignant	NoUV
1	567	Lundi 16h-17h	1	1
2	568	Mardi 16h-17h	2	1
1	567	Jeudi 16h-17h	2	2

No étudiant	Nom Etudiant	Prénom Etudiant	Adresse Etudiant
1	Hardy	Oliver	Paris
2	Lloyd	Harold	Pau
3	Marx	Groucho	Havre

No enseignant	Nom Enseignant	Prénom Enseignant
1	Abbott	William
2	Costello	Lou

NoUV	NoTD	No étudiant	Date Inscription
1	1	1	1.9
1	2	2	1.9
2	1	3	3.9
1	1	1	4.9
1	2	2	6.9
1	2	3	10.9

TD (**NoTD**, Salle, Horaire, Noenseignant, **NoUV**)

Etudiant (**Noétudiant**, NomEtudiant, PrénomEtudiant, AdresseEtudiant)

Enseignant (**Noenseignant**, NomEnseignant, PrénomEnseignant)

Inscription (NoTD, **,NoUV, Noétudiant**, DateInscription)

UV (**NoUV**, NomUV)

NoUV	NomUV
1	Math
2	BDD

On a les même DF qu'avant ?

Si on a une DF que tous ses attributs se trouve dans une table, alors c'est sûr qu'on a pas perdu cette DF !

Pour les autres ? A vérifier !

Correction personnel (pas demandé pour ce TD)

SQL NATURAL JOIN

Dans le langage SQL, la commande NATURAL JOIN permet de faire une jointure naturelle entre 2 tables. Cette jointure s'effectue à la condition qu'il y ai des colonnes du même nom et de même type dans les 2 tables. Le résultat d'une jointure naturelle est la création d'un tableau avec autant de lignes qu'il y a de paires correspondant à l'association des colonnes de même nom.

Source : https://sql.sh/cours/jointures/natural-join

Jointure naturelle : R ⋈ S

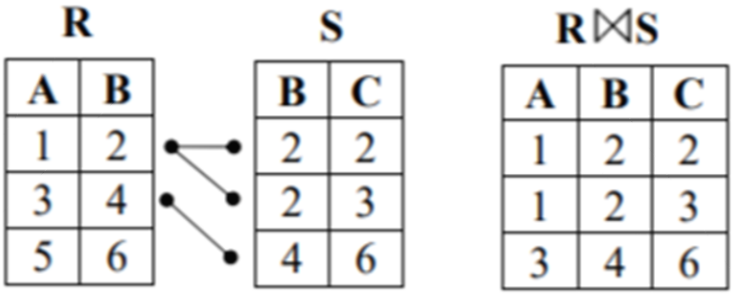
But : créer toutes les combinaisons entre tuples de deux relations ayant la même valeur pour tous les attributs en commun.

Précondition : les deux relations ont au moins un attribut en commun.

- schéma(R ⋈ S) = schéma(R) ∪ schéma(S)

⇒ Les attributs en commun n'apparaissent qu'une seule fois !

Exemple :



R

A	B
1	2
3	4
5	6

S

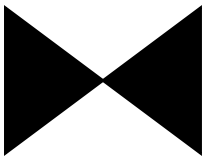
B	C
2	2
2	3
4	6

R ⋈ S

A	B	C
1	2	2
1	2	3
3	4	6

Source : Fichier mémo avec les définitions des opérateurs principaux de l'algèbre relationnelle

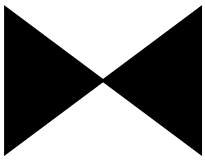
NoUV	NomUV
1	Math
2	BDD



NoTD	Salle	Horaire	No enseignant	NoUV
1	567	Lundi 16h-17h	1	1
2	568	Mardi 16h-17h	2	1
3	567	Jeudi 16h-17h	2	2

NoTD	Salle	Horaire	No enseignant	NoUV	NomUV
1	567	Lundi 16h-17h	1	1	Math
2	568	Mardi 16h-17h	2	1	Math
1	567	Jeudi 16h-17h	2	2	BDD

NoTD	Salle	Horaire	No enseignant	NoUV	NomUV
1	567	Lundi 16h-17h	1	1	Math
2	568	Mardi 16h-17h	2	1	Math
1	567	Jeudi 16h-17h	2	2	BDD



No enseignant	Nom Enseignant	Prénom Enseignant
1	Abbott	William
2	Costello	Lou

NoTD	Salle	Horaire	No enseignant	Nom Enseignant	Prénom Enseignant	NoUV	NomUV
1	567	Lundi 16h-17h	1	Abbott	William	1	Math
2	568	Mardi 16h-17h	2	Costello	Lou	1	Math
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD

NoTD	Salle	Horaire	No enseignant	Nom Enseignant	Prénom Enseignant	NoUV	NomUV
1	567	Lundi 16h-17h	1	Abbott	William	1	Math
2	568	Mardi 16h-17h	2	Costello	Lou	1	Math
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD

