Bases de Données : Normalisation (introduction)

Stéphane Devismes

Université Grenoble Alpes

26 août 2020

Plan

Définition

La normalisation d'une base de données a pour but

1. d'évaluer la **qualité** de son schéma (les relations et leurs attributs) et
2. de **modifier** ce schéma (si besoin) afin d'améliorer la **qualité** de la base.

Le but d'une base de donnée (relationnelle) est d'éviter la redondance et les problèmes sous-jacents de mise à jour ou de cohérence.

Normalisation *vs* Conception

Si on crée le schéma d'une base de données en suivant la démarche vue précédemment dans ce cours (modèle conceptuel, modèle logique, modèle physique), alors en général la base obtenue est « bien normalisée ».

Mais **si on part d'un schéma préexistant** « de mauvaise qualité », alors la normalisation permet de **restructurer la base**.

Démarche

Nous allons considérer **une seule relation** R.
(Bien sûr, la normalisation doit être appliquée à toutes les relations !)

On suppose que l'on connaît :

1. **Tous les attributs** de R.
   Le type des attributs n'importe pas.
   On suppose qu'il n'y a pas de valeurs absentes.
   (On ne se préoccupe que de la structure, pas des données.)
2. **Certaines dépendances (fonctionnelles) entre les attributs.**
   Les dépendances fonctionnelles sont des contraintes qui généralisent la contrainte de clé (ou d'identifiant).

   Une dépendance $A \rightarrow B$ signifie que **toute valeur de** A dans R **définit de manière unique les valeurs de** B dans R.

On suppose que les dépendances données (une base) « engendrent » toutes les dépendances fonctionnelles entre attributs de la relation R.

Dépendances et clés

Toutes les clés de R peuvent être **obtenues à partir des dépendances données**. (nous étudierons un algorithme)

Les clés sont toutes au même niveau : il n'y a pas de différence entre clé primaire et clé secondaire.

On dit parfois que ce sont les clés candidates de la relation R.

Notations

Étant donnée une relation R et un ensemble A d'attributs de R, on note souvent $A = a_1 ... a_n$ au lieu de $A = \{a_1, ..., a_n\}$.

On note $R[A]$ ou $R[a_1, ..., a_n]$ ou $R[a_1 ... a_n]$ la relation projection de R sur l'ensemble d'attributs A.

Dans tout le chapitre on note S l'ensemble de tous les attributs de R.

Définition

Soient une relation R et deux ensembles d'attributs A et B de R.

On dit que A détermine (fonctionnellement) B, ou que B dépend (fonctionnellement) de A, et on note $A \rightarrow B$ lorsque :

Dès que deux lignes de R ont la même valeur en chaque attribut de A elles ont aussi la même valeur en chaque attribut de B.

Une autre façon de le dire : $a_1 \dots a_n \rightarrow b_1 \dots b_p$ signifie que

Si deux lignes ont les mêmes valeurs pour a_1 **et** . . .**et** a_n, alors elles ont aussi les mêmes valeurs pour b_1 **et** . . .**et** b_p.

Exemple ≪ Livres ≫ (1/2)

Soit $R(l, o, s, b)$: penser à Livres (l), Œuvres (o), Salles (s), Bâtiments (b).

Supposons que les attributs vérifient les dépendances $\{l \to os,\ s \to b\}$.

Exemple ≪ Livres ≫ (2/2)

Soit $R(l, o, s, b)$ comme précédemment, mais avec les dépendances $\{l \to os,\ s \to b,\ o \to b\}$.

Alors :

- la nouvelle dépendance $o \to b$ signifie que **tous les exemplaires d'une même œuvre** sont dans **le même** bâtiment, mais peut-être dans des salles différentes de ce bâtiment.

Exemple ≪ Clients ≫

Soit $R(c, n, p, a)$: penser à Client (identifié par un numéro de client c), Nom (n), Prénom (p), Adresse (a) avec les dépendances $\{c \to np, np \to c, np \to a\}$.

- $c \to np$: le numéro d'un client **détermine** son nom et son prénom.

  On peut remplacer $c \to np$ par $c \to n$ et $c \to p$.
- $np \to c$ signifie que **deux clients ne peuvent pas avoir à la fois le même nom et le même prénom**.

  On NE PEUT PAS remplacer $np \to c$ **par** $n \to c$ **et** $p \to c$: si deux clients ne peuvent pas avoir à la fois le même nom et le même prénom, cela n'interdit pas qu'ils aient soit le même nom soit le même prénom.
- $np \to a$ signifie qu'à partir du nom et du prénom d'un client **on connaît** son adresse.

  On NE PEUT PAS remplacer $np \to a$ **par** $n \to a$ **et** $p \to a$: le nom ne suffit pas à déterminer l'adresse, le prénom non plus.

Exemple ≪ Professeurs ≫ (1/2)

Soit $R(p, m, g, s)$ (pour Professeur p, Matière m, Groupe g, Salle s) avec l'ensemble de dépendances fonctionnelles $\{gp \to ms,\ gm \to ps\}$.

Exemple « Professeurs » (2/2)

Soit $R(p,m,g,s)$ (pour Professeur p, Matière m, Groupe g, Salle s) avec l'ensemble de dépendances $\{gp \to ms, gm \to ps\}$.

- $g \not\to s$:
  un groupe d'étudiants n'est pas toujours dans la même salle.

  On peut le vérifier en donnant un **contre-exemple**, c'est-à-dire une valeur pour la relation $R(p,m,g,s)$ qui vérifient $gp \to ms$ et $gm \to ps$ mais pas $g \to s$:

p	m	g	s
p1	m1	g1	s1
p2	m2	g1	s2

  avec $p1 \neq p2$, $m1 \neq m2$ et $s1 \neq s2$.

Pourquoi dépendance « fonctionnelle » ?

Soient $A = a_1 \dots a_n$ et $B = b_1 \dots b_p$ deux ensembles d'attributs de R.

