Systèmes de Gestion de Bases de Données
L3 informatique, MIage, ASR

S. Cerrito

premier sémestre 2007-2008

1 Introduction

SGBD= **S**ystème de **G**estion d'une **B**ase de **D**onnées

Quelles sont les spécificités d'un SGBD ?

- Très grande quantité de données à gérer, qui doivent être stockées dans plusieurs fichiers, voir plusieurs sites.
- Besoin d'interroger et/ou mettre à jour souvent, rapidement et facilement ces données.
- Besoin d'accès concurrents.
- Besoin de sécurité.
- Besoin important de gérer des pannes éventuelles.

INTRODUCTION, suite

Important : indépendance du niveau "logique" (vision "conceptuelle" des données) par rapport au niveau physique (implémentation), car :

1. Utilisateur d'une BD (base de données) : pas forcément un pro de l'implémentation. Il doit juste comprendre comment les données sont "logiquement" organisées.
2. L'implémentation peut changer, sans que le "schéma" (la "forme conceptuelle") de la BD change.
3. Modèle logique clair $\Rightarrow$
   (a) possibilité d'un *langage de requêtes* facile pour l'utilisateur
   (b) si l'implémentation change, pas besoin d'écrire un nouveau programme pour poser la même question à la base !
4. Idem pour le *langage de mise à jour*.

INTRODUCTION, suite

Historique

- Avant 1970 : BD=fichiers d'enregistrements, "modèles" *réseaux* et *hiérarchique*; pas de vraie indépendance logique/physique.
- En 1970 : modèle *relationnel* (Codd) : vraie indépendance logique/physique.
- Années 80 et 90 : nouveaux modèles :
  modèle à objets
  modèle à base de règles (Datalog)
- Fin années 90 : données dites *semi-structurées* (XML).

Ce cours : modèle relationnel, le plus utilisé dans la pratique.

Pré-réquis pour ce cours :

- Notions de base du modèle relationnel
- Algébre relationnelle
- SQL
- Méthode de conception de schéma EA.

Notions vues dans le cours de Mise à Niveau. Support du cours de mise à niveau (et supports pour ce cours) :

`www.lami.univ-evry.fr/~serena`

Plan du Cours

- Partie I :
  1. + sur les fondements des langage de requête
  2. Raffinement d'un schéma de base.
- Partie II :
  Aspects "système" : stockage physique des données, etc. .

2 Notions essentielles des BD relationnelles

Mots clés :

- Univers U, Attributs $A_1, \cdots, A_n$
- Domaine $Dom(A)$ d'un attribut A
- Schéma d'une relation dont le nom est R.
- n-uplet sur un ensemble E d'attributs
- Relation (ou "table") sur un schéma de relation
- Schéma d'une BD
- Base de données B sur un schéma de base

Un *univers* U est un ensemble fini et non-vide de noms, dits *attributs*.
Le *domaine* d'un attribut A ($Dom(A)$) est l'ensemble des valeurs possibles associ
A.

Exemple :
$U = \{NomFilm, Realisateur, Acteur, Producteur, NomCinema, Horaire\}$

$Dom(NomFilm) = Dom(Realisateur) = Dom(Acteur) = Dom(Producteur) =$
$Dom(NomCinema)=$ chanes de caractères.

$Dom(Horaire) = \{h.m \mid h \in [1, \cdots, 24],\ m \in [0, \cdots, 60]\}$

Un *schéma* d'une relation dont le nom est R est un sous-ensemble non-vide de l'univers U.

Suite de l'exemple :

- Schéma de la relation $Film$=$\{NomFilm, Realisateur, Acteur, Producteur\}$
- Schéma de la relation $Projection$=$\{NomFilm, NomCinema, Horaire\}$

Intuition : Format de deux tables.

$Film$:

NomFilm	Realisateur	Acteur	Producteur
⋮	⋮	⋮	⋮

$Projection$:

NomFilm	NomCinema	Horaire
⋮	⋮	⋮

Soit $E = \{A_1, \cdots, A_n\}$ le schéma d'une relation. Un *n-uplet* n sur E est une fonction $E \to Dom(A_1) \cup \cdots \cup Dom(A_n)$ telle que, pour tout $A_i \in E$, $n(A_i) \in Dom(A_i)$.

Si $E' \subset E$, la restriction de n à E se note $n(E')$.

Exemple.

Un n-uplet possible sur le schéma de $Projection$:
$\langle$“$Jugez - moi\ coupable''$, “$Gaumont\ Alesia''$, 13.35$\rangle$.

Sa restriction à $\{NomCinema, NomFilm\}$:
$\langle$ “Jugez-moi coupable”, “Gaumont Alesia”$\rangle$.

Pourquoi définir un n-uplet comme une *fonction* plutôt que comme un élément de $Dom(A_1) \times \cdots \times Dom(A_n)$?

Une *relation* (table) r sur un schéma de relation S est un ensemble d'n-uplets sur S. On dit aussi : S est le schéma de r.

Exemple.

$Film$:

NomFilm	Réalisateur	Acteur	Producteur
nf1	r1	a1	p1
nf1	r1	a2	p1
nf2	r2	a1	p2
nf3	r2	a1	p2

$Projection$:

NomFilm	NomCinema	Horaire
nf1	nc1	h1
nf1	nc2	h2
nf2	nc1	h3
nf3	nc2	h1

Un schéma $\mathcal{S}$ *d'une base* sur un univers U est un ensemble non-vide d'expressions de la forme $N(S)$ où S est un schéma de relation et N un nom de relation.

