RAIRO. INFORMATIQUE THÉORIQUE

E. PICHAT

Algorithme de décomposition de clés

RAIRO. Informatique théorique, tome 19, n° 3 (1985), p. 213-232

<http://www.numdam.org/item?id=ITA_1985__19_3_213_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 19, n° 3, 1985, p. 213 à 232)

ALGORITHME DE DÉCOMPOSITION DE CLÉS (*)

par E. PICHAT (1)

Communiqué par J. BERSTEL

Résumé. — *La détermination d'une clé est un processus de complexité polynomiale, mais le nombre de clés d'un ensemble d'attributs peut être exponentiel. A partir du concept de graphe de contribution d'un ensemble de dépendances fonctionnelles, nous montrons que le processus de construction de clés sur un ensemble d'attributs peut être décomposé en un ensemble de processus similaires sur l'ensemble des attributs de chaque composante fortement connexe du graphe de contribution. Le mécanisme de décomposition a une complexité polynomiale. La notion de projection conditionnelle est utilisée pour la démonstration.*

Abstract. — *The determination of one key is a polynomial time process, but determining all the keys is potentially an exponential time one. We introduce the contribution graph of a set of functional dependencies. We show that the key construction process of the whole attributes can be decomposed in a set of similar processes on the set of attributes relative to each strongly connected component of the contribution graph. This decomposition mecanism has a polynomial time complexity. A new notion is used to achieve the proofs : conditional projection.*

Le modèle relationnel de données proposé par Codd [6, 7, 8, 12, 13, 19] et manipulant des dépendances fonctionnelles (en abrégé DF) est un cadre adéquat pour la conception des bases de données et plus généralement des systèmes d'information. La recherche d'une clé dans cette approche est importante. D'une part le concept de clé permet d'identifier les n-uplets d'une relation ou les enregistrements d'un fichier ; il facilite l'adressage par le contenu, le stockage et l'accès des données. D'autre part les clés sont des moyens pour définir et représenter les DF. Les langages de définition de données autorisent souvent la déclaration de clés et non la déclaration explicite de DF. Le processus de normalisation d'une base de données est fondé sur le concept de clé [5, 15, 17, 21].

La complexité du calcul d'une clé est polynomiale par rapport au nombre des attributs et au nombre des DF, mais le nombre des clés peut être une fonc-

(*) Reçu en mai 1982, révisé en janvier 1985.
(1) Conservatoire National des Arts et Métiers, Centre National d'Informatique Appliquée de Montpellier, 860 rue de Saint-Priest, 34100 Montpellier.

0399-0540/85/03 213 20/$ 4,00/

tion exponentielle du nombre des attributs et du nombre des DF. Lucchesi et Osborn [16] donnent un algorithme de durée polynomiale en le nombre des attributs, le nombre des DF et le nombre des clés. Cet article montre que la recherche de clés peut se décomposer en autant de recherches de clés qu'il y a de composantes fortement connexes dans le graphe de contribution et donne un algorithme polynomial correspondant.

En I, nous introduisons brièvement les concepts, les propriétés du modèle relationnel de données et quelques algorithmes indispensables pour la suite. En II, nous donnons un algorithme polynomial calculant une représentation irredondante et réduite d'un ensemble de DF ; cet algorithme sera utilisé par l'algorithme de décomposition de clés. La Section III montre que la recherche de clés d'un schéma relationnel $R(U, F)$ se décompose en autant de recherches de clés qu'il y a de composantes fortement connexes dans le graphe de contribution de F. Cette décomposition est plus fine si F est un ensemble de DF réduites. La Section IV énonce l'algorithme décomposant le calcul de clés ; il exhibe les plus petites unités de décomposition, auxquelles est appliquée la recherche d'une clé, de clés ou des clés.

I. RAPPEL SUR LE MODÈLE RELATIONNEL

Dans cette Section nous rappelons les concepts, propriétés et algorithmes indispensables à la compréhension de la suite et nous précisons les notations. Pour plus de détails, on pourra se reporter à [3, 8, 19].

I.1. Attributs, relations et schémas

Dans le modèle relationnel de données, les variables sont appelées des **attributs.** Un attribut sera noté par une des lettres A, B, C..., l'ensemble des attributs par U, des sous-ensembles d'attributs par G, D, K, K', L, R, V, W, X, Y, Z. Un n-uplet valeur de $V = (A, B, ..., C)$ est noté $v = (a, b, ..., c)$.