Alors dans R **à chaque valeur** $\langle x_1, \dots, x_n \rangle$ **de** $a_1 \dots a_n$ **correspond** en général (au sens « figure sur la même ligne que ») **plusieurs valeurs** $\langle y_1, \dots, y_p \rangle$ de $b_1 \dots b_p$.

La définition dit que $A \to B$ si et seulement si **à chaque valeur** $\langle x_1, \dots, x_n \rangle$ de $a_1 \dots a_n$ correspond exactement une valeur $\langle y_1, \dots, y_p \rangle$ de $b_1 \dots b_p$.

Alors, en posant $f(\langle x_1, \dots, x_n \rangle) = \langle y_1, \dots, y_p \rangle$ on définit une fonction $f : R[A] \to R[B]$.

Clés et dépendances fonctionnelles

On retrouve la notion de clé relative à un ensemble de dépendances fonctionnelles $\mathcal{D}$.

Une superclé K est un ensemble d'attributs tels que $K \to S$, où S est l'ensemble des attributs de la relation R.

Une clé K est une **superclé minimale** : $K \to S$ et $A \not\to S$ pour tout $A \subsetneq K$.

Attention. « Superclé minimale » ne veut pas dire « superclé de longueur minimum », on utilise la minimalité au sens des ensembles (pour l'inclusion), pas au sens des cardinaux.

Dans l'exemple « Clients », on a $R(c,n,p,a)$ avec $\{c \to np, np \to c, np \to a\}$. R a deux clés possibles : c (qui est de taille minimum) et np (qui est minimale mais pas de taille minimum).

Superclés : propriétés

Soit S l'ensemble des attributs de la relation R.

- S est toujours une superclé de R.
- Si K est une superclé de R et A un ensemble d'attributs de R alors $K \to A$.
- K est une superclé de R si et seulement si K contient une clé de R.

Propriétés fondamentales

La relation de dépendance fonctionnelle « $\rightarrow$ » vérifie les propriétés suivantes :

- Réflexivité (« augmentée »). Si $B \subseteq A$ alors $A \rightarrow B$.
- Transitivité. Si $A \rightarrow B$ et $B \rightarrow C$ alors $A \rightarrow C$.
- Augmentation. Si $A \rightarrow B$ alors $A \cup C \rightarrow B \cup C$.

$$(\mathrm{refl})\ \frac{B \subseteq A}{A \rightarrow B} \quad (\mathrm{trans})\ \frac{A \rightarrow B \quad B \rightarrow C}{A \rightarrow C} \quad (\mathrm{augm})\ \frac{A \rightarrow B}{A \cup C \rightarrow B \cup C}$$

Propriétés dérivées (1/2)

On peut en déduire les propriétés suivantes :

- Réflexivité stricte. $A \rightarrow A$.
- Union. Si $A \rightarrow B$ et $A \rightarrow C$ alors $A \rightarrow B \cup C$.
- Décomposition. Si $A \rightarrow B \cup C$ alors $A \rightarrow B$ et $A \rightarrow C$.
- Transitivité augmentée. Si $A \rightarrow B$ et $B \cup C \rightarrow D$ alors $A \cup C \rightarrow D$.
- Union augmentée. Si $A \rightarrow B$ et $C \rightarrow D$ alors $A \cup C \rightarrow B \cup D$.
- Affaiblissement. Si $A \rightarrow B$ et $A \subseteq A'$ et $B' \subseteq A \cup B$ alors $A' \rightarrow B'$.

Propriétés dérivées (2/2)

$$(\mathrm{refl}_1)\ \frac{}{A \rightarrow A} \quad (\mathrm{union})\ \frac{A \rightarrow B \quad A \rightarrow C}{A \rightarrow B \cup C}$$

$$(\mathrm{dec}_1)\ \frac{A \rightarrow B \cup C}{A \rightarrow B} \quad (\mathrm{dec}_2)\ \frac{A \rightarrow B \cup C}{A \rightarrow C}$$

$$(\mathrm{trans}_1)\ \frac{A \rightarrow B \quad B \cup C \rightarrow D}{A \cup C \rightarrow D} \quad (\mathrm{union}_1)\ \frac{A \rightarrow B \quad C \rightarrow D}{A \cup C \rightarrow B \cup D}$$

$$(\mathrm{aff})\ \frac{A \subseteq A' \quad A \rightarrow B \quad B' \subseteq A \cup B}{A' \rightarrow B'}$$

Propriétés dérivées : Réflexivité stricte (preuve)

$$(\mathrm{refl})\ \frac{A \subseteq A}{A \rightarrow A}$$

Propriétés dérivées : Union (preuve)

$$\dfrac{(\text{augm})\ \dfrac{A \to C}{A \to A \cup C} \qquad (\text{augm})\ \dfrac{A \to B}{A \cup C \to B \cup C}}{A \to B \cup C}\ (\text{trans})$$

Propriétés dérivées : Décomposition (preuve)

$$(\text{trans})\ \dfrac{A \to B \cup C \qquad (\text{refl})\ \dfrac{B \subseteq B \cup C}{B \cup C \to B}}{A \to B}$$

$$(\text{trans})\ \dfrac{A \to B \cup C \qquad (\text{refl})\ \dfrac{C \subseteq B \cup C}{B \cup C \to C}}{A \to C}$$

Propriétés dérivées : Transitivité augmentée (preuve)

$$\dfrac{(\text{augm})\ \dfrac{A \to B}{A \cup C \to B \cup C} \qquad B \cup C \to D}{A \cup C \to D}\ (\text{trans})$$

Propriétés dérivées : Union augmentée (preuve)

$$\dfrac{(\text{augm})\ \dfrac{A \to B}{A \cup C \to B \cup C} \qquad (\text{augm})\ \dfrac{C \to D}{B \cup C \to B \cup D}}{A \cup C \to B \cup D}\ (\text{trans})$$

Propriétés dérivées : Affaiblissement (preuve)

$$\dfrac{(\text{refl})\ \dfrac{A \subseteq A'}{A' \to A} \qquad (\text{augm})\ \dfrac{A \to B}{A \to A \cup B} \qquad (\text{refl})\ \dfrac{B' \subseteq A \cup B}{A \cup B \to B'}}{A' \to B'}\ (\text{trans})$$

Conséquence

Soit $\mathcal{D}$ un ensemble de dépendances fonctionnelles.

