Systèmes de Gestion de Bases de Données

L3 Informatique : parcours ASR, Informatique, MIAGE initial et apprentissage

S. Cerrito

Premier semestre 2015-2016

Plan du cours

1. Introduction et notions de base des BD relationnelles;
2. Fondements des langages de requête : Algèbre Relationnelle et Calcul Relationnel à variables *n*-uplets;
3. SQL
4. Conception de schéma :
   - 4.1 Contraintes d'intégrité : les dépéndances fonctionnelles;
   - 4.2 Raffinements du schéma :
     - Anomalies;
     - Décompositions SPI et SPD;
     - Formes Normales
5. Si temps : stockage physique des données (indexes etc.)

Coordonnées

Mon adresse e-mail :
Serena.Cerrito@ibisc.univ-evry.fr

Si besoin de me rencontrer, prendre RDV par e-mail.

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

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 $\Rightarrow$
   - 3.1 possibilité d'un *langage de requêtes* facile pour l'utilisateur
   - 3.2 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

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).
- Après 2000 : popularité de *NoSQL*

Ce cours : modèle relationnel, le plus utilisé dans la pratique (même si XML... Mais voir le cours de BDA du M1 !).

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

Notions essentielles des BD relationnelles

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 associé à A.

Exemple : $U =$
$\{NomFilm, Realisateur, Acteur, Producteur, NomCinema, Horaire\}$
$Dom(NomFilm) = Dom(Realisateur) = Dom(Acteur) =$
$Dom(Producteur) = Dom(NomCinema)=$ chaînes de caractères.
$Dom(Horaire) = \{h.m \mid h \in [0, \cdots, 23],\ m \in [0, \cdots, 59]\}$

Notions essentielles des BD relationnelles

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
⋮	⋮	⋮

Notions essentielles des BD relationnelles

Soit $E = \{A_1, \cdots, A_n\}$ le schéma d'une relation. Un *n-uplet n* sur E est une suite de n éléments de la forme : $v_i : A_i$ où $1 \leq i \leq n$ et $v_i \in Dom(A_i)$.

Exemple.
Un n-uplet possible sur le schéma de *Projection* :
$\langle$"*Bird*" : *NomFilm*, "*Gaumont Alesia*" : *NomCinema*, 13.35 : *Horaire*$\rangle$
Si pas de confusion possible, on notera plus simplement :
$\langle$"*Bird*", "*Gaumont Alesia*", 13.35$\rangle$

Pourquoi on mentionne les attributs, dans la définition formelle de n-uplet ?

Notions essentielles des BD relationnelles

Si t est un n-uplet sur E et si $E' \subset E$, la restriction de t à E' se note $t(E')$.

Exemple.
La restriction de
$\langle''Bird'' : NomFilm,'' Gaumont\ Alesia'' : NomCinema,\ 13.35 : Horaire\rangle$
à $\{NomFilm, NomCinema\}$ est :

$$\langle''Bird'' : NomFilm,'' Gaumont\ Alesia'' : NomCinema,\rangle$$

ou

$$\langle''Bird'','' GaumontAlesia''\rangle$$

Notions essentielles des BD relationnelles

Une *relation* (table) r sur un schéma de relation S est un ensemble fini de 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

Notions essentielles des BD relationnelles

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 ?

Notions essentielles des BD relationnelles

- Une *base de données* B sur un schéma de base $\mathcal{S}$ (avec univers U) est un ensemble de relations (finies) $r_1, \cdots r_n$ où chaque r_i est associée à un nom de relation N_i;
  Si $N_i(S) \in \mathcal{S}$, alors r_i a S comme schéma.
  (Si le schéma $\mathcal{S}$ de la base dit que la relation nommée N_i doit avoir l'ensemble d'attribut S, alors la relation r_i "obeit"...)
- On peut aussi imposer des *contraintes* sur les données. Par exemple : les *dépendances fonctionnelles* (DF, 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.

Notions essentielles des BD relationnelles

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
nf4	r1	a1	p1

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

Un ex. de contrainte (qui n'est pas une DF) : *Toute valeur de la colonne NomFilm de Projection doît apparaître aussi dans la colonne NomFilm de Film.*

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 (SQL etc.)
  2. Evaluation de la puissance d'expression de chaque langage commercial
  3. Notion d'équivalence entre deux expressions de requête $\Rightarrow$ Optimisation "logique" de l'évaluation d'une requête

Fondements des Langages de Requête

Langages formels à voir dans ce cours : *algèbre relationnelle* (AR) et *Calcul Relationnel* (CR).

CR : fondé sur la logique des prédicats.

Langage commerciel SQL : des notions viennent de l'algèbre, d'autres du CR (version dite *à variables n-uplet*).
AR et CR équivalents, mais :

- Algèbre : "procédurale" (analogie : langage de programmation C)
- Calcul Relationnel : "déclaratif" (analogie : langages de programmation Caml et Prolog)

Les opérateurs de l'algèbre relationnelle

- 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).

Union, intersection, différence

Arguments : 2 relations r et r' de même schéma S. Résultat : une nouvelle relation, encore sur S.

Notation : ici et après, n indique un n-uplet.

$$r \cup r' = \{n \mid n \in r \text{ ou } n \in r'\}$$

$$r \cap r' = \{n \mid n \in r \text{ et } n \in r'\}$$

$$r \setminus r' = \{n \mid n \in r \text{ et } n \notin r'\}$$

Projection

Notation : π_E, où E= ensemble d'attributs.

Arguments : 1 relation r. Résultat : une nouvelle relation dont le schéma est inclus dans celui de r.

Si S = schéma de r, alors $E \subseteq S$.

$$\pi_E(r) = \{n(E) \mid n \in r\}$$

Ecriture équivalente :

$$\pi_E(r) = \{m \mid \exists n\, (n \in r \text{ et } m = n(E))\}$$

Sélection

.

- **Condition de Sélection** C.
  Atomes : A_i *op* A_j ou A_i *op* v où :
  A_i et A_j sont des attributs, $v \in Dom(A_i)$,
  $op \in \{=, \neq, >, <, \geq, \leq\}$.
  C est une formule booléenne construite à partir des atomes.
- **Opérateur de Sélection** σ_C. Arguments : 1 relation r.
  Résultat : une nouvelle relation sur le même schéma que r.

$$\sigma_C(r) = \{n \mid n \in r \text{ et } n \text{ satisfait } C\}$$

Produit Cartesien et Jointure Naturelle

- **Produit Cartésien** $\times$. Arguments : 2 relations r et r', de schémas S et S', telles que S et S' sont disjoints. Résultat : une nouvelle relation dont le schéma est $S \cup S'$.

$$r \times r' = \{n \ sur \ S \cup S' \ \mid \ n(S) \in r \ et \ n(S') \in r'\}$$

- **Jointure "naturelle"** $\bowtie$. Arguments : 2 relations r et r', de schémas S et S'. Résultat : une nouvelle relation dont le schéma est $S \cup S'$.

$$r \bowtie r' = \{n \ sur \ S \cup S' \ \mid \ n(S) \in r \ et \ n(S') \in r'\}$$

N.B. : Si S et S' sont disjoints, le résultat de $r \bowtie s$ est le même que celui de $r \times s$. Donc : $\times$ = cas particulier de $\bowtie$.

Division

Arguments : 2 relations r et r', de schémas S et S' tels que $S' \subset S$. Résultat : une nouvelle relation dont le schéma est $S \setminus S'$.
Notation : si n et x sont 2 n-uplets, notons $n \bowtie x$ l'unique élément de la relation $\{n\} \bowtie \{x\}$.

$$r \div r' = \{n \text{ sur } S \setminus S' \mid n \in \pi_{S \setminus S'}(r) \text{ et } \forall x \in r', \ n \bowtie x \in r\}$$

Exemple.