Exemple(on omet les {} dans les schémas des relations).

$U =$
$\{NomFilm, Realizateur, Acteur, Producteur, NomCinema, Horaire, Spectateur\}$

$$\mathcal{S} =$$

$\{$
$Film(NomFilm, Realizateur, Acteur, Producteur),$
$Projection(NomFilm, NomCinema, Horaire), Aime(Spectateur, NomFilm)$
$\}$

Schéma de la base = Format des données de la base.

Quel est le format de la base de l'exemple ?

- Une *base de données* B sur un schéma de base $\mathcal{S}$ (avec univers U) est un ensemble de relations finies $r_1, \cdots r_n$ o chaque r_i est associée à un nom de relation N_i et est telle que si $N_i(S) \in \mathcal{S}$, alors r_i a S comme schéma.
- On peut aussi imposer des *contraintes* sur les données. Par exemple : les *dépendances fonctionnelles* (à voir), qui fixent, entre autres, les *clés* des relations (à voir).
- Ces contraintes, dites d'*intégrité*, font aussi partie de la spécification du format des données de la base.

Exemple d'une base.

Film			
NomFilm	Réalisateur	Acteur	Producteur
nf1	r1	a1	p1
nf1	r1	a2	p1
nf2	r2	a1	p2
nf3	r2	a1	p2

Projection		
NomFilm	NomCinema	Horaire
nf1	nc1	h1
nf1	nc2	h2
nf2	nc1	h3
nf3	nc2	h1

Aime	
NomFilm	Spectateur
nf1	s1
nf1	s2
nf2	s1
nf3	s3

3 Fondements des Langages de Requête

- Informellement : *Requête sur une base* = question que l'on pose à la base.
- *Langage de requête*= langage permettant d'écrire des requêtes
- Importance d'un langage de requête formel et rigoureux :
  1. Conception de langages commerciaux
  2. Evaluation de la puissance d'expression de chaque langage commercial
  3. Possibilité de déterminer ce qu'un langage commercial ne pourra pas exprimer
  4. Notion d'équivalence entre deux expressions de requête ⇒ Optimisation "logique" de l'évaluation d'une requête

- Deux langages formels de requête :
  algèbre relationnelle (**vue dans la Mise à Niveau**)
  calcul relationnel à variables n-uplets (nouveau).
  Equivalents, mais :
  - Algèbre : "procédurale" (analogie : langage de programmation C)
  - Calcul Relationnel : "déclaratif" (analogie : langages de programmation Caml et Prolog)
- SQL (le langage commercial de requête le + utilisé, **vu dans la Mise à Niveau**) :
  des notions viennent de l'algèbre, d'autres du calcul relationnel.

3.1 Les opérateurs de l'algèbre relationnelle : mini rappel

- Opérateurs ensemblistes : union ($\cup$), intersection ($\cap$), différence ($\setminus$), produit cartésien ($\times$)
- projection sur un ensemble d'attributs E (π_E), sélection d'un ensemble de n-uplets selon une condition C (σ_C), jointure "naturelle" ($\bowtie$), division ($\div$), renommage (ρ).

3.2 Calcul Relationnel à Variables n-uplets

- Langage de la logique des prédicats avec plusieurs *sortes* ou "types".
- Avec $\forall t$ et $\exists t$ on quantifie sur les n-uplets de la base, pas sur les valeurs d'un attribut.
- Par ex., une valeur possible de la variable t est le triplet $\langle nf1, nc1, h1\rangle$ de la relation $Projection$, une autre est $\langle nf1, nc2, h2\rangle$, une autre encore est $\langle nf3, nc2, h2\rangle$.
- Notation : si t est une variable et A un attribut, l'expression $t.A$ indique la valeur pour A de la valeur de t. Exemple : $t.NomFilm$, où t varie sur les 3-plets de la table $Projection$.
- Variante où l'on quantifie sur les valeurs des attributs : $h1, h2$ etc. : Calcul Relationnel à Variables Domaine".

Pour un schéma de base $\mathcal{S}$ donné sur un univers U, l'*alphabet* des formules du calcul relationnel est :

- $V=$ ensemble infini de variables. Pour chaque $t \in V$, un *type*, noté $type(t)$, où $type(t) \subseteq U$.
- $C =$ ensemble de constantes $= \bigcup_{A_i \in U} Dom(A_i)$
- Symboles de prédicat : $=$ et tout nom de relation dans $\mathcal{S}$.

Formules Atomiques (F.A.) :

1. **Si** $t, t' \in V$, A attribut $\in type(t)$, B attribut $\in type(t')$, **alors** le mot $t.A = t'.B$ est une F.A.
2. **Si** $t \in V$, A attribut $\in type(t)$, $k \in C$, $k \in Dom(A)$, **alors** le mot $t.A = k$ est une F. A.
3. **Si** $t \in V$, R est un nom de relation, S est le schéma de relation associé à R et $type(t) = S$, **alors** $R(t)$ est une F.A.

Exemples de F.A..

Soit t et t' de type $\{NomFilm,\ NomCinema,\ Horaire\}$.

1. $t.NomCinema = t'.NomCinema$
2. $t.NomCinema =$ “Gaumont Alésia”
3. $Projection(t)$

Signification (sémantique) des F.A.