Une dépendance fonctionnelle $A \to B$ est conséquence de $\mathcal{D}$ si $A \to B$ peut être obtenue **à partir** des dépendances fonctionnelles de $\mathcal{D}$ en utilisant les trois propriétés **réflexivité, transitivité, augmentation**.

(On peut donc aussi utiliser toutes les propriétés qui découlent de ces trois propriétés.)

Exemple de conséquence

Soit $\mathcal{D} = \{ab \to c, bc \to d, d \to e\}$. Démontrons que $ab \to e$ est une conséquence de $\mathcal{D}$.

$$\dfrac{(\text{trans})\ \dfrac{(\text{augm})\ \dfrac{ab \to c}{ab \to bc} \qquad bc \to d}{ab \to d} \qquad d \to e}{ab \to e}\ (\text{trans})$$

Remarque sur les conséquences

Une preuve est un arbre dont chaque nœud est une prémisse et/ou une conclusion de règle et dont la racine est la conclusion de la preuve, dans l'exemple précédent $ab \to e$.

Dans une preuve à partir de $\mathcal{D}$, **les feuilles (non-racines)** de l'arbre **sont** :

- **soit des dépendances de $\mathcal{D}$,**
- **soit des inclusions $X \subseteq Y$.**

Hauteur d'un arbre de preuve

Nous avons besoin de la notion de hauteur pour effectuer des récurrences sur les arbres de preuves.

Soit T un arbre de preuve de $A \rightarrow B$. Nous noterons $T(x)$ le sous-arbre de T enraciné en le nœud x.

Nous définissons inductivement la hauteur $h(T)$ de T comme suit.

- **Si** T consiste uniquement en $A \rightarrow B$, **alors** $h(T) = 0$.
- **Sinon**, $A \rightarrow B$ est la racine de T : la conclusion $A \rightarrow B$ est obtenue à partir d'une premise P (réflexivité ou augmentation) ou deux prémises P' et P'' (transitivité).

  Dans le premier cas, si $A \rightarrow B$ est la conclusion d'une réflexivité alors $h(T) = 1$, sinon (en cas d'augmentation) $h(T) = 1 + h(T(P))$.

  Dans le second cas, $h(T) = 1 + \max\{h(T(P')), h(T(P''))\}$.

Exemple : L'arbre de preuve de l'exemple précédent est de hauteur 3.

Singleton

Une dépendance fonctionnelle $A \rightarrow B$ est singleton si elle est de la forme $a_1 \ldots a_n \rightarrow b$ avec $b \in S$ et $b \notin \{a_1, \ldots, a_n\}$, où S est l'ensemble des attributs de la relation R.

Etant donné un ensemble de dépendances fonctionnelles $\mathcal{D}$, l'ensemble de dépendances fonctionnelles singletons associé à $\mathcal{D}$ est obtenu en remplaçant

chaque dépendance fonctionnelle $a_1, \ldots, a_n \rightarrow b_1 \ldots b_p$ de $\mathcal{D}$

par

les dépendances fonctionnelles $a_1, \ldots, a_n \rightarrow b_i$ pour tout i entre 1 et p tel que $b_i \notin \{a_1, \ldots, a_n\}$.

Propriété

Soit $\mathcal{D}$ un ensemble de dépendances fonctionnelles et $\mathcal{D}'$ l'ensemble de dépendances fonctionnelles singletons associé à $\mathcal{D}$.

Alors $\mathcal{D}$ et $\mathcal{D}'$ sont équivalents, au sens suivant :

une dépendance fonctionnelle $A \rightarrow B$ est conséquence de $\mathcal{D}$ si et seulement si elle est conséquence de $\mathcal{D}'$.

Preuve : utiliser les règles d'union et de décomposition.

Clôture

Soit S l'ensemble des attributs de la relation R.
Soit $\mathcal{D}$ un ensemble de dépendances fonctionnelles.

Pour tout ensemble d'attributs $A \subseteq S$, la clôture (ou fermeture) A^+ de A (relativement à $\mathcal{D}$) est **le plus grand ensemble d'attributs tel que la dépendance fonctionnelle $A \rightarrow A^+$ est conséquence de $\mathcal{D}$.**

Propriété

- $A \subseteq A^+ \subseteq S$
- K est une **superclé** si et seulement si $K^+ = S$.
- K est une **clé** si et seulement si $K^+ = S$ et pour tout $A \subsetneq K$ on a $A^+ \neq S$.

Algorithme de clôture d'un ensemble d'attributs

Entrée. Un ensemble $\mathcal{D}$ de dépendances fonctionnelles singletons, sur un ensemble d'attributs S.
Un ensemble d'attributs A.

Sortie. La clôture A^+ de A relativement à $\mathcal{D}$.

Principe. L'ensemble « courant » d'attributs est initialisé à A, il est enrichi progressivement, à la fin il vaut A^+.

-I- Initialiser A^+ à A.
-B- Tant qu'il existe une dépendance $A' \to b$ dans $\mathcal{D}$
    avec $A' \subseteq A^+$ et $b \notin A^+$,
    ajouter b à A^+.
-R- Retourner A^+.

Exemple (1/5)

Soit $S = \{a, b, c, d, e, f\}$ un ensemble d'attributs.
Soit l'ensemble de dépendances fonctionelles :
$ab \to c,\ d \to e,\ bc \to ad,\ cf \to b$.
ou de façon équivalente l'ensemble $\mathcal{D}$ de dépendances fontionnelles singletons :
$(d1)\, ab \to c,\ (d2)\, d \to e,\ (d3)\, bc \to a,\ (d4)\, bc \to d,\ (d5)\, cf \to b$.

Quelle est la clôture de ab relativement à $\mathcal{D}$?