AimeLivre

Personne	NomLivre
p1	l1
p2	l2
p1	l2

Livre

NomLivre
l1
l2

"Qui aime tous les livres ?" : $AimeLivre \div Livre$

Renommage

Arguments : 1 relation r , de schéma S, un attribut $A \in S$ et un nouveau attribut $A' \notin S$. Résultat : une copie de la relation r où l'attribut A est renommé en A'.
Schéma de la copie : $(S \setminus \{A\}) \cup \{A'\}$.

Ecriture : $\rho_{A \rightsquigarrow A'}(r)$

Exemple :

$\rho_{NomLivre \rightsquigarrow NomLivre2}(AimeLivre)=$

Personne	NomLivre2
p1	l1
p2	l2
p1	l2

"Qui aime au moins deux livres ?" :

$$\pi_{Personne}(\sigma_{NomLivre \neq NomLivre2}(AimeLivre \bowtie \rho_{NomLivre \rightsquigarrow NomLivre2}(AimeLivre)))$$

Est-il possible de se passer de ρ ??

Requêtes

Requête : expression d'un langage L qui, évaluée sur une BD calcule une relation.
Par ex. pour une BD sur le cinèma (schéma déjà vu, mais notations raccourcies) :

- Requête R1, L=français :

  Quels spectateurs aiment au moins deux films différents réalisés par Ken Loach ?

- Requête R1, L= langage de l'algèbre relationnelle :
  - Posons : $Aime_{KL} = (\sigma_{Real='K.L.'}\pi_{NomFi,Real}(FILM)) \bowtie Aime$
  - Alors :
    $\pi_{Spe}[\sigma_{NomFi \neq NomFi2}\ (Aime_{KL} \bowtie (\rho_{NomFi \rightsquigarrow NomFi2}(Aime_{KL})))]$

Requêtes et expressions algébriques

- Expression E de l'algèbre relationnelle = mot construit "proprement" en utilisant les opérateurs de l'algèbre.
- Les expressions de l'algèbre calculent des requêtes : la réponse à la requête E_1 pour une BD est la relation résultat de l'évaluation de l'expression algébrique E_1 sur la BD.

Equivalence entre expressions algébriques, suite

U : univers, $\mathcal{S}$: schéma de base sur U

- E est équivalent à E' ($E \equiv E'$) ssi E et E', évaluées sur la même base, calculent toujours la même requête.
- Pour *demontrer* que $E_1 \equiv E_2$ on montre que : qque soit le n-uplet n, $n \in$ réponse à E_1 ssi $n \in$ réponse à E_2.
  Utilisation des définitions des opérateurs de l'algèbre.
  Exemple : Si tout attribut de $C \in$ schéma de r, alors $\sigma_C(r \bowtie s) \equiv (\sigma_C(r)) \bowtie s$.
- Pour montrer q'une $\equiv$ est fausse un contre-exemple suffit.
  Exemple. Soit S le schéma de r_1 et r_2, soit $A \in S$. C'est faux que $\pi_A(r_1 \cap r_2) \equiv (\pi_A(r_1)) \cap (\pi_A(r_2))$.
  Prendre $S = \{personne,\ livre\}$, $A = personne$, r_1 et r_2 non-vides et disjointes, r_1 et r_2 ayant une même valeur pour A.
- Utilité des $\equiv$: *optimisation algébrique*. Par ex., quelle est l'expression la plus coûteuse parmi $\sigma_C(R \bowtie S)$ et $(\sigma_C(R)) \bowtie S$?

Calcul Relationnel à Variables n-uplets (CR)

C'est un langage de la logique des prédicats avec plusieurs *sortes* ou "*types*".
Une variable x aura un type T et on note $x : T$.

Les valeurs des variables sont des n-uplets de la base.

Lien intuitif avec SQL $\rightsquigarrow$

CR

Relation CINEMA, dont deux des attributs sont nom_cinema et adresse, relation SEANCE, dont deux des attributs sont nom_cinema et id_film, et relation FILM dont deux des attributs sont id_film et nom_realisateur.
Quel cinémas ne projettent aucun film réalisé par Tarantino ? Donner aussi les adresses.

```
SELECT c.nom_cinema, c.adresse
FROM  cinema c
WHERE NOT EXISTS (SELECT s.nom_cinema
            FROM seance s, film f
           WHERE f.nom_realisateur = 'Tarantino'
            AND   f.id_film = s.id_film
           AND   s.nom_cinema = c.nom_cinema);
```

Ici, c est une variable dont les valeurs (successives) sont les n-uplets (lignes) de la table SEANCE.
NOT EXISTS (...) code une formule logique qui s'évalue à Vrai ou Faux, selon les valeurs de c.

CR

Deux différences avec la logique des prédicats du cours de logique du L2.
1) La valeur d'une variable n'est pas un individu "indécomposable", mais un n-uplet. Elle a donc des "composantes" :
si $x : ABC$ vaut $\langle a1 : A, b1 : B, c1 : C \rangle$, alors on a une syntaxe permettant d'accéder à la composante $a1$, à la composante $b1$ et à la composante $c1$.
On peut voir le "$.Attribut$" comme un *symbole de fonction* qui associe à un individu, le triplet x, un autre individu du domaine de discours, sa composante $x.Attribut$ (cours de logique du L2).

CR

2) Les variables sont typées.
Utilité ?
Par ex. on a : univers $= \{A, B, C, ..., Z\}$. Parmi les tables, on a r, de schéma $\{A, B, C\}$ (souvent on note juste ABC). On veut exprimer, en logique : tous les n-uplets de r ont une propriété Q.
Si on ne type pas :

$$\forall x(si\ x \in r\ alors\ Q(x))$$

Pour évaluer, on examine **tous** les n-uplets possibles :
avec une seule composante, sur A, avec une seule composante, sur B, $\cdots$ avec une seule composante, sur Z,
avec 2 composantes, sur AB, sur AC,.....,WZ
$\vdots$
avec 26 composantes, sur $AB...Z$
Mais on est intéressé seulement aux n-uplets sur ABC ! Alors :

$$\forall x : ABC(si\ x \in r\ alors\ Q(x))$$

CR

Encore sur les principes du Calcul Relationnel à variables n-uplets (CR à n-uplets).

- Une variable aura comme valeur un n-uplet qui PEUT apparaître dans la base (bon type), mais qui n'y est pas forcement.
- Notation : si x est une variable (n-uplet) et A un attribut, l'expression $x.A$ indique la valeur pour l'attribut A du n-uplet qui est la valeur de x (syntaxe pour accéder aux composantes).

CR

Pour un schéma de base $\mathcal{S}$ donné sur un univers U, le vocabulaire des formules du calcul relationnel est :

- $V=$ ensemble infini de variables.
- $C =$ ensemble de constantes $= \bigcup_{A_i \in U} Dom(A_i)$
- Symboles de prédicat : $=$, $>$, $\geq$, $<$, $\leq$, et tout nom de relation dans $\mathcal{S}$.

On a aussi une fonction *type* qui associe à certaines variables des sous-ensembles de U.
Pour $t \in V$, lire $type(t)$ comme "le type de t".

CR

Syntaxe des **Formules Atomiques** (ou Atomes) (F.A.) :

1. **Si** :
   $t, t' \in V$, A est un attribut $\in type(t)$ et B est un attribut $\in type(t')$
   alors le mot $t.A = t'.B$ est une F.A.
2. **Si** $t \in V$, A est un attribut $\in type(t)$, $k \in C$ et $k \in Dom(A)$
   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 le mot $R(t)$ est une F.A.

Exemples de F.A.

Soit *Projection* un nom de relation, dont le schéma est $\{NomFilm,\ NomCinema,\ Horaire\}$ et soit *Aime* une relation de schéma $\{Spectateur, NomFilm\}$.
Soient t et t' des variables, avec
$type(t) = type(t') = \{NomFilm,\ NomCinema,\ Horaire\}$.
Les mots :

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

sont des F.A.
Le mot $Aime(t)$ n'est pas une F.A. : erreur de typage !

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 !