Une relation $r(U)$ peut se définir :

— soit énumérativement par un ensemble de n-uplets (n est l'ordre de la relation ou le nombre des attributs appartenant à U),

— soit à l'aide d'un prédicat, c'est-à-dire d'une règle qui à toute affectation de valeur u à U, donne une réponse exclusivement affirmative (u est un n-uplet de la relation) ou négative (u n'est pas un n-uplet de la relation).

Une relation $r(U)$ varie au cours du temps, par contre le prédicat associé est indépendant du temps. Le couple d'un ensemble U d'attributs et d'un prédicat s'appelle un **schéma relationnel** et est noté R. Un schéma relationnel à un

instant donné, a une réalisation, une valeur $r(U)$. On peut rapprocher cette distinction entre relation et schéma relationnel de celle, pour une donnée, entre son type (par exemple entier) et sa valeur (par exemple 17).

I.2. Dépendance fonctionnelle, fermeture et clé

Un type important de prédicat sont les dépendances fonctionnelles. Un schéma relationnel R défini sur un ensemble U d'attributs, vérifie la **dépendance fonctionnelle** (abrégé en DF) $f : G \to D$, où G et D sont des parties de U, si, quelle que soit la réalisation r de R, on a :

$$u, u' \in r, \quad u[G] = u'[G] \quad \text{impliquent} \quad u[D] = u'[D]$$

où $u[G]$ note la projection de u sur G. On dit que G **détermine fonctionnellement** ou **identifie** D et on dit que D est **fonctionnellement dépendant** ou est **identifié** par G. G est la **partie gauche** de la DF f, D sa **partie droite.**

Un schéma relationnel défini sur U et vérifiant un ensemble F de DF est noté $R(U, F)$.

Armstrong [1] a exhibé des règles permettant de construire à partir de DF de nouvelles DF. Il y en a plusieurs ensembles équivalents ; donnons-en un :

DF1. — réflexivité : si $X \supseteq Y$, alors $X \to Y$
DF2. — augmentation : si $X \to Y$, alors pour tout Z, $X \cup Z \to Y$
DF3. — pseudo-transitivité : si $X \to Y$ et $Y \cup Z \to W$, alors $X \cup Z \to W$
DF4. — composition-décomposition : $X \to Y \cup Z$ si seulement si $X \to Y$ et $X \to Z$.

Les DF obtenues par DF1 sont appelées **triviales.** Les DF $X \to Y$ et $X \to Z$ intervenant dans la règle DF4 seront appelées **composantes** de $X \to Y \cup Z$ et plus généralement composantes d'un ensemble de DF contenant $X \to Y \cup Z$.

La **fermeture** d'un ensemble d'attributs G (par rapport à F), notée G^+ ou G^{+F}, désigne l'ensemble des attributs déterminés fonctionnellement par G, à l'aide de F et des propriétés DF1 à DF4. Si F^* désigne l'ensemble des DF engendrées à partir de F et des propriétés DF1 à DF4, on a :

$$(X \to Y) \in F^* \text{ si et seulement si } Y \subseteq X^+$$

F^* est appelée la **fermeture** de F. Deux ensembles de DF F et H de même fermeture sont dits **fonctionnellement équivalents** et notés $F \equiv H$. En appelant DF **simple** une DF dont la partie droite est constituée d'un seul attribut, la règle DF4 entraîne que la DF $X \to \{ A, ..., B \}$ notée abusivement $X \to A \ldots B$ est fonctionnellement équivalente à l'ensemble des DF simples

$$\{ X \to A, ..., X \to B \}.$$

Algorithme de fermeture FERMETURE (F, G, G^+)

Données G : ensemble d'attributs, F : ensemble de DF
Résultat G^+ : ensemble d'attributs (* la fermeture de G *)
Algorithme voir Beeri et Bernstein [2]
Complexité $m = 0$ (card U * card F)
où card U est le nombre d'attributs de U et card F le nombre de DF de F.

Les algorithmes suivants s'en déduisent :

Algorithme d'appartenance APPARTENANCE (F, G, D, APPARTIENT)
Données F : ensemble de DF, $G \rightarrow D$: DF
Résultat APPARTIENT : booléen (* valant *vrai* si $(G \rightarrow D) \in F^*$, *faux* sinon *)
Procédure FERMETURE
Algorithme [2]

FERMETURE (F, G, G^+)
si $D \subseteq G^+$ *alors* APPARTIENT $\leftarrow$ *vrai*
sinon APPARTIENT $\leftarrow$ *faux*
Complexité 0(card U * card F)

Étant donné le schéma relationnel $R(U, F)$ et la partie X de U, K est une **clé de** X si et seulement si

$K \rightarrow X$
K est une partie de X
il n'existe pas $K' \subset K$ avec $K' \rightarrow X$.