Soit B une base sur le schéma $\mathcal{S}$. Soit n_1 un n-uplet choisi comme valeur de la variable t et soit n_2 un n-uplet choisi comme valeur de la variable t'

1. $t.A = t'.B$ est vraie par rapport à B ssi la valeur associée à l'attribut A par n_1 est la même que celle associée à l'attribut B par n_2.
2. $t.A = k$ est vraie par rapport à B ssi la valeur associée à l'attribut A par n_1 est k.
3. $R(t)$ est vraie par rapport à B ssi la relation de nom R contient n_1.

N.B. Si $type(t) = E$, où E est un ensemble d'attributs, tout n-uplet sur E peut être une valeur de t, même si n n'appartient à aucune relation !

Exemple. Base B :

$AimeLivre$

Personne	NomLivre
p1	l1
p2	l2
p1	l2

$Livre$

NomLivre
l1
l2

Associons $\langle p1, l1\rangle$ à t et $\langle l1\rangle$ à t'.

$t.NomLivre = t'.NomLivre$ est vraie par rapport à B.

$t.NomLivre = l2$ est fausse.

$AimeLivre(t)$ est vraie.

Associons $\langle p2, l1\rangle$ à t et $\langle l2\rangle$ à t'.

$t.NomLivre = t'.NomLivre$ est fausse.

$t.NomLivre = l2$ est fausse.

$AimeLivre(t)$ est fausse.

Formules

$E \subset U$=ensemble d'attributs

- Toute F.A. est une formule.
- Si F est une formule alors $(\neg F)$ l'est aussi.
- Si F_1 et F_2 sont des formules et $* \in \{\wedge, \vee, \rightarrow\}$ alors $(F_1 * F_2)$ est une formule.

Formules : suite

- Si F est une formule et $t \in V$ alors :
  $(\forall t : E\ F)$ et $(\exists t : E\ F)$
  sont des formules.
  F= *portée* de $\forall t$ (resp. $\exists t$).

Lecture intuitive de "$\forall t : E$" : quelque soit t de type E. Lecture intuitive de "$\exists t : E$" : il existe au moins un t de type E.

Conventions : droit de ne pas écrire les () le plus à l'extérieur. Autres droits : $F_1 \wedge F_2 \wedge F_3$ à la place de $(F_1 \wedge F_2) \wedge F_3$ ou $F_1 \wedge (F_2 \wedge F_3)$; idem pour $\vee$.

Lien avec le cours de logique du L2 : syntaxe du calcul des prédicats.

Signification (Sémantique)des formules

F : formule, B : base. ch : choix de valeurs pour les variables libres de F.
t : variable n-uplet, $type(t) = E$.
Par rapport à B et ch on a :

- Si F est une F.A. , F est vraie ou fausse selon les critères déjà vus.
- Si F est $(\neg F_1)$, F est vraie ssi F_1 est fausse.
- Si F est $(F_1 \wedge F_2)$, F est vraie ssi F_1 est vraie et F_2 aussi.
- Si F est $(F_1 \vee F_2)$, F est vraie au moins une de F_1, F_2 est vraie.
- Si F est $(F_1 \rightarrow F_2)$, F est vraie ssi ou F_1 est fausse ou F_2 est vraie.
- Si F est $\forall t : E\ F_1$, F est vraie ssi, quelle que soit la valeur de t, F_1 est vraie.
- Si F est $\exists t : E\ F_1$, F est vraie ssi il existe au moins une valeur de t pour laquelle F_1 est vraie.

Lien avec le cours de logique du L2 : ici B est l'interprétation !

Révenons à la syntaxe et au typage.

Erreurs d'écriture possibles :

Non-Formules :

1. $$(Aime(t) \wedge Livre(t')$$

   Erreur de syntaxe, pas de typage.

2. $Aime(t) \wedge t.age = 40$ où $type(t) = \{NomPersonne, NomLivre\}$
   Erreur de typage!

3. $$\forall t : Personne \neg Livre(t)$$

   Erreur de typage!

Variables Libres d'une Formule F : celles qui ne sont pas dans la portée d'un quantificateur ($\forall$ ou $\exists$).

Exemple. F :

$$[\exists t : Personne, NomLivre\ (AimeLivre(t) \wedge t'.Personne ='' Jean'')] \\ \wedge\, t''.NomLivre = t.NomLivre$$

$$Varlib(F) = \{t', t''\}$$

Formulation d'une requête dans le calcul relationnel à variables n-uplets.

- Format :

$$\{t : E \mid F(t)\}$$

  où $F(t)$ est une formule ayant t comme seule variable libre et E est $type(t)$.

- Réponse : l'ensemble des des valeurs de t pour lesquelles $F(t)$ est vraie.

- Exemple : "Quels sont les (titres des) films projetés au Gaumont Alésia ?
  $\{t : NomFilm \mid \exists t' : NomFilm, NomCinema, Horaire\ (Projection(t')$
  $\wedge t'.NomCinema = \text{“}GaumontAlesia'' \wedge t.NomFilm = t'.NomFilm)\}$

Un autre exemple.

“Quels sont les titres et les acteurs des films projetés au Gaumont Alésia ?