CR

Exemple : Base B

AimeLivre

Personne	NomLivre
p1	l1
p2	l2
p1	l2

Livre

NomLivre
l1
l2

Donnons la valeur $\langle p1, l1 \rangle$ 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.
$t.NomLivre =$ l1 est vraie.
$AimeLivre(t)$ est fausse.

Formules

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

Formules : suite

Lecture intuitive de "$\forall t : E\ FF$" : quelque soit le n-uplet t de type E, F est vraie pour t.

Lecture intuitive de "$\exists t : E\ F$" : il existe au moins un n-uplet t de type E tel que F est vraie pour t.

Formules : suite

La même définition de l'ensemble des formules par une grammaire :

$Formule := F.A. \mid (\neg Formule) \mid (Formule * Formule)$
$\mid (\forall t : E\ Formule) \mid (\exists t : E\ Formule)$

où $* \in \{\wedge, \vee, \rightarrow\}$.

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 (mais types en plus).

CR

Pour $(\forall t : E\ F)$ et $(\exists t : E\ F)$:
$F =$ *portée* de $\forall t$ (resp. $\exists t$).

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

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

$$Ensemble\ des\ variables\ libres\ de\ F = \{t'\}$$

F va être vraie pour certain valeurs de t', fausse pour d'autres.

Signification (= Sémantique) des formules

F : formule, B : base.
s : choix de valeurs pour les variables libres de F.
t : variable n-uplet, $type(t) = E$.

Par rapport à B et s 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 si 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 (ayant le type E), F_1 est vraie pour t.
- Si F est $\exists t : E\ F_1$, F est vraie ssi il existe au moins une valeur de t (ayant le type E) telle que F_1 est vraie pour t.

CR

Révenons à la syntaxe et au typage.
Erreurs d'écriture possibles :

Non-Formules :

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

   Erreur de syntaxe, pas (forcement) de typage.
2. $Aime(t) \wedge t.age = 40$ où
   $type(t) = \{NomPersonne, NomLivre\}$
   Erreur de typage!
3. (Rappel : le schéma de *Livre* est $\{NomLivre\}$)

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

   Erreur de typage!

CR

Exemples suivants : sur le schéma de base du premier cours :
$U =$
$\{NomFilm, Realizateur, Acteur, Producteur, NomCinema, Horaire, Spectat$

$$\mathcal{S} =$$

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

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

- Format :

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

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

- 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 ?

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

Un autre exemple.

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

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

NB : En algèbre on aurait écrit :

$$\pi_{NomFilm,Acteur}\sigma_{NomCinema=''GaumontAlesia''}(Film \bowtie Projection)$$

CR

Question. Même base B, mais domaines des attributs $\neq$.
A-t-on forcement une réponse unique à une requête dans le calcul ?
NON !

Exemple : Schéma de la base $=\{R(A)\}$, c.-à-d. on a une seule table, R, dont le schéma est $\{A\}$.

A
1
2
3

Trois Cas
1) $Dom(A) = \{1, 2, 3\}$
2) $Dom(A) = \{1, 2, 3, 4\}$
3) $Dom(A) = \{1, 2, 3, 4, 5, 6, ...\}$

Requête : $\{t : A \mid \neg R(t)\}$
Réponses respectives dans les trois cas : $\emptyset$, $\{4\}$, $\{n \mid n$ est un entier positif et $n > 3\}$.
Dans le troisième cas, la réponse est même infinie ! Pas de correspondance avec AR.
Cette requête est dépendante du domaine. Pas Bien !

CR

- $\exists$ 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.).

CR

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. : $\{t : NomFilm \mid \forall x : Spectateur, NomFilm$
$(x.NomFilm = t.NomFilm \rightarrow \neg Aime(x))\}$
Evaluer, par exemple, sur $Aime = \{\langle s1, f1\rangle, \langle s2, f2\rangle\}$ avec
$Dom(NomFilm) = \{f1, f2, f3, ...\}$.

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

Exemples, suite

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

D.D. : $\{t : NomFilm \mid \neg\exists t'' : NomFilm, NomCinema, Horaire$
$(Projection(t'') \wedge t''.NomCinema = \text{“}Gau.Al.\text{”} \wedge t''.NomFilm = t.NomFilm)\}$
Evaluer, par exemple, sur
$Projection = \{\langle f1, c1, h1\rangle, \langle f2, G.A., h22\rangle\}$ avec
$Dom(NomFilm) = \{f1, f2, f3, ...\}$.

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

Exemples, suite

Encore :
"Quels films ne sont pas projettés au Gaumont-Alésia ?"

Les deux formules suivantes sont I.D. et sont **équivalentes** :

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

(déjà vue).

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

Pourquoi ? $\neg\exists x F(x) \equiv \forall x \neg F(x)$

Et encore une autre écriture équivalente si on remarque que $\neg(A \wedge B) \equiv (A \rightarrow \neg B)$.

CR

- 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.

- Une relation n'est pas implémentée comme un ensemble, mais comme un *multi-ensemble* (m.e.).
- Multi-ensemble : les répétitions sont permises, mais l'ordre ne compte pas. Par ex., dans le cas des m.e. :
  $\{1, 4, 2, 5\} = \{1, 2, 4, 5\} \neq \{1, 2, 4, 5, 5\} \neq \{1, 2, 4, 5, 5, 5\}$
  $\{1, 4, 2, 5\} \cup \{1, 7\} = \{1, 1, 4, 2, 5, 7\}$
- Pourquoi ce choix de structure de données ? Rapidité des calculs ! Penser au calcul de l'union ensembliste...

Format d'une Requête Simple

A1,...,AN : attributs, R: nom de relation

SELECT A1,...,AN
FROM R
WHERE C;

Ici, C est une condition de sélection sur les n-uplets de R, comme dans le σ_C de AR.

Le SELECT fait une projection (op. π de AR) du résultat de la sélection sur les attributs A1,...,AN. Le σ_C est simulé par le WHERE !

SQL

Exemple de schéma de base pour illustrer SQL

$\mathcal{S} =$
$\{Films(Titre, Date, Longueur, Couleur, NomStudio, IdProducteur),$
$Joue(NomFilm, FilmDate, NomActeur, Paye),$
$Acteur(Nom, Adresse, Sex, Date\_Naissance),$
$Studio(Nom, Adresse),$
$Producteurs(Nom, Adresse, Id)\}$

```
SELECT Titre, Longueur
FROM Films WHERE NomStudio='Disney' AND Date=1990;
```

Réponse : une table

Titre	Longueur
Pretty Woman	119
⋮	⋮

SQL

```
SELECT *
FROM Films
WHERE NomStudio='Disney' AND Date=1990;
```

* est une syntaxe pour indiquer tous les attributs de R

Réponse : une table

Titre	Date	Longueur	Couleur	NomStudio	IdProducteur
Pretty Woman	1990	119	Vrai	Disney	1
⋮	⋮	⋮	⋮	⋮	⋮

Utilisation d'*alias*

```
SELECT Titre AS Nom, Longueur AS Durée
FROM Films
WHERE NomStudio='Disney' AND Date=1990;
```

Réponse : une table

Nom	Durée
Pretty Woman	119
⋮	⋮

C'est le rennomage ρ de l'AR !

SQL

Conditions de sélection plus complexes

SELECT Titre
FROM Films
WHERE (Date $>$ 1970 OR Longueur $<$ 90) AND NOT Couleur;

NB :
1) Partie WHERE C d'une requête SQL = opérateur σ_C de l'algèbre relationnelle !
2) Ici, Couleur est de type bool;

Valeurs Nulles

- A différence de l'algèbre, une colonne peut avoir la valeur NULL.
- "NULL *op valeur*" s'évalue NULL si *op* est un opération arithmétique ($+, \times$ etc.).
- "NULL *rel valeur*" s'évalue UNKNOWN, si *rel* est $=$ ou $>$ ou $<$".

SELECT *
FROM Films
WHERE Date$>=$ 1970 OR Date $<$ 1970;
renvoie la table Films
privée des *n*-uplets où la valeur de Date est NULL. Pourquoi ?