Notons

$$F[X] = \{ G \rightarrow (D \cap X) \mid (G \rightarrow D) \in F, G \subseteq X \} \tag{1}$$

l'ensemble des DF de F de partie gauche incluse dans X et de partie droite la projection de leur partie droite sur X. Au lieu de clé de X, on peut alors parler plus explicitement de **clé du schéma relationnel** $(X, F^*[X])$. On ne peut pas par contre identifier clé de X et clé de $(X, F[X])$ comme le montre l'**exemple** suivant :

$$U = \{ A, B, C, D \}, \quad F = \{ A \rightarrow B, BC \rightarrow D \}, \quad X = \{ C, D \}.$$

AC est clé de U. Vouloir décomposer le calcul des clés en utilisant de telles projections de DF conduirait à construire $AC \rightarrow D$ à partir de F, donc F^* ; ce n'est pas pensable. C'est pourquoi la Section IV utilisera un autre type de projection de DF.

Algorithme de calcul d'une clé CLE (F, X, K)
Données F : ensemble de DF, X : ensemble d'attributs
Résultat K : ensemble d'attributs (* une clé de X *)
Procédure APPARTENANCE
Algorithme
$K \leftarrow X$
pour tout attribut $A \in X$ *faire*
  si card $K > 1$ *alors*
    APPARTENANCE (F, $K - \{ A \}$, X, APPARTIENT)
    si APPARTIENT *alors* $K \leftarrow K - \{ A \}$
Complexité 0((card U)^2 * card F)

Algorithme de calcul des clés CLES (F, X, CLES) [16]
Données F : ensemble de DF, X : ensemble d'attributs
Résultat CLES : ensemble d'ensembles d'attributs (* l'ensemble des clés de X *)
Procédure CLE

```
Algorithme
  CLE (F, X, K)
  CLES, CLESAEXAMINER ← { K }
  tant que CLESAEXAMINER non vide faire
  | extraire K de CLESAEXAMINER
  | pour tout G → D ∈ F faire
  | | SURCLE ← (K ∪ G) − D
  | | SURCLECANDIDATE ← vrai
  | | pour chaque J ∈ CLES tant que SURCLECANDIDATE faire
  | | | si SURCLE ⊇ J alors
  | | | | SURCLECANDIDATE ← faux
  | | si SURCLECANDIDATE alors
  | | | CLE (F, SURCLE, K)
  | | | CLES ← CLES ∪ { K }
  | | | CLESAEXAMINER ← CLESAEXAMINER ∪ { K }
Complexité O(card CLES + (card U)²) * card F * card CLES
```

II. REPRÉSENTATION IRREDONDANTE ET RÉDUITE

I.1. Introduction

Nous rappelons d'abord les concepts d'irredondance et de réduction conduisant à représenter des DF de façon plus ramassée. L'irredondance permet dans l'algorithme qui suit, de construire plus rapidement une représentation réduite, mais n'est pas essentielle. Par contre la réduction joue dans tout l'article un rôle essentiel ; apparenté au concept de clé, il permet, dans la section suivante, de définir une structure, l'ordre de contribution, décomposant la recherche de clés.

Une DF simple $f = (G \to A)$ **irredondante par rapport** à F appartient à F et est non engendrée par $F - \{ f \} : A \notin G^{+F-\{f\}}$.

Une DF simple $(G \to A)$ appartenant à F **réduite** est telle que $A \notin G$ et il n'existe pas $G' \subset G$, $(G' \to A) \in F^+$.

Une **représentation** H de F est tout ensemble de DF qui a la même fermeture que F : $H^* = F^*$. F et ses représentations sont donc fonctionnellement équivalents.

Une DF simple $(G \to A)$ appartenant à F **réduite** est telle que $A \notin G$ et il si et seulement si chacune de ses composantes simples est réduite (respectivement irredondante). Nous noterons MAX F une représentation réduite de F ; c'est une partie de F^* (MAX $F \subseteq F^*$) non obligatoirement incluse dans F (MAX $F \not\subseteq F$). Par *exemple*, si $F = \{ I \to N, N \to I, CIN \to HNP \}$, alors :

$$\text{MAX } F = \{ I \to N, N \to I, CN \to HP \}$$

ou $$\text{MAX } F = \{ I \to N, N \to I, CI \to H, CI \to P \}$$

ou $$\text{MAX } F = \{ I \to N, N \to I, CN \to HP, CI \to HP \}.$$

La notation MAX F pour n'importe quelle représentation réduite de F est justifiée dans le cadre de cet article : en effet la proposition 1 de la Section III montre que l'ordre de contribution de MAX F ne varie pas avec la représentation réduite de F.