$\{t : NomFilm, Acteur \mid$
$\exists t'' : NomFilm, Acteur, Realisateur, Producteur$
$\exists t' : NomFilm, NomCinema, Horaire$
$(Film(t'') \wedge (Projection(t')$
$\wedge\, t'.NomCinema = \text{“}GaumontAlesia'' \wedge t''.NomFilm = t'.NomFilm$
$t.NomFilm = t'.NomFilm \wedge t.Acteur = t''.Acteur)\}$

Question. Même base B, mais domaines des attributs $\neq$: réponse unique à une requête dans le calcul ?

NON.

Exemple : Schéma $=\{R(A)\}$.

Base B :

A
1
2
3

Trois cas : 1) $Dom(A) = \{1, 2, 3\}$, 2) $Dom(A) = \{1, 2, 3, 4\}$ 3) $Dom(A) = \mathbb{N}$

Requête : $\{t : A \mid \neg R(t)\}$

Réponses respectives dans les trois cas : $\emptyset$, $\{4\}$, $\{n \mid n \in Nat \; et \; n > 3\}$

Dans le troisième cas, la réponse n'est même pas une relation finie ! Pas de correspondance avec l'algèbre relationnelle.

Requête *dépendante du domaine*.

- $\exists$ restriction syntaxique sur la forme d'une requête R du calcul qui assure que R est indépendante du domaine et équivalente à une expression algébrique : notion de requête *saine*.
- Définition de *saine* : techniquement compliquée, pas donnée ici.
- Mais : exemples de réécriture d'une requête dépendante du domaine (D.D.) de façon qu'elle soit indépendante du domaine (I.D.).

Exemples

Schéma de la base : {
$Film(NomFilm, Realizateur, Acteur, Producteur)$,
$Projection(NomFilm, NomCinema, Horaire), Aime(Spectateur, NomFilm)$
}

"Quels films ne sont aimés par personne ?"

D.D. : $\{t.NomFilm \mid \neg Aime(t)\}$

I.D. : $\{t.NomFilm \mid \exists t' : NomFilm, Realizateur, Acteur, Producteur$
$(Film(t') \wedge t'.NomFilm = t.NomFilm \wedge \neg Aime(t))\}$

Exemples, suite

"Quels films ne sont pas projetés au Gaumont-Alésia ?"

D.D. : $\{t.NomFilm \mid \neg\exists t'' : NomFilm, NomCinema, Horaire$
$(Projection(t'') \wedge t''.NomCinema = \text{“}Gau.Al.\text{”} \wedge t''.NomFilm = t.NomFilm)\}$

I.D. : $\{t.NomFilm \mid \exists t' : NomFilm, Realizateur, Acteur, Producteur$
$[Film(t') \wedge$
$\neg\exists t'' : NomFilm, NomCinema, Horaire\ (Projection(t'') \wedge t''.NomCinema =$
$\text{“}Gau.Al.\text{”} \wedge t''.NomFilm = t'.NomFilm \wedge t'.NomFilm = t.NomFilm)]\}$

- algèbre relationnelle et calcul relationnel à variables n-uplets (requêtes I.D.) : expression des mêmes requêtes.
- Un langage commercial est dit *relationnellement complet* ssi il peut exprimer toutes les requêtes que l'algèbre peut exprimer.
- SQL est relationnellement complet.
- SQL utilise des opérateurs de l'algèbre, mais aussi des variables n-uplets.

4 Concevoir et raffiner des schémas de bases

4.1 Les étapes de la conception d'un schéma relationnel

1. Analyse des besoins (informelle)
2. Utilisation d'un outil "graphique" pour une première modélisation : méthode *Entité-Association* (**vue dans la M. à N.**).
3. Passage de la représentation graphique au schéma relationnel (avec ses *dépendances fonctionnelles* (D.F, une classe de contraintes sur les donnes)) : presque mécanique.
4. Analyse du schéma $\mathcal{S}$ obtenu. Satisfaisant ? Si oui, terminé. SINON, *décomposition* de $\mathcal{S}$ selon certains critères (formels), qui utilisent les DF.

4.2 Dépendances fonctionnelles

- Notion fondamentale pour la problèmatique de la conception de schéma.
- **Vue en partie dans la M. à Niveau**. Court rappel des notions de la M. à N.
- Puis, + sur les D.F.

4.2.1 Notions de base sur les Dépendances Fonctionnelles

- Dépendances fonctionnelles : un type de contraintes d'intégrité. Participent à la spécification du format des données.
- X et Y ensembles d'attributs inclus dans le schéma d'une table R. Ecriture d'une dép.fonct. :

$$X \to Y$$

- Lecture : X détermine (ou donne) Y.
  Signification : La table r dont le nom est R satisfait $X \to Y$ ssi :
  qques soient les n-uplets t, t' de r, si $t(X) = t'(X)$ alors $t(Y) = t'(Y)$.
  (Notation : $r \models X \to Y$).
- Convention : si $X = \{A_1, \cdots, A_n\}$ et $Y = \{B_1, \cdots, B_m\}$ on écrit :
  $A_1 \cdots A_n \to B_1 \cdots B_m$ (pas de $\{,\}$, pas de virgule).

Exemple. Une table possible pour Films :

Titre	Date	Long	Couleur	NomStudio	IdProd
King Kong	1933	100	False	RKO-Pathé	601
King Kong	2005	187	True	Universal	798
Million Dollar Baby	2004	142	True	Warner	900

$Titre \rightarrow Couleur$ n'est pas satisfaite.

$Titre\ Date \rightarrow Couleur$ est satisfaite.