Requêtes sur Plusieurs Relations

Soient R1,...,Rk des relations.

Format :

SELECT A1,...,AN
FROM R1,...,Rk
WHERE C;

Ici, la relation dans laquelle on va chercher les n-uplets est le produit cartésien des relations nommées R1,...,Rk.

SQL

Exemple.
"Quel est le nom du producteur de "Star Wars" ?

```
SELECT Nom
FROM Films, Producteurs
WHERE Titre ='Star Wars' AND IdProducteur=Id;
```

NB. : Ici, on a utilisé un AND, pour simuler $\bowtie$, comme en Calcul relationnel !

SQL

En algèbre, le produit cartésien requiert que les schémas des arguments soient disjoints. Et si des attributs sont communs ?

Exemple(Acteur et Producteur partagent les attributs Nom et Adresse)

"Quels acteurs et quels producteurs ont la même adresse ?
SELECT Acteur.Nom, Producteurs.nom
FROM Acteur, Producteurs
WHERE Acteur.Adresse=Producteurs.Adresse;

Réponse :

Acteur.Nom	Producteur.Nom
Jane Fonda	Ted Turner
⋮	⋮

Renommage des attributs partagés, encore une implementation de l'operateur ρ de l'AR.

SQL

Ordonnancement des n-uplets ($\notin$ AR !)

Format :
Si { B1,...,Bk} $\subseteq$ A1,...,An

SELECT A1,...,An
FROM R
WHERE C
ORDER BY B1,...,Bk;

Exemple
SELECT *
FROM Films
WHERE NomStudio='Disney' AND Date=1999
ORDER BY Longueur, Titre;

Dans la réponse, si n(Longueur)$<$ n'(Longeur) alors n avant n' et si n(Longueur)=n'(Longueur), n avant n' ssi n(Titre) $<$ n'(Titre).

Quelques autres Opérations de l'Algèbre et les Sous-Requêtes Imbriquées

Exemple

(SELECT Nom, Adresse FROM Acteur)
MINUS
(SELECT Nom, Adresse FROM Producteurs);

En AR :

$$\pi_{Nom,Adresse}(Acteur) \setminus \pi_{Nom,Adresse}(Producteurs)$$

SQL

Opérateur IN

"Quels sont les producteurs de films dans lesquels Harrison Ford joue ?"

```
SELECT Nom
FROM Producteurs
WHERE Id IN
            (SELECT IdProducteur
             FROM Films
             WHERE Titre IN
                    (SELECT NomFilm
                     FROM JOUE
                     WHERE NomActeur = 'Harrison Ford'
                     )
            );
```

SQL

Utilisation des variables n-uplets dans SQL

Utilité : raisonner sur plusieurs n-uplets d'un même relation, les comparer etc.

Exemple
"Quel acteurs ont la même adresse ?"

```
SELECT Star1.Nom, Star2.Nom
FROM Acteur Star1, Acteur Star2
WHERE Star1.Adresse = Star2.Adresse AND Star1.Nom <
Star2.Nom;
```

N.B : ici, Star1 et Star2 sont des variables pour des n-uplets. Chacune d'elles varie sur les n-uplets de la table Acteur. Ecriture équivalente :

```
SELECT t1.Nom, t2.Nom
FROM Acteur t1, Acteur t2
WHERE t1.Adresse = t2.Adresse AND t1.Nom < t2.Nom;
```

SQL

Suite de l'Exemple.

Format de la réponse :

Acteur1.Nom	Acteur2.Nom
Balwin	Basinger
Cruise	Fonda
⋮	⋮

Même requête SQL, mais **seulement** :
WHERE t1.Adresse = t2.Adresse

Risque de :

Acteur1.Nom	Acteur2.Nom
Balwin	Balwin
Balwin	Basinger
Basinger	Balwin
⋮	⋮

Pourquoi ?

Suite de l'Exemple

Si :
WHERE t1.Adresse = t2.Adresse AND t1.Nom <> t2.Nom
encore risque de redondances :

Acteur1.Nom	Acteur2.Nom
Balwin	Basinger
Basinger	Balwin
⋮	⋮

Quantification Sur Les Variables *n*-uplets, Opérateur EXISTS

Exemple

"Quels acteurs n'ont joué dans aucun films paru après 2000?"'

```
SELECT t1.NomActeur
FROM Joue t1
WHERE NOT EXISTS
            (SELECT t2.FilmDate
             FROM Joue t2
             WHERE t2.FilmDate > 2000 AND t1.NomActeur =
t2.NomActeur)
```

Comment on écrirait en CR ?

Quantification Sur Les Variables *n*-uplets, Opérateur ANY Exemple

"Quels titres ont été utilisés pour plusieurs films" ? (remakes)

```
SELECT t1.Titre
FROM Films t1
WHERE t1.Date < ANY
                (SELECT Date
                  FROM FILMS
                  WHERE Titre =t1.Titre);
```

"*any* date" en anglais (dans ce contexte) : au moins une date, peu importe la quelle.

Comment on écrirait en C.R. ?

SQL

Opérations de SQL qui ne sont pas dans l'AR

- Elimination des répétitions dans un multi-ensemble
- Aggregation
- Formation de groupes

SQL

DISTINCT : après le SELECT, produit une seule copie de chaque *n*-uplet du résultat.

Exemple

"Quel est l'ensemble des producteurs de films dans lesquels Harrison Ford joue" ?

```
SELECT DISTINCT Nom
FROM Producteur, Films, Joue
WHERE ProducteurId=Id AND
                    Titre = NomFilm AND
                    Date = FilmDate AND
                    NomActeur = 'Harrison Ford';
```

Les opérateur d'Aggregation SUM et AVG

Après le SELECT, appliqués à un attribut approprié : SUM somme les valeurs, AVG calcule la moyenne.

Exemples
Schema de Contrats :$\{NomActeur, NomFilm, Paye\}$.

```
SELECT SUM(Paye)
FROM Contrats
WHERE NomActeur = 'Harrison Ford';
```

```
SELECT AVG(Paye)
FROM Contrats
WHERE NomActeur = 'Harrison Ford';
```

Comportement semblable : MIN et MAX.

L'opérateur d'Aggregation COUNT

Après le SELECT, appliqués à un attribut, compte le nombre des valeurs.

Exemples

```
SELECT COUNT(NomActeur)
FROM Joue;
```

```
SELECT COUNT(DISTINCT NomActeur)
FROM Joue;
```

Différence ?

Cas particulier où on compte toutes les n-uplet de la table :

```
SELECT COUNT(*)
FROM Joue;
```

Regroupements : l'opérateur GROUP BY

Après la partie WHERE C, appliqué à un attribut A, partitionne les n-uplets selon la valeur de A.

Exemple.

On veut, film par film, la liste des acteurs qui jouent dans ce film.

```
SELECT Titre, NomActeur
FROM Films, Joue
WHERE Titre = NomFilm
GROUP BY Titre ;
```

Exemple, suite

Format de la réponse :

Titre	NomActeur
Pretty Woman	Richard Gere
	a2
	a3
Eyes Wide Shut	Nicole Kidman
	TomCruise
	⋮

Concevoir et raffiner des schémas de bases

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*) (EA ou ER).
3. Passage de la représentation graphique au schéma relationnel (avec ses *dépendances fonctionnelles* : les DF, une classe de contraintes sur les données) : 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.

Concevoir et raffiner des schémas de bases

Dépendances fonctionnelles (DF) :
fondamentales pour la problèmatique de la conception de schéma.

Les DF jouent déjà un rôle dans la phase EA :

Fixer la clé "primaire" d'une table R

=

Fixer certaines DF qui devront être satisfaites par le contenu de la table.

DF

- Dépendances fonctionnelles : un type de contraintes d'intégrité.
- 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.
  Aussi : Y dépend fonctionnellement de X.
  Signification : La table 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 peut écrire, si pas de confusion possible :
  $A_1 \cdots A_n \to B_1 \cdots B_m$ (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
Les Choristes	2004	152	True	Pathé	886

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.