L'algorithme qui suit construit une représentation irredondante et réduite d'un ensemble F de DF. Sa première étape traite l'irredondance ; elle examine successivement chaque composante simple de F, l'élimine si elle est redondante ; elle tend donc à diminuer le nombre de DF. Sa deuxième étape « réduit » chaque DF restante si elle n'est pas déjà réduite, en la remplaçant par une DF réduite ; pour cela, il suffit de remplacer sa partie gauche par une clé de cette partie gauche ; c'est possible parce que les DF sont irredondantes ; montrons-le en quatre étapes :

LEMME 1 : *Une* DF *n'est engendrable qu'à partir de* DF *identifiées par sa partie gauche.*

Démonstration immédiate par récursivité et à partir des règles DF2, DF3 et DF4. □

LEMME 2 : *Une* DF *simple irredondante peut toujours être remplacée par la* DF *réduite obtenue en remplaçant sa partie gauche par l'une de ses clés.*

Démonstration : Soit $f : G \to A$ une DF simple appartenant à un ensemble F de DF et irredondante par rapport à F. Alors, si K est clé de G, $K \to A \in F^*$ et est réduit. En effet :

1. $K \to A \in F^*$ puisque $K \to G \to A$.

2. $K \to A$ est réduit. Sinon, il existerait $K' \subset K$ avec $(K' \to A) \in F^*$. K clé de G entraîne $K' \not\to K : K'^+ \subset K^+ = G^+$. D'après le lemme 1, $K' \to A$ ne serait pas engendrable à partir de f; donc $K' \to A$ serait engendrable à partir de $F - \{ f \}$; il y aurait contradiction avec f irredondant. □

Exemple :

$$F = (B \to C,\ C \to D,\ BCE \to D)$$

où BE est clé de BCE et où $BE \to D$ n'est pas réduit (en effet $B \to D$), montre que le lemme 2 ne s'étend pas à une DF redondante (en effet $BCE \to D$ est redondante par rapport à F). □

LEMME 3 : *Un ensemble de* DF *irredondant reste irredondant après remplacement d'une* DF *simple par une* DF *plus réduite.*

Démonstration : Écrivons l'ensemble irredondant de départ :

$$F = \{ X \cup \{ A \} \to B,\ Y \to C \} \cup H$$

où $X \cup \{ A \} \rightarrow B$ désigne la DF simple qui va être remplacée par la DF plus réduite $X \rightarrow B$, où $Y \rightarrow C$ désigne une DF simple arbitraire composant F autre que $X \cup \{ A \} \rightarrow B$ et où H désigne les DF restantes. Supposons que F engendre $X \rightarrow B$: $X \rightarrow B$ est engendré à partir de $X \cup \{ A \} \rightarrow B$ et d'autres DF, d'où d'après le lemme 1

$$X \rightarrow A \tag{2}$$

F irredondant entraîne

$$C \notin Y^{+\{X \cup \{A\} \rightarrow B\} \cup H}. \tag{3}$$

Remplaçons dans F, $X \cup \{ A \} \rightarrow B$ par $X \rightarrow B$. On obtient

$$G = \{ X \rightarrow B, Y \rightarrow C \} \cup H$$

et montrons que G est irredondant. En premier lieu, $\{ Y \rightarrow C \} \cup H$ ne représente pas G ou F. En second lieu, $\{ X \rightarrow B \} \cup H$ ne représente pas G car, sinon, on aurait

$$C \in Y^{+\{X \rightarrow B\} \cup H}$$

d'où par comparaison avec (3)

$$Y \nrightarrow A \tag{4}$$

$Y \rightarrow C$ serait engendré à partir de $X \rightarrow B$, d'où d'après le lemme 1 : $Y \rightarrow X$; soit, avec (2) : $Y \rightarrow A$, en contradiction avec (4). En troisième lieu, $\{ X \rightarrow B, Y \rightarrow C \} \cup (H - \{ Z \rightarrow D \})$, où $Z \rightarrow D$ désigne une DF simple arbitraire composant H, ne représente pas G puisque $Y \rightarrow C$ et $Z \rightarrow D$ jouent le même rôle vis-à-vis de $\{ Y \rightarrow C \} \cup H$ et puisque $\{ X \rightarrow B \} \cup H$ ne représente pas G. □