$Titre\ Date \rightarrow Couleur\ NomStudio$ est satisfaite.

$Titre\ Date \rightarrow Titre\ Date\ Long\ NomStudio\ Couleur\ IdProd$ est satisfaite.

- Associer un ensemble de dép. fonct. DF à un schéma de base $\mathcal{S}$ sert aussi à fixer en avance les *clés* des tables.
- Clé d'une table r ?? Intuition : identifiant de chaque n-uplet de r.
- Formellement : si S est un schéma d'une relation, r une relation sur S et $X \subseteq S$ alors :
  X est une clé de S ssi
  1. $r \models X \rightarrow S$
  2. X est minimal par rapport à cette propriété : $\nexists\, Z$ t.q. $Z \subset X$ et $r \models Z \rightarrow S$
- **Exemple précédent** : *Titre Date* est une clé, *Titre* seul non.
  Titre Date Couleur n'est pas une clé mais c'est une *super clé*.

Problème : Comment fixer en avance, avec le schéma, les clés aussi ?

- Exemple. Schéma(R)=$\{A, B, C, D\}$, $DF = \{AB \rightarrow CC \rightarrow D\}$. La dép.. fonct. $AB \rightarrow A\ B\ C\ D$ est impliquée par DF et AB sera une clé de toute table associée à R satisfaisant toutes les DF.

- "Impliquée" = ??
  Déf. : Soit F un ensemble de dépendances fonctionnelles et f une dép. fonct. On dit que F *implique* f (f est impliquée par F) ssi toute relation r satisfaisant toutes les dép. fonct. de F satisfait aussi f.

- Exemples :
  - $F_1 = \{A \rightarrow B,\ B \rightarrow CD\}$ implique $A \rightarrow C, D$ et implique aussi $A \rightarrow ABCD$.
  - $F_2 = \{A \rightarrow B,\ A \rightarrow C\}$ implique $A \rightarrow BC$.
  - $F_3 = \{A \rightarrow BC\}$ implique $A \rightarrow B$ et $A \rightarrow C$.
  - $F_4 = \{A\ B \rightarrow C,\ A\ D \rightarrow E\}$ **n'implique pas** $A\ B \rightarrow C\ E$: $\exists$ Contre-exemple !

4.2.2 Plus sur les dépendances fonctionnelles

Comment calculer de façon mécanique si f est impliquée par DF ?

- **Déf.** : Soit U l'ensemble des attributs dans DF, $X \subseteq U$.
  La *fermeture* de X par rapport à DF (${X^+}_{DF}$) est :

$$max\{Y \subseteq U \mid DF \; implique \; X \rightarrow Y\}$$

- **Théorème** : DF implique $X \rightarrow Y$ ssi $Y \subseteq {X^+}_{DF}$.
- Donc pour savoir si DF implique $X \rightarrow Z$ il suffit de savoir calculer ${X^+}_{DF}$.

ALGO DE CALCUL DE ${X^+}_{DF}$:

Entrée : DF, X, U, où U est l'univers (tous les att. dans DF)

Sortie : ${X^+}_F$ comme valeur de la variable res.

res:=X;

répéter :

pour chaque $Y \to Z \in DF$, **si** $Y \subseteq res$ **alors** $res := res \cup Z$

jusqu'à quand res n'a pas changé ou $res = U$;

renvoyer res

Question 1 : Complexité dans le pire des cas ?

Exemple de calcul de ${AB^+}_{DF}$ pour $DF = \{A \to C, BC \to D, ED \to E\} \rightsquigarrow$ tableau.

Question 2 : S=schéma d'une relation, $X \subseteq S$. Comment utiliser cet algo pour décider si X est une clé de toute relation sur S (fixée par DF) ?

4.3 Problèmes possibles pour un schéma d'une base et Décompositions du schéma

Relation FOURNISSEUR

Nom	Adr	$Item$	$Prix$
n1	a1	i1	p1
n1	a1	i2	p2
n2	a2	i2	p3
n2	a2	i3	p4

$DF = Nom \rightarrow Adr,\ Nom\ Item \rightarrow Prix$

- Redondances, car l'adresse est répétée.
- Anomalies de mise à jour : si on met à jour une adresse... attention, partout !
- Anomalies d'insertion : si pas de valeur nuls, impossibilité d'enregistrer une adresse d'une producteur qui n'a pas de produit.
- Anomalies de suppression : si effacement de tous les articles de n1, perte de l'adresse de n1.

Une solution possible : décomposer le schéma de relation

$$\text{FOURNISSEUR}(Nom,\ Adr,\ Item,\ Prix)$$

en :

$$\text{NA}(Nom,\ Adr) \text{ et } \text{NIP}(Nom,\ Item,\ Prix)$$

Contenu de la table NA : $\pi_{Nom,Adr}(\text{FOURNISSEUR})$.
Contenu de la table NIP : $\pi_{Nom,Item,Prix}(\text{FOURNISSEUR})$.

(Exemple de la façon générale de décomposer une table de schéma S en n tables de schémas $S_1, \cdots, S_n$ où $(S_1 \cup \cdots \cup S_n) = S$).