Exemple (2/5)

Soit $S = \{a, b, c, d, e, f\}$ un ensemble d'attribut.
Soit l'ensemble de dépendances fonctionelles :
$ab \to c,\ d \to e,\ bc \to ad,\ cf \to b$.
ou de façon équivalente l'ensemble $\mathcal{D}$ de dépendances fontionnelles singletons :
$(d1)\, ab \to c,\ (d2)\, d \to e,\ (d3)\, bc \to a,\ (d4)\, bc \to d,\ (d5)\, cf \to b$.

Quelle est la clôture de df relativement à $\mathcal{D}$?

-I- $A^+ = df$.
-B- On peut oublier $(d4)$, dont la partie droite est dans A.
-B- $(d2)$: $A^+ = def$.
-R- $df^+ = def$.

Exemple (3/5)

Soit $S = \{a, b, c, d, e, f\}$ un ensemble d'attribut.
Soit l'ensemble de dépendances fonctionelles :
$ab \to c,\ d \to e,\ bc \to ad,\ cf \to b$.
ou de façon équivalente l'ensemble $\mathcal{D}$ de dépendances fontionnelles singletons :
$(d1)\, ab \to c,\ (d2)\, d \to e,\ (d3)\, bc \to a,\ (d4)\, bc \to d,\ (d5)\, cf \to b$.

Quelle est la clôture de *acf* relativement à $\mathcal{D}$?

-I- $A^+ = acf$.
-B- On peut oublier $(d1)$ et $(d3)$, dont la partie droite est dans A.
-B- $(d5)$: $A^+ = abcf$.
-B- $(d4)$: $A^+ = abcdf$.
-B- $(d2)$: $A^+ = abcdef$.
-R- $acf^+ = abcdef$. Donc *acf* est une **superclé**.

Exemple (4/5) : trouver une clé

Soit $S = \{a, b, c, d, e, f\}$ un ensemble d'attribut.
Soit l'ensemble de dépendances fonctionelles :
$ab \to c,\ d \to e,\ bc \to ad,\ cf \to b$.
ou de façon équivalente l'ensemble $\mathcal{D}$ de dépendances fontionnelles singletons :
$(d1)\, ab \to c,\ (d2)\, d \to e,\ (d3)\, bc \to a,\ (d4)\, bc \to d,\ (d5)\, cf \to b$.

Pour trouver une clé, on peut partir du fait que *acf* est une superclé, et chercher une clé incluse dans *acf*.

- *a*, *c*, *f* ne sont pas des membres gauches de dépendance, donc ils sont leur propre clôture.
- aucune partie de *ac* ou de *af* n'est membre gauche d'une dépendance, donc ils sont leur propre clôture.
- il reste à examiner $A = cf$. On calcule $cf^+ = abcdef$.

Donc *acf* est une superclé-non-clé, et *cf* est une clé.

Exemple (5/5) : toutes les clés

Soit $S = \{a, b, c, d, e, f\}$ un ensemble d'attribut.
Soit l'ensemble de dépendances fonctionelles : $ab \to c,\ d \to e,\ bc \to ad,\ cf \to b$.
ou de façon équivalente l'ensemble $\mathcal{D}$ de dépendances fontionnelles singletons :
$(d1)\, ab \to c,\ (d2)\, d \to e,\ (d3)\, bc \to a,\ (d4)\, bc \to d,\ (d5)\, cf \to b$.

Pour trouver toutes les clés, on remarque que *f* n'est dans aucun membre droit de dépendance, donc toute clé doit contenir *f*.

Ensuite on calcule A^+ pour tous les A qui contiennent *f* et qui contiennent aussi la partie gauche d'une dépendance fonctionnelle de $\mathcal{D}$ (donc il n'y a rien à tester pour $|A| = 1$).

On constate que la relation a deux clés : *cf* (déjà vue) et *abf*.

$\lvert A\rvert$	A	A^+	clé ?
2	*cf*	*abcdef*	clé
	df	*def*	
3	*abf*	*abcdef*	clé
	adf	*adef*	
	bdf	*bdef*	
	def	*def*	
4	*adef*	*adef*	
	bdef	*bdef*	

Nous étudirons une méthode systématique pour trouver une ou toutes les clés.

Preuve de l'algorithme : notation

Afin d'éviter les confusions, nous adoptons les notations suivantes :

- A^+ est la sortie attendue de l'algorithme, c'est-à-dire, la clôture de A.
- Nous noterons A^i la valeur de l'ensemble courant d'attributs où
  - A^0 est la valeur après initialisation (avant le premier tour de boucle) et
  - A^i avec $i > 0$ la valeur à la fin du $i^{\text{ème}}$ tour de boucle.

Preuve de l'algorithme : terminaison

À tout moment A^i est un sous-ensemble de S, et il croît strictement à chaque passage dans la boucle, *i.e.*, pour tout $i > 0$, on a $A^{i-1} \subsetneq A^i \subseteq S$.

Donc **l'algorithme termine**.

Soit $A^{\oplus}$ la sortie de l'algorithme.
Il nous reste donc à prouver que $A^{\oplus} = A^{+}$:

la sortie de l'algorithme est bien la clôture transitive de A.

Preuve de l'algorithme : correction (1/8)

Montrons, par récurrence, qu'on a pour invariant de l'algorithme :

$$I = (A \subseteq A^i \text{ et } A \to A^i), \forall i \geq 0$$

(remarque : $A \subseteq A^i \implies A^i \to A$).

Pour $A \subseteq A^i$ c'est vrai pour $i = 0$: initialement $A^0 = A$.

D'après l'algorithme, $A^{i+1} = A^i \cup \{b\}$ pour un certain $b \in S$ tel que $b \notin A^i$. Donc, $A^i \subsetneq A^{i+1}$.
De plus, par hypothèse de récurrence, $A \subseteq A^i$.
D'où, par transitivité, $A \subseteq A^{i+1}$.

Résultat : $A \subseteq A^i$ est bien un **invariant** de l'algorithme.