DF

- Associer un ensemble de *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 pour la table r 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, *Date* seul non plus.
    Titre Date Couleur n'est pas une clé mais c'est une *super clé*.
  - *Super clé* de r = ? Un sous-ensemble S' de S tel que $\exists$ une clé C de r t.q. $C \subseteq S'$.

DF et clés

Problème : Comment fixer par avance, les clés (candidates) d'une table R à partir du schéma de R et des DF ?

- Exemple. Schéma(R)=$\{A, B, C, D\}$,
  $F = \{AB \rightarrow C,\ C \rightarrow D\}$.
  La dép. fonct. $AB \rightarrow A\ B\ C\ D$ est **impliquée** par F.
  Donc, AB sera une clé (ou une super clé) de toute table associée à R satisfaisant toutes les dépendances de F.
- "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. éléments de F satisfait aussi f.

Exemples

- $F_1 = \{A \to B,\ B \to CD\}$ implique $A \to CD$ et implique aussi $A \to ABCD$.
- $F_2 = \{A \to B,\ A \to C\}$ implique $A \to BC$.
- $F_3 = \{A \to BC\}$ implique $A \to B$ et $A \to C$.
- $F_4 = \{AB \to C,\ AD \to E\}$ **n'implique pas** $AB \to CE$.
  Voir au tableau comment construire un contre-exemple à cette implication.

Plus sur les DF

- Dans l'exemple on a vérifié des implications en utilisant la définition d'implication déjà donnee.
- Existe-t-il une autre façon de faire ?
- Le système axiomatique dit d'Armstrong (voir TD) est correct et complet par rapport à la notion d'implication définie :
  Thm. F implique f ssi il existe une déduction de f à partir de F dans le système d'Armstrong.
- Mais, Problème : ce système ne fournit aucun algorithme pour tester si F implique f !

Fermeture d'un ensemble d'attributs

Comment calculer de façon mécanique si f est impliquée par un ensemble de dépéndances fonctionnelles F ?

- **Déf.** : Soit Z l'ensemble des attributs qui apparaissent dans F et soit $X \subseteq Z$.
  La *fermeture* de X par rapport à F, notée ${X^+}_F$, est :

$$max\{Y \subseteq Z \mid F \text{ implique } X \rightarrow Y\}$$

- **Théorème** : F implique $X \rightarrow Y$ ssi $Y \subseteq {X^+}_F$.
- Donc, pour savoir si F implique $X \rightarrow Y$ il suffit de savoir calculer ${X^+}_F$!

Fermeture d'un ensemble d'attributs

ALGORITHME DE CALCUL DE $X^+{}_F$:

Entrée : F, X, Z, où Z est l'ensemble de tous les attributs dans F
Sortie : $X^+{}_F$ comme valeur de la variable *res*.

1) *res*:=X;
2) **répéter** :
   pour chaque $Y \to V \in F$, **si** $Y \subseteq res$ **alors** $res := res \cup V$
jusqu'à quand *res* n'a pas changé ou $res = Z$;
3) **renvoyer** *res*

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

Question 1 : Complexité de l'algorithme, dans le pire des cas ?

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 F) ?

Exemples

1. On fait tourner l'algorithme pour

$$U_1 = \{A, B, C, D\}, \qquad F_1 = \{A \to B,\ BC \to D\}$$

pour tester si F_1 implique $A \to D$, si si F_1 implique $AC \to D$ et si F_1 implique $B \to A$.

2. On fait tourner l'algorithme pour

$$U_2 = \{A, B, C, D, E\}, \qquad F_2 = \{AB \to C,\ B \to D,\ D \to B\}$$

pour calculer les clés d'une table $R(A, B, C, D, E)$ fixées par F_2.

Problèmes possibles pour un schéma d'une base

Relation FOURNISSEUR

Nom	*Adr*	*Produit*	*Prix*
n1	a1	i1	p1
n1	a1	i2	p2
n2	a2	i2	p3
n2	a2	i3	p4

$F = \{Nom \rightarrow Adr,\ Nom\ Produit \rightarrow Prix\}$

- 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'un fournisseur qui n'a pas (encore) de produit.
- Anomalies de suppression : si effacement de tous les articles de n1, perte de l'adresse de n1.

Problèmes possibles pour un schéma d'une base

Anomalies du schéma vu : lieés au comportement du schéma par rapport aux DF (ici : dépéndance partielle, seconde forme normale' violée).

(Voir la suite pour les déf. formelles des *formes normales*).

C'est pour éviter ce type d'anomalies que l'on essaie de concevoir, dès le départ, des schémas de base qui soient satisfaisants (conception selon la méthode EA).

Ou, alors, on essaye de réparer des schémas déjà existants mais mauvais.

Décompositions d'un schéma : propriétés SPI et SPD

A nouveau l'exemple de mauvais schéma, donnant lieu à des redondances et des anomalies, déjà consideré :

Relation FOURNISSEUR

Nom	*Adr*	*Produit*	*Prix*
n1	a1	i1	p1
n1	a1	i2	p2
n2	a2	i2	p3
n2	a2	i3	p4

$F = \{Nom \rightarrow Adr,\ Nom\ Produit \rightarrow Prix\}$

Il faudrais le remplacer par un autre schéma. Comment ?

Décompositions de schémas

Une solution possible : décomposer le schéma de relation en 2 ou plusieurs sous-schémas. Par ex. décomposer :

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

en :

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

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

Ceci est un exemple de la façon générale de décomposer une table de schéma S en n tables de schémas $S_1, \cdots, S_n$ où $(S_1 \cup \cdots \cup S_n) = S$:
on projette le contenu de la table decomposée sur les sous-schémas.

Décompositions de schémas

- **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 :
  $\pi_{Nom,Adr}(FOURNISSEUR) \bowtie$
  $\pi_{Nom,Produit,Prix}(FOURNISSEUR) = FOURNISSEUR$? OUI.
- Toujours vrai, peu importe la décomposition ? 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 !
Cette décomposition fait perdre l'information NEGATIVE :
"$\langle a1, b1, c2\rangle$ **n'est pas** une ligne de R".

Décompositions SPI

- Soit F un ensemble de DF. Une décomposition d'un schéma S d'une table R en $S_1, \cdots, S_n$ (où $(S_1 \cup \cdots \cup S_n) = S$) 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-exemple précédant).

Rôle de F ??

$$\Rightarrow$$

Décompositions SPI

Pour le schéma de table de l'example précédant, on n'aurait pas pu avoir perte d'information négative si $B \rightarrow C$ ou $B \rightarrow A$:

on obtient $\langle a1, b1, c2 \rangle \in (\pi_{A,B}(R) \bowtie \pi_{B,C}(R))$ mais $\langle a1, b1, c2 \rangle \notin R$ parce que R ne satisfait pas $B \rightarrow C$ et ne satisfait pas $B \rightarrow A$!

Soit $S = \{A, B, C\}$ un schéma de table et soit F un ensemble de DF sur S.
Si F implique $B \rightarrow C$ ou $B \rightarrow A$, alors la décomposition de S en $\{A, B\}$ et $\{B, C\}$ est forcement SPI par rapport à F.

Remarque :
$B \rightarrow A = (\{A, B\} \cap \{B, C\} \rightarrow (\{A, B\} \setminus \{B, C\}$
$B \rightarrow C = (\{A, B\} \cap \{B, C\} \rightarrow (\{B, C\} \setminus \{A, B\}$

Décompositions SPI

Théorème : Une décomposition de S en 2 sous-schémas S_1 et S_2 est SPI par rapport à un ensemble de dépendances fonctionnelles F ssi F implique

$$(S_1 \cap S_2) \rightarrow (S_1 \setminus S_2)$$

ou

$$(S_1 \cap S_2) \rightarrow (S_2 \setminus S_1)$$

NB : ce théorème donne un algo pour tester si une décomposition en 2 est SPI par rapport à F.