LEMME 4 : *Une* DF *irredondante peut toujours être remplacée par la* DF *réduite obtenue en remplaçant sa partie gauche par l'une de ses clés.*

Démonstration immédiate à partir des lemmes 2 et 3 et de la règle DF4. □

II.2. Algorithme construisant une représentation irredondante et réduite REPIR(F, I)

Données et résultat $F = (f_i : G[i] \rightarrow D[i] \mid i = 1, 2, ..., I)$: ensemble de DF (* en fin d'algorithme, une représentation irredondante et réduite de F, composée de I DF avec I au plus égal à la cardinalité de l'ensemble F de départ *)

Procédures APPARTENANCE, CLE

```
Algorithme
(* Étape 1 : élimination des DF simples redondantes *)
  i ← 0
  tant que i ⩽ I faire
  |  i ← i + 1
  |  pour tout A ∈ D[I] faire
  |  |     D[i] ← D[i] − { A }
  |  |     APPARTENANCE (F, G[i], A, APPARTIENT)
  |  |     si non APPARTIENT alors
  |  |     |     D[i] ← D[i] ∪ { A }
  |  |     si D[I] = ∅ alors
  |  si i ≠ I alors
  |  |  |     G[i] ← G[I]
  |  |  |     D[i] ← D[I]
  |  |  |     I ← I − 1
  |  |  |     i ← i − 1
  |  |  sinon
  |  |  |     I ← I − 1
(* Étape 2 : réduction des DF *)
  pour i ← 1 pas 1 jusqu'à I faire
        CLÉ (F, G[i], G[i])
```

II.3. Commentaires

Cet algorithme a le mérite conceptuel de traiter séparément irredondance et réduction ; il en est plus efficace puisque :

— sa première étape tend à diminuer le nombre I de lignes des tableaux contenant des DF,

— au cours de sa deuxième étape, I ne varie pas.

Avant d'utiliser l'algorithme II.2, on peut avoir intérêt à :

— partitionner les DF de départ : deux DF f_i et f_j appartiennent à la même classe si et seulement si il existe une suite d'attributs A, B, ..., C telle que $A \in f_i$, $C \in f_j$, deux attributs consécutifs appartiennent à une même DF ;

— composer les DF de départ ayant même partie gauche en une seule DF ;

— voir si un attribut a une occurrence unique et située dans une partie droite de DF : la DF simple correspondante appartient à la représentation irredondante obtenue par la première étape de l'algorithme.

II.4. Complexité

Si m désigne la complexité de l'algorithme d'appartenance, la complexité de l'élimination des DF simples redondantes est exactement

$$m * \sum_{i=1}^{\text{card } F} \text{card } D[i].$$

et la complexité de la réduction des DF obtenues est au plus de :

$$m * \sum_{i=1}^{I} \text{card } G[i] .$$

La complexité de l'algorithme est donc au plus

$$m * \sum_{i=1}^{\text{card } F} (\text{card } G[i] + \text{card } D[i]) .$$

Autrement dit l'algorithme d'appartenance est itéré au plus le nombre d'occurrences d'attribut dans F, soit card U * card F.

III. DÉCOMPOSITION DE CLÉS

III.1. Généralités sur la détermination de clés

Des algorithmes ont été proposés pour trouver toutes les clés d'un schéma relationnel [9, 11, 16]. De nombreux travaux ont concerné la complexité du calcul des clés [2, 4, 10, 16, 20]. Le nombre des clés peut être une fonction exponentielle du nombre des attributs et du nombre des DF. Lucchesi et Osborn [16] donnent un algorithme de durée polynomiale en le nombre des attributs, le nombre des DF de départ et le nombre des clés.

On ne peut que prendre acte de la complexité de la détermination de clés. Le seul remède est d'essayer de décomposer le problème. Cette section propose une telle décomposition : elle ne change pas la nature du problème de la détermination de clés, mais en restreint la taille.

La décomposition de la recherche de clés repose sur le concept de graphe de contribution : moins riche que la structure des DF, la structure de contribution ne permet pas de résoudre la recherche des clés, mais elle permet de diviser ce problème exponentiel. Un graphe de contribution représente une sorte de dépendance affaiblie entre attributs, un ordre de contribution entre des classes d'attributs ayant la signification suivante : un attribut ne peut dépendre (par DF réduite) que d'attributs le précédant (au sens large) dans le préordre de contribution. La propriété 1 montre aussi que l'information apportée par l'ordre de contribution d'une représentation réduite est indépendante de cette représentation et qu'elle se dégrade pour les représentations non réduites. La propriété 2 montre que, pour certaines parties d'attributs appelées sections et choisies en fonction de l'ordre de contribution, tout identifiant inclut une clé.