Preuve de l'algorithme : correction (2/8)

Montrons, par récurrence, qu'on a pour invariant de l'algorithme :

$$I = (A \subseteq A^i \text{ et } A \to A^i), \forall i \geq 0$$

(remarque : $A \subseteq A^i \implies A^i \to A$).

Pour $A \to A^i$, c'est vrai à l'initialisation par réflexivité, car initialement $A^0 = A$.

Montrons que c'est invariant dans la boucle :
Par hypothèse de récurrence, $A \to A^i$
De plus, $A^{i+1} = A^i \cup \{b\}$ avec $A' \to b$, $A' \subseteq A^i$ et $b \notin A^i$.

$$(\text{union})\ \dfrac{A \to A^i \qquad (\text{trans})\ \dfrac{A \to A^i \qquad (\text{trans})\ \dfrac{(\text{refl})\ \dfrac{A' \subseteq A^i}{A^i \to A'} \qquad A' \to b}{A^i \to b}}{A \to b}}{A \to A^i \cup b}$$

C'est-à-dire, $A \to A^{i+1}$.

Résultat : $A \to A^i$ est bien un **invariant** de l'algorithme.

Preuve de l'algorithme : correction (3/8)

$I = (A \subseteq A^i \text{ et } A \to A^i), \forall i \geq 0$ est donc bien un invariant de l'algorithme.

Il reste à montrer qu'à la sortie de la boucle $A^{\oplus}$ est maximal pour la propriété $A \to A^{\oplus}$.

Preuve de l'algorithme : correction (4/8)

D'abord, démontrons deux propriétés de l'algorithme.

Propriété : $(X^\oplus)^\oplus = X^\oplus$ (il est inutile d'appliquer 2 fois l'algorithme).

Preuve : Appliquons l'algorithme à $X^\oplus$ afin de calculer $(X^\oplus)^\oplus$.
L'initialisation pose $(X^\oplus)^0 = X^\oplus$, et la boucle n'est jamais exécutée car sa condition est initialement fausse : il n'existe aucune dépendance $A' \to b$ dans $\mathcal{D}$ avec $A' \subseteq X^\oplus$ et $b \notin X^\oplus$.
Par suite, $(X^\oplus)^\oplus = (X^\oplus)^0 = X^\oplus$.

Preuve de l'algorithme : correction (5/8)

Propriété : Si $X \subseteq Y$ alors $X^\oplus \subseteq Y^\oplus$ (l'algorithme préserve les inclusions).

Preuve : Soient X et Y deux ensembles d'attributs avec $X \subseteq Y$.
Montrons qu'à tout moment de l'algorithme appliqué à X nous avons $X^i \subseteq Y^\oplus$.

Initialisation : $X^0 = X$ donc par hypothèse $X^0 \subseteq Y$, or, d'après l'invariant, $Y \subseteq Y^\oplus$, donc $X^0 \subseteq Y^\oplus$.

Boucle : si $X^i \subseteq Y^\oplus$ à la fin du $i^{\text{ème}}$ tour de boucle, et b est ajoutée lors du $i+1^{\text{ème}}$ tour de boucle.
Alors, il existe $A' \to b$ dans $\mathcal{D}$ avec $A' \subseteq X^i$ et $b \notin X^i$.
Or $A' \subseteq Y^\oplus$ (par hypothèse) et donc, d'après l'algorithme, $b \in Y^\oplus$.
Par conséquent $X^{i+1} = X^i \cup \{b\} \subseteq Y^\oplus$.

Résultat : L'algorithme appliqué à X donne $X^\oplus \subseteq Y^\oplus$.

Preuve de l'algorithme : correction (6/8)

$A^\oplus$ est maximal si pour toute dépendance $A \to B$ conséquence de $\mathcal{D}$, on a $B \subseteq A^\oplus$.
Nous démontrons ce résultat par récurrence sur la hauteur des arbres de preuves à partir de $\mathcal{D}$ pour toute dépendance $A \to B$ conséquence de $\mathcal{D}$.

On se restreint à des preuves utilisant uniquement les trois règles de base, *i.e.*, réflexivité, transitivité, augmentation, puisque les autres règles en dérive.

Cas de base : Soit $A \to B$ une dépendance conséquence de $\mathcal{D}$ qui peut être prouvée par un arbre de hauteur 0. Donc $A \to B \in \mathcal{D}$. En particulier, $A \to B$ est une dépendance singleton.
D'après l'invariant, on a $A \subseteq A^i$ pour tout $i \geq 0$. De plus $B \notin A$, donc initialement $B \notin A^0$.
Ainsi, d'après l'algorithme, il existe $i \geq 1$, tel que $B \in A^i$.
Comme $A^i \subseteq A^\oplus$, pour tout i, on a donc $B \subseteq A^\oplus$.

Preuve de l'algorithme : correction (7/8)

Hypothèse de récurrence : Supposons que pour toute dépendance $A \to B$ conséquence de $\mathcal{D}$ qui peut être prouver par un arbre de hauteur $n \geq 0$, on a $B \subseteq A^\oplus$.

Preuve de l'algorithme : correction (8/8)

Soit $A \to B$ une dépendance conséquence de $\mathcal{D}$ qui peut être prouver par un arbre T de hauteur $n+1$. Nous avons trois cas, en fonction de la dernière règle appliquée pour obtenir la conclusion $A \to B$.

Réflexivité : Alors $B \subseteq A$. Or, d'après l'invariant de l'algorithme, on a toujours $A \subseteq A^{\oplus}$. Donc, $B \subseteq A \subseteq A^{\oplus}$.

Transitivité : On a un arbre prouvant $A \to C$ à partir de $\mathcal{D}$ de hauteur $\leq n$ et un arbre prouvant $C \to B$ à partir de $\mathcal{D}$ de hauteur $\leq n$, où C est un ensemble d'attributs.
Par hypothèse de récurrence, $C \subseteq A^{\oplus}$ et $B \subseteq C^{\oplus}$.
D'après les deux propriétés précédentes, $C \subseteq A^{\oplus}$ implique $C^{\oplus} \subseteq (A^{\oplus})^{\oplus}$ et $A^{\oplus} = (A^{\oplus})^{\oplus}$, et donc $C^{\oplus} \subseteq A^{\oplus}$.
De $B \subseteq C^{\oplus}$ et $C^{\oplus} \subseteq A^{\oplus}$, on déduit $B \subseteq A^{\oplus}$.