Et si on veut décomposer en n tables où $n > 2$? Algo qui teste si une décomposition est SPI : chase. **A voir en TD**.

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

Décompositions : un problème avec le DF

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

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

INFO1	
R	C
r1	c1
r1	c2
r3	c3

INFO2	
C	V
c1	v1
c2	v2
c3	v1

d est SPI car F implique $((RC) \cap (CV)) \rightarrow (CV \setminus RC)$.

Mais que dévient $VR \rightarrow C$? Pas applicable.
Insertion de $\langle r1, c3\rangle$ dans INFO1 : OK par rapport aux DF de INFO1, mais avec la conséquence :
$\langle r1, c3, v1\rangle$ **et** $\langle r1, c1, v1\rangle \in$ INFO1 $\bowtie$ INFO2. Violation de $VR \rightarrow C$!

Décompositions : un problème avec le DF

La décomposition de l'exemple précédent ne fait pas perdre d'information mais elle fait perdre des dependances fonctionnelles.

Pour savoir si une insertion préserve la contrainte $VR \rightarrow C$ il faudrait réfaire la jointure INFO1 $\bowtie$ INFO2 !
On aimerait travailler sans besoin de faire des $\bowtie$ chaque fois qu'on insère !

Formalisons : $\Rightarrow$

Décompositions SPD

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

Déf. La projection d'un ensemble de dép. F su'un ensemble d'attributs E est l'ensemble d'éléments de F^+ (pas seulement de F !) comportant seulement des attributs de E. Notation : F_E.

Déf. Une décomposition d d'un schéma de relation S en $S_1 \cdots S_n$ est dite 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.
(Le pb. crucial est : risque-t-on de perdre des dépendances impliquees par F, donc des éléments de F^+, en décomposant ?)

Décompositions SPD

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 $A \to C$ et $C \to A \notin F$!.

Attention : Oublier que $A \to C$ et $C \to A$ sont dans F_{S_1} est un exemple typique d'erreur !

Décompositions SPD

Exemple de décomposition : 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 $F^+ \subseteq (F_{S_1} \cup F_{S_2})^+$ (tout ce qui est impliqué par F le sera aussi par $F_{S_1} \cup F_{S_2}$).

OUI, SPD !

Décompositions SPD

Un algorithme qui teste si une décomposition est SPD.

Voir en TD.

Décompositions SPD

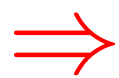
- Critère 2 qu'une décomposition doit respecter : être SPD par rapport aux depéndances fonctionnelles.
- Une déc. d peut être SPI sans être SPD : exemple déjà vu.
- Une déc. d peut être SPD sans être SPI. Etudier la déc. de :

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

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

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**.



Algorithme qui décompose SPI et SPD

Préliminaires à la déf. de l'algo qui engendre une décomposition SPI **et** SPD par rapport à un ensemble de dépendances fonctionnelles F

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

DEFINITIONS

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

Algorithme qui décompose SPI et SPD

Exemple.

Soit $F=$

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

1. Les dép. de F sont réduites à droite, mais ne sont pas réduites à gauche. On a : $AB \to C \in F$, mais F implique $B \to C$. (Calculer $\{B\}_F^+$!) L'ensemble $\{B\}$ est strictement inclus dans $\{A, B\}$.
2. F est redondant, car $F \setminus \{AB \to C\}$ implique $AB \to C$. (Calculer $\{AB\}^+_{F\setminus\{AB\to C\}}$!)

Algorithme qui décompose SPI et SPD

- **Définition**. Soit F un ensemble de DF.
  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**

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

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

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

  G est une couverture minimale de F (développement au tableau).
- Calcul mécanique d'une couverture minimale pour un ensemble de DF quelconque ?

Algorithme qui décompose SPI et SPD

Algorithme de Calcul d'une Couverture Minimale

Entrée : Un ensemble F de dépendences fonctionnelles.

Sortie : Une couverture minimale G de F.

1. **Initialisation** : $G{:=}F$;
2. **Faire**, dans l'ordre :
   - 2.1 (réduction à droite)
     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;
   - 2.2 (réduction à gauche)
     Remplacer toute $X \to Y \in G$ par $X' \to Y$ où X' est un sous-ensemble minimal de X tq G implique $X' \to Y$.
     G := l'ensemble de dép. ainsi obtenu;
   - 2.3 (élimination des redondances)
     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

Algorithme qui décompose SPI et SPD

Dans l'algorithme de calcul de la couverture minimale de F :

- Dans l'étape de réduction à gauche, comment trouver un sous-ensemble minimal X' de X tq G implique $X' \rightarrow Y$?
  On peut calculer ${E^+}_G$ pour tout sous-ensemble propre E de X, en partant des sous-ensembles les + petits.
- Dans l'étape d'élimination des redondances comment tester s'il existe $X \rightarrow A \in G$ tq $G \setminus \{X \rightarrow A\}$ implique $X \rightarrow A$?
  On peut calculer ${X^+}_{G \setminus \{X \rightarrow A\}}$ pour toute $X \rightarrow A \in G$.

Algorithme qui décompose SPI et SPD

Attention ! Dans l'algorithme de calcul de la couverture minimale de F, il faut faire les étapes de reductions gauche et droite avant l'étape d'élimination des redondances !

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 de faire

1. On fait l'étape d'élimination des redondances tout de suite : rien est redondant !
2. Etape de réduction à droite : rien à faire.
3. Etape de réduction à gauche : on 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

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

Algorithme qui décompose SPI et SPD

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 de réduction à droite : rien à faire
2. Etape de réduction à gauche : on obtient
   $F_1' = \{AC \to E, E \to D, A \to B, AC \to D\}$
3. Etape d'élimination des redondances : on élimine $AC \to D$.
4. Fini, avec le bon ensemble $F3$ comme résultat !

On est prêts à donner l'algoritme pour le calcul d'une décomposition qui soit, au même temps, 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 (calculées 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 $XA_1 \ldots A_k$ afin d'obtenir la valeur finale de d.

N.B Si plusieurs choix de couvertures minimales et de 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.

Formes Normales : Motivations

Soit $S = ABC$ et $F = \{A \rightarrow B, BC \rightarrow A\}$. Unique clé : $\{B, C\}$ (autre écriture admise : BC).

Soit la relation r satisfaisant toutes les dépendances fonctionnelles de F :

A	B	C
a	b	c
a	b	c'

Pas de redondance sur les valeurs de BC : il n'y a pas deux couples ayant les mêmes valeurs sur BC, car BC est une clé. Mais redondance sur AB.

Pourquoi la redondance sur AB ? A cause de la dépendance $A \rightarrow B$, où A n'est pas une clé.

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

Formes Normales : Terminologie et Convention

Terminologie :

- Une dépendance $X \to Y \in F$ est dite *banale* ssi $Y \subseteq X$.
  Exemples de dépéndances banales : $A \to A$, $AB \to B$, $ABC \to BC$.
- Un attribut A est dit premier (par rapport à F) ssi
  il existe au moins une clé C tq $A \in C$.
- Un ensemble E_1 est un sous-ensemble propre de E_2 si $E_1 \subset E_2$ mais $E_1 \neq E_2$.

Convention :

Dans la suite, on suppose que toute dépendance fonctionnelle est réduite à droite, c.à.d. sa forme est $X \to A$ où :

- X est un ensemble quelconque d'attributs
- A est un unique attribut.

Pourquoi on a le droit de le faire ?

Formes Normales : BCNF

- **Déf.** Soit S un schéma de relation et F un en ensemble de dépendances sur S.
  S est en Forme Normale de Boyce Codd (BCNF) par rapport à F ssi quelque soit la dépendance $X \to A$ impliquée par F, où A est un attribut, soit $X \to A$ est banale, soit X est une super-clé (par rapport à F).
  Intuitivement : les attributs dépendent seulement des clés.
  Dit autrement :
  S n'est pas en BCNF par rapport à F quand il existe au moins une dép. $f = X \to A$ impliquée par F telle que: f n'est pas banale et X n'est pas une super-clé (par rapport à F).
- **Exemple précédent** : $S = ABC$ et $F = \{A \to B, BC \to A\}$.
  Ici, S n'est pas BCNF ! En fait :
  $A \to B \in F$, n'est pas banale mais A n'est pas une super-clé.