- **Toute décomposition possible est OK** ?
- Un des critères qu'une bonne décomposition doit respecter : il faut pouvoir reconstruire, par jointure, la relation de départ.
- Dans notre exemple :
  $\pi_{Nom,Adr}(FOURNISSEUR) \bowtie \pi_{Nom,Item,Prix}(FOURNISSEUR) = FOURNISSEUR$? OUI.
- En général ? NON. Contre-ex : Table R :

A	B	C
a1	b1	c1
a2	b1	c2

  Décomposition en R1(A,B) et R2(B,C) (contenus=projections de R).
  $\langle a1, b1, c2\rangle \in (\pi_{A,B}(R) \bowtie \pi_{B,C}(R))$ mais $\langle a1, b1, c2\rangle \notin$ R !
  La décomposition fait perdre l'information négative :
  "$\langle a1, b1, c2\rangle$ **n'est pas** une ligne de R".

- Soit F un ensemble de DF. Une décomposition d'un schéma S d'une table R en $S_1, \cdots, S_n$ (o $(S_1 \cup \cdots \cup S_n) = S$) est dite sans perte d'information (SPI) par rapport à F ssi, qque soit le contenu de la table de nom R :

$$(\pi_{S_1}(R) \bowtie \cdots \bowtie \pi_{S_n}(R)) = R$$

- *N.B.* :

$$R \subseteq (\pi_{S_1}(R) \bowtie \cdots \bowtie \pi_{S_n}(R))$$

  est toujours vrai, la $\subseteq$ réciproque non (contre-ex. précédent).

- Rôle de F ?? Remarque : ce contre-ex. impossible si $B \rightarrow C$ ou $B \rightarrow A$.

- **Thm.** Soit : $df1 = B \rightarrow C$, $df2 = B \rightarrow A$, F un ensemble de DF sur $S = \{A, B, C\}$.
  Si F implique $df1$ ou $df2$, alors la déc. de S en $\{A, B\}$ et $\{B, C\}$ est SPI par rapport à F.

- + général : Une déc. de S en 2 sous-schémas S_1 et S_2 est SPI ssi F implique $(S_1 \cap S_2) \rightarrow (S_1 \setminus S_2)$ ou $(S_1 \cap S_2) \rightarrow (S_2 \setminus S_1)$.

- Et si $n > 2$? Algo qui teste si une déc. est SPI : chase. **A voir en TD**.

- En résumé :
  Critère 1 qu'une décomposition doit respecter : être SPI par rapport aux DF.

- SPI : seul critère à souhaiter pour une déc. d ?
- **Exemple**. V : ville, R : rue, C : Code Postal. Schéma : INFOS(R,C,V).
  DF=$\{VR \to C,\ C \to V\}$.
  Déc. d en INFO1(RC) et INFO2(CV) :

	INFOS	
R	C	V
r1	c1	v1
r1	c2	v2
r3	c3	v3

INFO1	
R	C
r1	c1
r1	c2
r3	c3

INFO2	
C	V
c1	v1
c2	v2
c3	v1

  d est SPI car DF implique (RC) $\cap$ (CV) $\to$ (CV $\setminus$ RC).

  Mais que dévient $VR \to C$? Pas applicable.

  Insertion de $\langle r1, c3\rangle$ dans INFO1 :
  $\langle r1, c3, v1\rangle$ **et** $\langle r1, c1, v1\rangle \in$ INFO1 $\bowtie$ INFO2. Violation de $VR \to C$!

Intuition pour l'exemple précédent : ailas, pour savoir si une insertion préserve la contrainte $VR \rightarrow C$ il faudrait réfaire la jointure !

On aimerait travailler sans $\bowtie$!

Déf. La *fermeture* d'un ensemble de DF F, noté F^+, est l'ensemble de *toutes* les dép. impliquées par F.

Déf. La *projection* d'un ensemble de dép. F su'un ensemble d'attributs X est l'ensemble d'élément de F^+ (pas seulement de F !) comportant seulement des attributs de X. Notation : F_X.

Déf. Une décomposition d de S en $S_1 \cdots S_n$ est sans perte de dépendances (SPD) par rapport à un ensemble de dép. F ssi

$$(F_{S_1} \cup \cdots \cup F_{S_n})^+ = F^+$$

NB. $\subseteq$ est banal. C'est l'inclusion reciproque qui compte.

Exemple de calcul de F_{S_i}. Soit $S = \{A, B, C\}$, $S_1 = \{A, C\}$

$F = \{A \to B, B \to A, B \to C, C \to B\}$.

$F^+ = \{A \to A,\ A \to B,\ A \to C,$
$B \to A,\ B \to B,\ B \to C,$
$C \to A,\ C \to B,\ C \to C, ...\}$

F_{S_1} contient $A \to C$ et $C \to A$ (même si $C \to A \notin F$!) et toutes les dépendances impliquées par ces deux la.

Exemple de déc. Soit $S = \{A, B, C\}$, $S_1 = \{A, B\}$, $S_2 = \{B, C\}$,
$F = \{A \to B, B \to C, C \to A\}$. Soit d une décomposition de S en S_1 et S_2.

SPD ? Clair que l'on retrouve $A \to B$ et $B \to C$. Mais $C \to A$??

$B \to A \in F_{S_1}$
$C \to B \in F_{S_2}$
$\{C \to B,\ B \to A\} \subset (F_{S_1} \cup F_{S_2})$
$C \to A \in (F_{S_1} \cup F_{S_2})^+$
Donc $F \subseteq F_{S_1} \cup F_{S_2}$.
Donc $F^+ \subseteq (F_{S_1} \cup F_{S_2})^+$.