Augmentation : On a $A = X \cup C$ et $B = Y \cup C$ et un arbre prouvant $X \to Y$ à partir de $\mathcal{D}$ de hauteur n.
Par hypothèse de récurrence, $Y \subseteq X^{\oplus}$.
Or, $X \subseteq A$ et $C \subseteq A$.
D'une part, les inclusions sont préservées, donc $X \subseteq A$ implique $X^{\oplus} \subseteq A^{\oplus}$.
D'autre part, d'après l'invariant de l'algorithme, on a $A \subseteq A^{\oplus}$, donc $C \subseteq A$ implique $C \subseteq A^{\oplus}$.
De $Y \subseteq X^{\oplus}$ et $X^{\oplus} \subseteq A^{\oplus}$ et $C \subseteq A^{\oplus}$ on déduit $Y \cup C \subseteq A^{\oplus}$, c-à-d, $B \subseteq A^{\oplus}$.

Dépendances fonctionnelles : fermeture, conséquence, équivalence

Soient $\mathcal{D}$ et $\mathcal{D}'$ deux ensembles de dépendances fonctionnelles.

- La clôture (ou fermeture) $\mathcal{D}^+$ de $\mathcal{D}$ est l'ensemble de toutes les dépendances fonctionnelles qui sont conséquences de $\mathcal{D}$.
- $\mathcal{D}'$ est conséquence de $\mathcal{D}$ si chaque dépendance fonctionnelle de $\mathcal{D}'$ est conséquence de $\mathcal{D}$; on note $\mathcal{D} \implies \mathcal{D}'$.
- $\mathcal{D}$ et $\mathcal{D}'$ sont équivalents si $\mathcal{D} \implies \mathcal{D}'$ et $\mathcal{D}' \implies \mathcal{D}$; on note $\mathcal{D} \iff \mathcal{D}'$.

  (par exemple, on a déjà vu que $\mathcal{D}$ et l'ensemble de dépendances fonctionnelles singletons associé à $\mathcal{D}$ sont équivalents).

Propriété

- $\mathcal{D} \subseteq \mathcal{D}^+$ et $\mathcal{D}^+$ est en général « très grand » !
- si $\mathcal{D}' \subseteq \mathcal{D}$ alors $\mathcal{D} \implies \mathcal{D}'$.
- $\mathcal{D} \iff \mathcal{D}^+$.

Table des clôtures

Soit $\mathcal{D}$ un ensemble de dépendances fonctionnelles.

On appelle table des clôtures de $\mathcal{D}$ la table à deux colonnes dont les lignes sont les couples (A, A^+) pour tous les ensembles non vides d'attributs A, sauf lorsque A est une supercllé-non-clé.

Remarque. Si s est le nombre total d'attributs alors le nombre d'ensembles non vides d'attributs est $2^s - 1$.

Grâce à la dernière condition dans la définition de la table des clôtures le nombre de lignes de la table est souvent nettement plus petit que $2^s - 1$ (voir les exemples).

Algorithme de calcul de la table des clôtures

Entrée. Un ensemble $\mathcal{D}$ de dépendances fonctionnelles singletons, sur un ensemble S de s attributs.

Sortie. La table des clôtures T de $\mathcal{D}$ et l'ensemble $\mathcal{K}$ des clés de $\mathcal{D}$.

Principe. Calculer les lignes de T en utilisant le calcul de clôtures, pour les A de cardinal croissant.

-I- $\mathcal{K} = \emptyset$, $T = \emptyset$.
-Bn- Pour chaque n de 1 à s,
-BA- pour chaque A de cardinal n qui ne contient aucun élément de $\mathcal{K}$
calculer A^+ (par l'algorithme de clôture)
ajouter la ligne (A, A^+) dans T
si $A^+ = S$ alors ajouter A dans $\mathcal{K}$
-R- Retourner $\mathcal{K}$ et T.

Remarque. Il est important de ranger les ensembles A par cardinal croissant, pour rencontrer les clés avant les superclés-non-clés, mais pour un cardinal donné l'ordre n'a pas d'importance.

Utilisation de la table des clôtures

Soit $\mathcal{D}$ un ensemble de dépendances fonctionnelles et soit T sa table des clôtures. Soit S un ensemble des attributs de D.

Étant donnés A et B des sous-ensembles de S, on peut dire **si** $A \to B$ **est dans** $\mathcal{D}^+$ **ou non** :

- si A est une partie gauche d'un couple (A, C) de T, alors $C = A^+$ et :
  - si $B \subseteq C$ alors oui $A \to B$ est dans $\mathcal{D}^+$
  - sinon non $A \to B$ n'est pas dans $\mathcal{D}^+$
- si A n'est pas une partie gauche d'un couple de T alors A est une superclé et oui $A \to B$ est dans $\mathcal{D}^+$.

De plus, on connaît les clés :

- A est une clé si et seulement si A figure dans T et $A^+ = S$.

Exemple

Soit $S = \{a, b, c, d, e\}$ avec l'ensemble de dépendances fonctionnelles singletons : (1) $ab \to c$, (2) $d \to e$, (3) $bc \to a$, (4) $bc \to d$.

$\lvert A\rvert$	A	A^+	clé ?
1	a	a	
	b	b	
	c	c	
	d	de	
	e	e	
2	ab	$abcde$	clé
	ac	ac	
	ad	ade	
	ae	ae	
	bc	$abcde$	clé
	bd	bde	
	be	be	
	cd	cde	
	ce	ce	
	de	de	
3	acd	$acde$	
	ace	ace	
	ade	ade	
	bde	bde	
	cde	cde	
4	$acde$	$acde$	

Pour calculer toutes les clés

Si on veut seulement connaître les clés et pas toute la table, il est facile d'accélérer l'exécution de cet algorithme.