Formes Normales : difficultés de la BCNF

On dit qu'un schéma de base est BCNF ssi tous ses schémas de relation sont en BCNF.

Schéma de base en en BCNF : souhaitable, car on n'aura jamais de redondances de données.
Toutefois : parfois trop fort ! Par ex., $\not\exists$ déc. pour l'exemple que l'on vient de voir qui soit, au même temps, SPI, SPD et BCNF !

On peut assouplir les contraintes imposées par la BCNF. On aura une hiérarchie de formes normales, la BCNF étant la plus contraignante.

Dans la suite : S=schéma de relation, F ensemble de dépendances fonctionnelles sur S, qu'on suppose toujours réduit à droite.

FN1

Déf. Un schéma de relations S est en première forme normale (FN1) par rapport à F) ssi

1. Tout attribut est atomique;
2. Tout attribut "a une valeur constante dans le temps, au sens que seulement une mise à jour explicite pourra le changer".

Exemples

Pas en FN1 :

Personne1(IdPersonne, ville, rue, numCiv, liste_num_tél)

Personne2(Idpersonne, ville, rue, numCiv, age)

En FN1 :

Personne1bis(IdPersonne, ville, rue, numCiv, num_tél)

Personne2bis(Idpersonne, ville, rue, numCiv, date_naiss)

(et on calcule l'age avec une requête).

Un schéma de base $\mathcal{S}$ est en FN1 si tous les schémas des relations sont en FN1.

FN1

FN1 : primordiale, pour le modèle relationnel !

Dans la suite, on supposera toujours que le schéma de la base est, quand même, en FN1.

FN2

Déf. :
Un schéma de relation S est en seconde forme normale (FN2) par rapport à F ssi S est en FN1 et toute dépendance $X \rightarrow A$ impliquée par F est t.q. ou bien X n'est est pas un sous-ensemble propre d'une clé C ou bien A est un attribut premier.
Dit autrement :
S est en seconde forme normale (FN2) par rapport à F ssi S est en FN1 et F n'implique pas de dépendances $X \rightarrow A$ où X est un sous-ensemble propre d'une clé C et A est un attribut qui n'est pas premier.
Dit encore autrement :
S n'est pas en FN2 par rapport à F ssi ou bien S en'est pas en FN1 ou bien F implique au moins une dépendance $X \rightarrow A$ où X est un sous-ensemble propre d'une clé C et A est un attribut qui n'est pas premier.
Un schéma de base $\mathcal{S}$ est en FN2 si tous ses schémas des relations sont en FN2.

FN2, Exemple

NON-FN2

Schéma de table S : $InfoEmp(EmpNom, Fonction, AdrEmp)$.

Puisque on suppose qu'un employé peut avoir plusieurs fonctions et qu'une fonction peut tre partagée parmi plusieurs employés, la clé sera $\{EmpNom, Fonction\}$
Mais on aura aussi, dans F : $EmpNom \rightarrow AdrEmp$, e cette dépendance viole la FN2.

Ici, $\{EmpNom\}$ es un sous-ensemble propre de la clé $\Rightarrow$ on risque d'avoir beaucoup de redondances sur AdrEmp.

NB. : une violation de FN2 signifie l'existence de dépendances partielles au sens de la méthode EA.

FN3

Déf. Un schéma de relation S est en troisième forme normale (FN3) par rapport à F ssi :
$\forall\ X \to A$ impliquée par F, où A est un attribut, on a :

- ou bien $X \to A$ est banale
- ou bien X est une super-clé (par rapport à F),
- ou bien A est premier.

Dit autrement :
S n'est pas FN3 par rapport à F ssi il existe au moins une dépendance $X \to A$ impliquée par F telle que :
elle n'est pas banale, X n'est pas une super-clé et A n'est pas premier.

Un schéma de base $\mathcal{S}$ est en FN3 si tous ses schémas des relations sont en FN3.

FN3, Exemple

Non-FN3

$S = Etudiant(IdEt, NomEt, NomDept, TelDpt)$
$F = \{IdEt \rightarrow NomEt\ NomDept\ TelDept,\ NomDept \rightarrow TelDept\}$.
Clé : $IdEt$.
La dépendance $NomDept \rightarrow TelDep$ n'est pas banale,$\{NomDept\}$ n'est pas une super-clé, et $telDep$t n'est pas premier. Elle viole la FN3.

Remarques

- Ceci revient à dire que l'on a une dépendance dite transitive (terminologie EA) entre $IdDept$ et $TelDept$:
  Puisque F implique $IdEt \rightarrow NomDept$ et
  $NomDept \rightarrow TelDept$, alors $IdEt \rightarrow TelDept$ peut être obtenue par trasitivité.
- On ne peut pas associer une valeur de $NomDept$ avec une valeur de $IdEt$ sans connaître la valeur de $TelDept$. Risque : anomalies d'insertion (ou de mise à jour).

FN3, Exemple : Suite

En revanche, si on décompose
$S = Etudiant(IdEt, NomEt, NomDept, TelDpt)$
(où
$F = \{IdEt \rightarrow NomEt\ NomDept\ TelDept,\ NomDept \rightarrow TelDept\}$)
en :
$S_1 = Etudiant(IdEt, NomEt, NomDept)$
avec $F_{S_1} = \{IdEt \rightarrow NomEt\ NomDept\}$
et
$S_2 = Dept(NomDept, TelDept)$
avec $F_{S_2} = \{Dept \rightarrow TelDept\}$

on obtient deux schémas de relation qui sont en FN3.

Forme Normales, Remarques

- Si S est en FN3 par rapport à F alors S est aussi en FN2.
  Pourquoi ?
  Mais l'implication réciproque n'est pas vraie. Voir l'exemple suivant, qui est en FN2 mais pas en FN3 :
  $S = R(A, B, C, D)$, $F = \{AB \rightarrow C, C \rightarrow D\}$.
- Si S est en BNCF par rapport à F, alors S est aussi en FN3.
  Pourquoi ?
  Mais l'implication réciproque n'est pas vraie :
  le tout premier exemple de non-BCNF était en FN3 !

Hiérarchie de Formes Normales

HIERARCHIE DE FORMES NORMALES : voir le tableau

Algorithme de décomposition SPI, SPD et FN3

$\exists$ un algorithme pour décomposer un schéma de relation de façon à assurer SPI, SPD **et** FN3 ?

Oui, l'algo de décomposition SPI, SPD déjà donné produit aussi FN3 !

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

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

au tableau.

Algorithme de décomposition SPI, SPD et FN3, Exemple Long

Exemple plus complexe d'application de l'algo de décomposition SPI, SPD, FN3

Application à :

$S = ABCDEFG$

$$F = \{AB \rightarrow C, B \rightarrow A, CB \rightarrow CD, AB \rightarrow D, C \rightarrow B, A \rightarrow B\}$$

$\Rightarrow$

Algorithme de décomposition SPI, SPD et FN3. Exemple Long, Etape 1

Exemple plus complexe d'application de l'algo de décomposition SPI, SPD, FN3

Etape 1 :Calcul d'une couverture minimale F' de :

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

Réduction à droite

F est réécrit en $F_1 = \{AB \to C, B \to A, CB \to C, CB \to D,$
$AB \to D, C \to B, A \to B\}$

Suite : $\Rightarrow$

Algorithme de décomposition SPI, SPD et FN3. Exemple Long, Etape 1

Etape 1 :Calcul d'une couverture minimale, suite

On a obtenu

$F_1 = \{AB \to C, B \to A, CB \to C, CB \to D,$
$AB \to D, C \to B, A \to B\}$

Réduction à gauche

${A^+}_{F_1} = ABCD$, $\{C\} \subseteq {A^+}_{F_1}$ et $\{D\} \subseteq {A^+}_{F_1}$. Donc $AB \to C$ est remplacée par $A \to C$ et $AB \to D$ est remplacée par $A \to D$.