NoUV	NoTD	No étudiant	Date Inscription
1	1	1	1.9
1	2	2	1.9
2	1	3	3.9
2	1	1	4.9
1	2	2	6.9
1	2	3	10.9

NoTD	Salle	Horaire	No enseignant	Nom Enseignant	Prénom Enseignant	NoUV	NomUV	No étudiant	Date Inscription
1	567	Lundi 16h-17h	1	Abbott	William	1	Math	1	1.9
2	568	Mardi 16h-17h	2	Costello	Lou	1	Math	2	1.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	3	3.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	1	4.9
2	568	Mardi 16h-17h	2	Costello	Lou	1	BDD	2	6.9
2	568	Mardi 16h-17h	2	Costello	Lou	1	Math	3	10.9

NoTD	Salle	Horaire	No enseignant	Nom Enseignant	Prénom Enseignant	NoUV	NomUV	No étudiant	Date Inscription
1	567	Lundi 16h-17h	1	Abbott	William	1	Math	1	1.9
2	568	Mardi 16h-17h	2	Costello	Lou	1	Math	2	1.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	3	3.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	1	4.9
2	568	Mardi 16h-17h	2	Costello	Lou	1	BDD	2	6.9
2	568	Mardi 16h-17h	2	Costello	Lou	1	Math	3	10.9

No étudiant	Nom Etudiant	Prénom Etudiant	Adresse Etudiant
1	Hardy	Oliver	Paris
2	Lloyd	Harold	Pau
3	Marx	Groucho	Havre

NoTD	Salle	Horaire	No enseignant	Nom Enseignant	Prénom Enseignant	NoUV	NomUV	No étudiant	Nom Etudiant	Prénom Etudiant	Adresse Etudiant	Date Inscription
1	567	Lundi 16h-17h	1	Abbott	William	1	Math	1	Hardy	Oliver	Paris	1.9
2	568	Mardi 16h-17h	2	Costello	Lou	1	Math	2	Lloyd	Harold	Pau	1.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	3	Marx	Groucho	Havre	3.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	1	Hardy	Oliver	Paris	4.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	2	Lloyd	Harold	Pau	6.9
2	568	Mardi 16h-17h	2	Costello	Lou	1	Math	3	Marx	Groucho	Havre	10.9

même table après la jointure !

La table au début :

NoTD	Salle	Horaire	No enseignant	Nom Enseignant	Prénom Enseignant	NoUV	NomUV	No étudiant	Nom Etudiant	Prénom Etudiant	Adresse Etudiant	Date Inscription
1	567	Lundi 16h-17h	1	Abbott	William	1	Math	1	Hardy	Oliver	Paris	1.9
2	568	Mardi 16h-17h	2	Costello	Lou	1	Math	2	Lloyd	Harold	Pau	1.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	3	Marx	Groucho	Havre	3.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	1	Hardy	Oliver	Paris	4.9
1	567	Jeudi 16h-17h	2	Costello	Lou	2	BDD	2	Lloyd	Harold	Pau	6.9
2	568	Mardi 16h-17h	2	Costello	Lou	1	Math	3	Marx	Groucho	Havre	10.9

Exercice 5 : *Dépendances fonctionnelles*

Correction personnel (pas fait en TD)

On considère une société de transport par bus.

- Le réseau de bus est composé d'un certain nombre de **lignes** identifiées par leur numéro (numLigne),
  - chaque ligne comporte un certain nombre d'**arrêts** dont on connait le nom (arrêt) et le numéro par rapport au début de cette ligne (noArret).
  - On suppose que le trajet d'une ligne de bus ne contient qu'une seule fois un même arrêt (pas de boucle).
- **Les bus** sont également numérotés (numBus).
- Par ailleurs, la société emploie un certain nombre de **conducteurs**, chacun étant identifié par son matricule (matConducteur),
  - on connaît également le nom (nomConducteur) et le prénom (prénomConducteur).

Questions

Donnez les dépendances fonctionnelles de la relation

R(numLigne, arrêt, noArret, numBus, matConducteur, nomConducteur, prénomConducteur)

dans les deux cas suivants :

Cas A :

- Un bus donné ne roule que sur une ligne.
- Chaque conducteur ne travaille également que sur une ligne et conduit toujours le même bus.
- Un même arrêt peut être desservi par plusieurs lignes de bus.

numLigne,noArret -> arret	matConducteur -> nomConducteur, prenomConducteur
numBus -> numLigne	matConducteur -> numLigne, numBus

Cas B :

- arrêt (arrêt) n'appartient qu'à une seule ligne.
- Un bus peut desservir des lignes différentes,
- par contre il est toujours conduit par le même conducteur.
- Par ailleurs un conducteur peut conduire sur des lignes différentes.

numLigne,noArret -> arret	matConducteur -> nomConducteur, prenomConducteur
arret -> noArret, numLigne	numBus -> matConducteur