OUI, SPD ! Comparer avec l'exemple de p. 49...

- Critère 2 qu'une décomposition doit respecter : être SPD par rapport aux DF.
- Une déc. d peut être SPI sans être SPD : ex. vu.
- Une déc. d peut être SPD sans être SPI. Etudier la déc. de :

A	B	C	D
a	b	c1	d1
a1	b1	c	d

$F = \{A \to B, C \to D\}$

en AB et CD.

- Question. Existe-t-il un algo qui engendre une décomposition qui est forcement SPI **et** SPD par rapport aux DF ? **OUI**.

Préliminaires à la déf. de l'algo qui engendre une décomposition SPI **et** SPD par rapport aux DF

Soit $X \rightarrow Y$ une dép. fonct. et soit DF un ensemble de dép. fonct..

DEFINITIONS

- $X \rightarrow Y$ est réduite à droite si Y est un singleton.
- $X \rightarrow Y$ est réduite à gauche (par rapport à DF) s'il n'existe pas de X' strictement inclus dans X t.q. DF implique $X' \rightarrow Y$.
- DF est redondant s'il existe une dép. fonct. $Z \rightarrow W \in DF$ t.q. $DF \setminus \{Z \rightarrow W\}$ implique $Z \rightarrow W$.
- DF est réduit si
  1. DF n'est pas redondant
  2. $\forall\, f \in DF$, f est réduite à gauche **et** à droite.

Exemple.

Soit $DF=$

$$\{AB \to C, B \to A, A \to D, D \to C\}$$

1. Les dép. de DF sont réduites à droite, mais ne sont pas réduites à gauche.
   On a : $AB \to C \in DF$, mais DF implique $B \to C$.
   (Calculer $\{B\}^+_{DF}$!)
2. DF est redondant, car $DF \setminus \{AB \to C\}$ implique $AB \to C$.
   (Calculer $\{AB\}^+_{DF\setminus\{AB\to C\}}$!)

- **Définition**

  Soit F un ensemble de dép. fonct. Une couverture de F est un ensemble G de dép. fonct. tq. F est équivalent à G (c.à.d. : $F^+ = G^+$). Une couverture minimale de F est un ensemble G de dép. fonct. tq. G est réduit et F est équivalent à G.

- **Exemple très simple**

$$F = \{A \rightarrow BC, AC \rightarrow D, A \rightarrow D\}$$

  F n'est pas réduit : $A \rightarrow BC$ n'est pas réduite à droite, $AC \rightarrow D$ n'est pas réduite à gauche, et F est redondant, car $F \setminus \{A \rightarrow D\}$ implique $A \rightarrow D$.

  Soit :

$$G = \{A \rightarrow B, A \rightarrow C, A \rightarrow D\}$$

  G est une couverture minimale de F (développement au tableau).

- Calcul mécanique d'une couverture minimale pour un ensemble de dép. fonct. quelconque ?

Algorithme de Calcul d'une Couverture Minimale

Entrée : Un ensemble F de dép. fonct.

Sortie : Une couverture minimale G de F

1. *Initialisation* : $G{:=}F$;
2. **Faire**, dans l'ordre :
   (a) Remplaçer toute $X \to A_1, \cdots, A_n$ de F tq $n > 1$ par les dép.
   $X \to A_1, \ \cdots \ , X \to A_n$.
   $G :=$ l'ensemble de dép. ainsi obtenu;
   (b) Remplacer toute $X \to Y \in G$ par $X' \to Y$ où X' est le sous-ensemble minimal de X tq G implique $X' \to Y$.
   $G :=$ l'ensemble de dép. ainsi obtenu;
   (c) Tant que il existe $X \to A \in G$ tq $G \setminus \{X \to A\}$ implique $X \to A$,
   faire $G := G \setminus \{X \to A\}$;
3. **Retourner** G

Attention ! Il faut faire les étapes 2a et 2b avant létape 2c !!

Exemple

On fait tourner l'algo de calcul d'une couverture minimale sur :
$F_1 = \{ABCD \to E, E \to D, A \to B, AC \to D\}$

Mauvaise façon !

1. On fait l'étape 2c tout de suite : rien est redondant !
2. Etape 2a : rien á faire.
3. Etape 2b : we remplace $ABCD \to E$ par $AC \to E$.

Fini ! On a obtenu $F_2 = \{AC \to E, E \to D, A \to B, AC \to D\}$ qui est équivalent à F_1, mais qui contient la dépendance redondante $AC \to D$! Une couverture minimale aurait été

$$F_3 = \{AC \to E, E \to D, A \to B, \}$$

On fait à nouveau tourner l'algo de calcul d'une couverture minimale sur : $F_1 = \{ABCD \to E, E \to D, A \to B, AC \to D\}$, mais avec le bon ordre des étapes.

1. On fait l'étape 2a : rien á faire
2. Etape 2a : on obtain $F_1' = \{AC \to E, E \to D, A \to B, AC \to D\}$
3. Etape 2c : on élimine $AC \to D$.
4. , Fini, avec le bon ensemble $F3$!

On est prêts à donner l'algo pour le calcul d'une déc. qui soit SPI et SPD $\rightsquigarrow$ page suivante.

Algorithme qui génére une dec. SPI et SPD

Entrée : Un schéma de relation S, un ensemble F de dép. fonct. sur S.
Sortie : Une décomposition d de S qui est SPI et SPD par rapport à F.