De même, ${C^+}_{F_1} = ABCD$, $\{C\} \subseteq {C^+}_{F_1}$ et $\{D\} \subseteq {C^+}_{F_1}$. Donc $CB \to C$ est remplacée par $C \to C$ et $CB \to D$ est remplacée par $C \to D$.

Le résultat de la réduction à gauche est :

$F_2 : \{A \to C, B \to A, C \to C, C \to D, A \to D, C \to B, A \to B\}$

Suite : $\Rightarrow$

Algorithme de décomposition SPI, SPD et FN3. Exemple Long, Etape 1

Etape 1 :Calcul d'une couverture minimale, suite. On a obtenu
$F_2 : \{A \to C, B \to A, C \to C, C \to D, A \to D, C \to B, A \to B\}$

Elimination des redondances

$A^+_{F_2 \setminus \{A \to C\}} = ABD \Rightarrow$ garder $A \to C$.
$B^+_{F_2 \setminus \{B \to A\}} = B \Rightarrow$ garder $B \to A$.
$C \to C$ est banale, donc impliquée par **tout** ensemble de DF $\Rightarrow$ la supprimer ! On obtient :
$F_3 : \{A \to C, B \to A, C \to D, A \to D, C \to B, A \to B\}$
$C^+_{F_3 \setminus \{C \to D\}} = ABCD \Rightarrow$ supprimer $C \to D$.
On obtient : $F_4 : \{A \to C, B \to A, A \to D, C \to B, A \to B\}$
$A^+_{F_4 \setminus \{A \to D\}} = ABC \Rightarrow$ garder $A \to D$.
$C^+_{F_4 \setminus \{C \to B\}} = C \Rightarrow$ garder $C \to B$.
$A^+_{F_4 \setminus \{A \to B\}} = ABCDD \Rightarrow$ supprimer $A \to B$.
$\Rightarrow$ Fin de l'étape 1, avec :
$F' = \{A \to C, B \to A, A \to D, C \to B\}$

Algorithme de décomposition SPI, SPD et FN3. Exemple Long, Etape 2

Etape 2 : Première décomposition, suivant les DF.

On a obtenu $F' = \{A \to C, B \to A, A \to D, C \to B\}$
qu'on va écrire :

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

pour plus de lisibilité.

$$d := \{AC, AD, BA, CD\}$$

Algorithme de décomposition SPI, SPD et FN3. Exemple Long, Etape 3

Etape 3 : Ajout d'une clé ?.

Rappel :

$$d = \{AC, AD, BA, CD\}$$

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

3 clés : $AEFG$, $BEFG$ et $CEFG$. Pourquoi ?

$$\Downarrow$$

$$d := \{AC, AD, BA, CD, CEFG\}$$

(on aurait pu choisir une autre clé)

Algorithme de décomposition SPI, SPD et FN3. Exemple Long, Etape 4

Etape 4 : Gestion des inclusions entre sous-schémas.
Rappel :

$$d = \{AC, AD, BA, CD, CEFG\}$$

Aucun des 5 sous-schémas est inclus dans un autre.
Rien à faire, ici.

Algorithme de décomposition SPI, SPD et FN3. Exemple Long, Etape 5

Etape 5 : Fusion éventuelle des sous-schémas.

Rappel :

$$d = \{AC, AD, BA, CD, CEFG\}$$

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

Nouvelle affectation :

$$d := \{ACD, BA, CD, CEFG\}$$

(voir la construction des sous-schémas AC, AD à l'étape 2)

Fin de l'étape 5 et de l'algo. Le résultat de l'algo est la décomposition

$$d = \{ACD, BA, CD, CEFG\}$$

Révision, un ancien examen

Exercice 1, 4 points

Soit $\mathcal{B}$ une base de données, concernant la vente de plantes par des producteurs, ayant le schéma suivant :
Producteur(IdProd, Nom, Prénom, AdresseMail)
Client(IdClient, Nom, Prénom, AdresseMail)
Produit(RefProduit, IdProd, Type, NomProduit, Prix)
Commande(NumCommande, IdClient,Refproduit, Quantité, Date).
Dans la table Produit, Type indique la sorte des plantes vendues; par exemple, *ArbresFruitiers* appartient au domaine de l'attribut Type, tandis que *Cérisier* appartient au domaine de l'attribut NomProduit.
Pour chaque schéma de table, la clé est soulignée.

1. Exprimer en AR la requête :
   Quels clients ont commandé au moins deux types de plantes différentes le 20/11/2014 ? Donner leur idéntifiants.
2. Exprimer en CR et SQL la requête :
   Quels producteurs produisent tout type de produit ? Donner leur idéntifiants.

Révision, un ancien examen

Exercice 2, 5 points Soit $R(A, B, C, D, E, F, G)$ un schéma de relation et soit F l'ensemble de dépendances fonctionnelles suivant :
$\{AB \rightarrow C,\ AB \rightarrow D, D \rightarrow E,\ D \rightarrow F,\ DF \rightarrow G,\ AB \rightarrow G\}$

1. Donner l'ensemble des clés.
2. Soit d_1 la décomposition du schéma dans les 3 schémas ACG, AD et $ABEF$.
   - 2.1 Utilisez un algoritme approprié pour déterminer si d_1 est SPI ou pas par rapport à F. Si votre réponse est $\langle\langle$ Pas SPI $\rangle\rangle$ donnez un contre-exemple. Un contre exemple sera donné en fournissant des valeurs appropriées pour la table $R(A, B, C, D, E, F, G)$ que l'on veut décomposer.
   - 2.2 Déterminez si d_1 est SPD par rapport à F ou pas. Pour le faire, on calculera d'abord les projections de F sur les sous-schémas constituant d_1 (dans ce cas, ce seront F_{ACG}, F_{AD} et F_{BEF}). Si cette décomposition n'st pas SPD, indiquez une dépendance perdue qui justifie votre réponse.
3. Mêmes questions pour la décomposition d_2 de $R(A, B, C, D, E, F, G)$ en 2 schémas, qui sont $ABCDG$ et $ABEF$.

Révision, un ancien examen

Exercice 3, 3 points
Soit F l'ensemble de dépendances fonctionnelles de l'exercice 2, c'est à dire :

$$\{AB \rightarrow C,\ AB \rightarrow D, D \rightarrow E,\ D \rightarrow F,\ DF \rightarrow G,\ AB \rightarrow G\}$$

Calculer une couverture minimale de F, en précisant le résultat de chacune des 3 étapes.

Révision, un ancien examen

Exercice 4, 6 points

On suppose que tout schéma de relation est en FN1. Soit $R(A, B, C, D, E, F, G)$ un schéma de relation.

1. Soit F l'ensemble de dépendances fonctionnelles des deux exercices précédents. Déterminer quelles formes normales sont valables, en justifiant la réponse donnée. Si la FN3 n'est pas valable, donner une décomposition de $R(A, B, C, D, E, F, G)$ qui soit SPI, SPD et FN3. On utilisera l'algorithme du cours et on précisera le résultat de chacune de ses étapes.

2. Mêmes questions, toujours avec le schéma de relation $R(A, B, C, D, E, F, G)$ mais avec l'ensemble de dépendances :

$$F_1 = \{A \to B,\ A \to C, B \to E,\ E \to B,\ E \to F,\ B \to A\}$$

Révision, un ancien examen

Exercice 5, 3 points

1. Démontrer que tout schéma de relation qui est en FN3 (par rapport à un ensemble de dépendances fonctionnelles F donné) est aussi en FN2, en utilisant les définitions de FN2 et FN3 du cours.
2. Démontrer que l'implication reciproque n'est pas vraie, en donnant un exemple de schéma de table S et d'ensemble de dépendances fonctionnelles F tels que S est en FN2 par rapport à F mais n'est pas en FN3.