En effet dans les deux cas suivants on sait rapidement que A n'est pas une clé, sans qu'il soit nécessaire de calculer A^+ :

- **Si aucun membre gauche de $\mathcal{D}$ n'est contenu dans A et si** $A \neq S$ alors A n'est pas une clé : en effet alors $A^+ = A$ et $A \neq S$, donc $A^+ \neq S$.
- **S'il existe un attribut a qui n'apparaît dans aucun membre droit de $\mathcal{D}$ et qui n'est pas contenu dans** A alors A n'est pas une clé : en effet alors $a \notin A^+$, donc $A^+ \neq S$.

Pour calculer une clé (de taille minimum)

Si on veut seulement connaître une clé, on procède comme pour toutes les clés mais on s'arrête dès qu'on trouve la première clé.

De plus, puisqu'on considère les ensembles A par cardinal croissant, on est sûr que la clé trouvée est de taille minimum parmi toutes les clés.

Base

Une base de dépendances fonctionnelles est un ensemble $\mathcal{D}$ de dépendances fonctionnelles tel qu'une dépendance fonctionnelle est satisfaite par les attributs si et seulement si elle est conséquence de $\mathcal{D}$.

Propriété : si $\mathcal{D}$ est une base de dépendances fonctionnelles et si $\mathcal{D}$ et $\mathcal{D}'$ sont équivalents alors $\mathcal{D}'$ est aussi une base.

Définition

Soient $S_1, ..., S_n$ des ensembles d'attributs de R tels que $S_1 \cup ... \cup S_n = S$, où S est l'ensemble des attributs de R.

Le produit naturel (ou jointure naturelle) $R[S_1] * ... * R[S_n]$ a les mêmes attributs que R.

Les relations $R[S_1], ..., R[S_n]$ forment une décomposition de R si $R = R[S_1] * ... * R[S_n]$.
(on dit parfois décomposition sans perte : « lossless join »)

Propriété : Soient $S_1, ..., S_n$ tels que $S_1 \cup ... \cup S_n = S$. Alors $R \subseteq R[S_1] * ... * R[S_n]$ mais **en général cette inclusion n'est pas une égalité.**

Preuve : Par définition, $\forall \ell \in R$, on a :

1. $\forall i \in \{1, \dots, n\}$, $\{\ell\}[S_i] \in R[S_i]$, et
2. $\forall j, k \in \{1, \dots, n\}$ tels que $j \neq k$, $(\{\ell\}[S_j])[S_j \cap S_k] = (\{\ell\}[S_k])[S_j \cap S_k]$ (en particulier quand $S_j \cap S_k = \emptyset$)

En utilisant ces deux propriétés, on peut démontrer par récurrence que $\forall i \in \{1, \dots, n\}$, $\{\ell\}[S_1 \cup \dots \cup S_i] \in R[S_1] * \dots * R[S_i]$.
Ainsi, $\{\ell\}[S_1 \cup \dots \cup S_n] = \{\ell\}[S] = \ell \in R[S_1] * \dots * R[S_n]$.

Exemple : « Livres »

Soit $R(l, o, s, b)$ avec la **base** de dépendances fonctionnelles $\{l \to os,\ s \to b\}$.

La seule clé de R est l.

Soit $S_1 = lo$ et $S_2 = osb$, alors on peut avoir $R \neq R[lo] * R[osb]$:

R

l	o	s	b
l1	o1	s1	b1
l2	o1	s2	b1

R[lo]

l	o
l1	o1
l2	o1

R[osb]

o	s	b
o1	s1	b1
o1	s2	b1

R[lo]*R[osb]

l	o	s	b	$\in$ R ?
l1	o1	s1	b1	oui
l1	o1	s2	b1	non
l2	o1	s1	b1	non
l2	o1	s2	b1	oui

Par contre si $S_1 = los$ et $S_2 = sb$ alors $R = R[los] * R[sb]$.

Une condition suffisante pour obtenir une décomposition

Propriété : Soient S_1 et S_2 tels que $S_1 \cup S_2 = S$.
Si $S_2 \to S$ et $S_1 \cap S_2 \to S_1$, alors $R = R[S_1] * R[S_2]$.

Preuve :

- Puisque $S_2 \to S$, **toute ligne de** $R[S_2]$ **se prolonge en exactement une ligne de** R (en effet, S_2 est une superclé de R).
- Puisque $S_1 \cap S_2 \to S_1$, **toute ligne de** $R[S_2]$ **se prolonge en exactement une ligne de** $R[S_1] * R[S_2]$ (en effet, S_2 est une superclé de $R[S_1] * R[S_2]$).

Il y a donc **autant de lignes dans** R **que dans** $R[S_1] * R[S_2]$.
Or, d'après la propriété précédente, on a $R \subseteq R[S_1] * R[S_2]$.
Par suite, $R = R[S_1] * R[S_2]$.

Théorème de décomposition

Soit A un ensemble d'attributs,

alors bien sûr $A \subseteq A^+ \subseteq S$, où S est l'ensemble des attributs de la relation.

Soient $R_1 = R[A^+]$ et $R_2 = R[A \cup (S - A^+)]$, alors :

- Les attributs communs à R_1 et R_2 sont exactement les attributs de A.
- $R[A] = R_1[A] = R_2[A]$.
- A est une superclé pour R_1.
- $R_1 = R$ si seulement si $A^+ = S$.
- $R_2 = R$ si seulement si $A^+ = A$.

De plus, $R = R_1 * R_2$.

Preuve du théorème (1/3)

- Les attributs communs à R_1 et R_2 sont exactement les attributs de A :
  $A^+ \cap (A \cup (S - A^+)) = (A^+ \cap A) \cup (A^+ \cap (S - A^+)) = A \cup \emptyset = A$.
- $R[A] = R_1[A] = R_2[A]$.
  $R_1[A] = (R[A^+])[A]$ et $R_2[A] = (R[A \cup (S - A^+)])[A]$.
  Or, $A \subseteq A^+$ et $A \subseteq A \cup (S - A^+)$.
  Donc $R_1[A] = (R[A^+])[A] = R[A]$ et
  $R_2[A] = (R[A \cup (S - A^+)])[A] = R[A]$, c'est-à-dire,
  $R[A] = R_1[A] = R_2[A]$.
- A est une superclé pour R_1 : par définition, $A \to A^+$ et $R_1 = R[A^+]$.