1. Choisir un ensemble de DF $F' = \{X_1 \rightarrow A_1, \cdots, X_n \rightarrow A_n\}$ tq F' est une couverture minimale de F;
2. Pour chaque $f_i = X \rightarrow A_j \in F'$, créer un schéma $S_i = XA_j$ et poser d= l'ensemble de ces S_i.
3. Si aucune des clés de S (calculée grâce à F') n'est contenue dans un élément de d, modifier d en y ajoutant une clé Y comme élément.
4. S'il existe 2 éléments S_i et S_j de d tels que $S_i \subset S_j$ supprimer S_i, afin d'obtenir la valeur courante de d.
5. Si on a des éléments de d de la forme $XA_1, \cdots, XA_k$ obtenus car $X \rightarrow A_1, \cdots, X \rightarrow A_k$ sont des éléments de F', les remplaçer par l'unique (sous)schéma $XA_1...A_k$ afin d'obtenir la valeur finale de d.

N.B Si plusieurs choix de couvertures min. et de sur-clés, possibilité de plusieurs décompositions.

Exemple d'application.

Soit $S = ABCDE$. Soit

$$F_1 = \{ABCD \to E, E \to D, A \to B, AC \to D\}$$

Application de l'algo de décomposition SPI et SPD : au tableau.

4.3.1 Formes Normales : Motivations

Soit $S = ABC$ et $F = \{A \rightarrow B, BC \rightarrow A\}$

Soit la relation r satisfaisant toutes les dép. de F :

A	B	C
a	b	c
a	b	c'

Pas de redondance sur les valeurs de BC : 1 seul couple de valeurs sur BC, car BC est une clé. Redondance sur AB. Pourtant, puisque $A \rightarrow B \in F$, répétition de ab inutile !

Redondances $\rightsquigarrow$ risques d'anomalies d'insertion, suppression, etc. .

- **Déf.** Soit S un schéma de relation et F un en ensemble de dépendances sur S. S es en Forme Normale de Boyce Codd (BNCF, dite aussi 4FN) par rapport à F ssi $\forall\ X \rightarrow Y$ impliquée par F, soit $X \rightarrow Y$ est "triviale" (i.e. $Y \subseteq X$) soit X est une sur-clé (par rapport à F).

- **Exemple précédent** : $S = ABC$ et $F = \{A \rightarrow B, BC \rightarrow A\}$. Ici, S n'est pas BNCF !
  $A \rightarrow B \in F$, non-triviale, mais A ne suffit pas à déterminer C $\Rightarrow$ Source de la redondance sur AB.

- Tous les schémas de relations en BNCF : souhaitable, car on n'aura jamais de redondances de données.
  Toutefois : parfois trop fort ! Par ex., $\nexists$ déc. pour l'exemple ci-dessus qui soit, au même temps, SPI, SPD et FNBC !

On relâche des contraintes imposées par FNBC...

S=schéma de relation, F ensemble de dépendances fonctionnelles sur S.

- Un attribut A est premier (par rapport à F) ssi $\exists$ une clé C tq $A \in C$
- **Déf.** Soit S un schéma de relation et F un en ensemble de dépendances sur S. S est en 3FN par rapport à F ssi $\forall\ X \rightarrow A$ impliquée par F, où A est un attribut :
  - soit $X \rightarrow A$ est "triviale" (i.e. $\{A\} \subseteq X$)
  - soit X est une sur-clé (par rapport à F),
  - soit A est premier.

Pourquoi si NON-3FN alors problèmes potentiels ?

Soit $X \to A$ non-triviale et impliquée par F, A pas premier. Cas de violation 3NF :

1. X n'est pas une surclé et $X \subset C$, où C est une clé. On dit : dépendance partielle.
   Risque : stocker des valeurs (X, A) plusieurs fois.
   Ex. Reservation(Marinier,Bateau,Jour,CarteC). $F = \{M \to C\}$.
   Clé=$\{$M,B,J$\}$. Ici, $X = \{M\}$, $A = C$. On stocke le numéro de CarteC pour un marinier plusieurs fois.

2. X n'est pas une surclé et, quelque soit la clé C, $X \not\subset C$. On dit : dépendance transitive.
   Ex. T(S,N,L,R,W,H). $F = \{S \to NLRWH, R \to W\}$. Clé : S. Mais on a la chaîne de DF : $S \to R \to W$. Ici : $X = \{R\}, A = W$. On ne peut pas associer une valeur de S avec une valeur de R sans connaître la valeur de W.Risque : anomalies d'insertion (ou de m. à j.).

N.B. : Si S est en FNBC alors S est aussi en 3FN, l'implication réciproque n'est pas toujours vraie.

Dans l'exemple déjà vu : $S = ABC$ et $F = \{A \rightarrow B, BC \rightarrow A\}$, S n'est pas BNCF, car $A \rightarrow B$ n'est pas triviale et A n'est pas une super-clé.

Mais on a 4FN.
En fait : BC est une clé, donc B est premier; $A \rightarrow B$ viole BNCF, mais ne viole pas 3FN.

Algo pour décomposer de façon à assurer SPI, SPD **et** 3FN ?

L'algo de décomposition SPI, SPD déjà donné marche !

Application de l'algo à :
$S = ABCDE$,

$$F = \{A \to BC, C \to A, D \to E, C \to B\}$$

au tableau.