Preuve du théorème (2/3)

- $R_1 = R$ si seulement si $A^+ = S$.
  Si $R_1 = R$ alors, R_1 et R ont, en particulier, le même ensemble d'attributs, donc $A^+ = S$.
  Si $A^+ = S$, alors $R_1 = R[A^+] = R[S] = R$.
- $R_2 = R$ si seulement si $A^+ = A$.
  Si $R_2 = R$ alors, R_2 et R ont, en particulier, le même ensemble d'attributs, donc $A \cup (S - A^+) = S$. Or, $A \cup (S - A^+) = S$ implique que $A^+ \subseteq A$. Et, par définition, $A \subseteq A^+$. Donc $A = A^+$.
  Si $A^+ = A$, alors
  $R_2 = R[A \cup (S - A^+)] = R[A \cup (S - A)] = R[S] = R$.

Preuve du théorème (3/3)

Nous démontrons maintenant que $R = R_1 * R_2$ en appliquant **la condition suffisant pour la décomposition** avec $S_1 = A^+$ et $S_2 = A \cup (S - A^+)$.

$S_1 \cup S_2 = A^+ \cup A \cup (S - A^+) = A \cup S = S$.

Par définition on a $A \subseteq A \cup (S - A^+)$ et $A \rightarrow A^+$.

$$\dfrac{\dfrac{\dfrac{A \subseteq A \cup (S - A^+)}{A \cup (S - A^+) \rightarrow A}\,(\text{refl}) \qquad A \rightarrow A^+}{A \cup (S - A^+) \rightarrow A^+}\,(\text{trans})}{A \cup (S - A^+) \rightarrow S}\,(\text{augm})$$

(Pour la dernière ligne, notez que $A^+ \cup (S - A^+) = S$.)

Donc, $S_2 \rightarrow S$.

Enfin, on sait déjà que $A^+ \cap (A \cup (S - A^+)) = A$, donc $A^+ \cap (A \cup (S - A^+)) \rightarrow A$ (par réflexivité stricte).

Comme par définition, $A \rightarrow A^+$, on a (par transitivité), $A^+ \cap (A \cup (S - A^+)) \rightarrow A^+$, c-à-d, $S_1 \cap S_2 \rightarrow S_1$.

On peut donc appliquer **la condition suffisante de décomposition** :
$R = R[S_1] * R[S_2] = R[A^+] * R[A \cup (S - A^+)] = R_1 * R_2$.

Propriétés

Soit R une relation et $\mathcal{D}$ une base de R.

Une décomposition $R = R[S_1] * \ldots * R[S_n]$ **peut vérifier, ou pas,** une des deux propriétés suivantes :

- Une décomposition $R = R[S_1] * \ldots * R[S_n]$ préserve les dépendances de $\mathcal{D}$ si toute dépendance fonctionnelle $a_1 \ldots a_n \rightarrow b$ de $\mathcal{D}$ est une dépendance fonctionnelle d'une des $R[S_i]$ (c'est-à-dire, si $a_1, \ldots, a_n, b$ sont tous dans un même S_i).

  Cela signifie que des attributs « reliés » par une dépendance sont dans la même table de la décomposition.
- Une décomposition $R = R[S_1] * \ldots * R[S_n]$ évite les redondances si on n'a jamais $S_i \subseteq S_j$ avec $i \neq j$.

  Cela signifie que les données stockées dans les différentes tables ne sont « pas trop » redondantes : en effet si $S_i \subseteq S_j$ alors toutes les données de $R[S_i]$ sont aussi dans $R[S_j]$.

Formes normales

On va définir deux formes normales :

- BCNF, forme normale de Boyce-Codd : toutes les dépendances de R se déduisent de la connaissance des clés des relations $R[S_i]$.
- 3NF, 3ème forme normale : la condition de BCNF est affaiblie.

On va étudier **trois algorithmes** pour calculer des décompositions d'une relation R, à partir d'une base de dépendances $\mathcal{D}$ sur les attributs de R :

- **Algorithme récursif pour la BCNF** qui évite les redondances, mais peut perdre des dépendances.
- **Algorithme de synthèse pour la BCNF**, qui préserve les dépendances mais peut contenir des redondances.
- **Algorithme de synthèse pour la 3NF**, qui préserve les dépendances et évite les redondances.

Il est impossible de trouver un algorithme pour obtenir « à coup sûr » une décomposition BCNF qui préserve les dépendances et évite les redondances.

Exemple : « Livres » (1/2)

Soit $R(l, o, s, b)$ avec la base de dépendances fonctionnelles $\{l \rightarrow os,\ s \rightarrow b\}$.

Soit $A = s$.
Alors, $A^+ = sb$, et le théorème donne la décomposition $R = R[\underline{s}b] * R[\underline{l}os]$, **qui évite les redondances et préserve les dépendances.**

Exemple : « Livres » (2/2)

Soit $R(l, o, s, b)$ avec la base de dépendances fonctionnelles $\{l \to os,\ s \to b,\ o \to b\}$.

Soit $A = s$.
Alors $A^+ = sb$, et le théorème donne la décomposition $R = R[\underline{s}b] * R[\underline{l}os]$, **qui évite les redondances mais ne préserve pas les dépendances**, car la dépendance fonctionnelle $o \to b$ n'est pas préservée.

On a aussi la décomposition $R = R[\underline{s}b] * R[\underline{l}os] * R[\underline{o}b]$, **qui évite les redondances et préserve les dépendances.**

Il y a quand même une forme « faible » de redondance ici, car $\{o, b\} \subseteq \{l, o, s\} \cup \{s, b\}$, donc on retrouve les données de $R[ob]$ dans la relation produit $R[los] * R[sb]$.