Le paragraphe III.3 définit la projection conditionnelle : les DF, ainsi projetées sur X, avec les attributs n'appartenant pas à X permettent d'identifier X. La propriété 3 montre que tout identifiant de certaines parties inclut une clé conditionnelle. La propriété 4 montre que la projection conditionnelle, si l'ensemble d'attributs de projection est bien choisi, est indépendante de la représentation projetée.

Le paragraphe III.4 exhibe un processus de décomposition de la recherche des clés en fonction de l'existence de composantes fortement connexes (en abrégé cfc) dans le graphe de contribution, sans construction de DF supplémentaires.

III.2. Graphe et ordre de contribution

Dans [18] un ensemble d'attributs est préordonné de façon à ce qu'une paire de DF simples ne peut donner naissance à une troisième DF simple par pseudo-transitivité que si leurs attributs droits sont comparables. Cette idée était dans [14]. Ci-après, ce préordre est à nouveau utilisé sous le nom de graphe de contribution; l'ordre induit par un graphe de contribution sur ses cfc est appelé ordre de contribution.

Le **graphe de contribution** d'un ensemble de DF F est un graphe :

— de sommets les différents attributs de F; leur ensemble est noté U',

— d'arcs (A, B) (on dira que A **contribue** à B) si et seulement si F contient une DF $X \rightarrow Y$ avec A appartenant à X et B appartenant à Y.

Sur U', définissons la relation d'équivalence suivante : $A \equiv_F B$ si et seulement si $A = B$ et/ou le graphe de contribution a un chemin de A à B et un chemin de B à A. La classe d'équivalence de A sera notée $[A]_F$. Les classes d'équivalence sont ordonnées selon la relation de contribution; cet ordre sera appelé **ordre de contribution** de F et noté $(U'/\equiv_F, \leqslant_F)$:

— ses éléments, notés S ou $[A]_F$ si A est un attribut appartenant à S, sont les différences cfc du graphe de contribution de F,

— $[A]_F \geqslant_F [B]_F$ si et seulement si $[A]_F = [B]_F$ et/ou le graphe de contribution de F a un chemin de A à B.

Exemple [16] : Un fichier d'enregistrements « étudiant » avec les attributs : cours C, numéro d'identification d'élève I, nom d'élève N, professeur P, heure H :

$$U = \{ C, H, I, N, P \}$$

et les DF :

$$F = \{ I \rightarrow N, N \rightarrow I, CN \rightarrow HP, HP \rightarrow C \}.$$

Le graphe de contribution de F et l'ordre de contribution de F représenté par son diagramme de Hasse sont donnés respectivement figures 1 et 2.

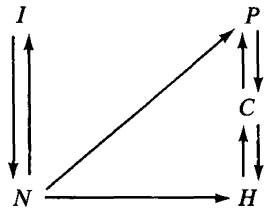

Figure 1. — Graphe de contribution de F.

$$\{ I, N \} \longrightarrow \{ C, H, P \}$$

Figure 2. — Diagramme de Hasse de l'ordre de contribution de F.

PROPRIÉTÉ 1 : *L'ordre de contribution d'une représentation réduite* MAX F *d'un ensemble* F *de* DF *est un invariant de* F. *L'ordre de contribution d'un ensemble* F *de* DF *s'obtient à partir de l'ordre de contribution de* MAX F *par fusion de sommets et/ou adjonction d'arcs. Il en résulte que si* $X \cup \{ A \} \rightarrow B$ *est une* DF *simple réduite, alors la classe d'équivalence de* A *précède (et/ou est) la classe d'équivalence de* B *dans l'ordre de contribution de* F :

$$[A]_F \geqslant_F [B]_F .$$

Démonstration : Partons du graphe de contribution de F et voyons son évolution quand F engendre toutes les DF réduites de F^* :

— soit par la règle DF3 de pseudo-transitivité

$$X \rightarrow Y \quad \text{et} \quad Y \cup Z \rightarrow W \text{ entraînent } X \cup Z \rightarrow W .$$

L'application de cette règle introduit dans le graphe de contribution des arcs (A, C) avec $A \in X$, $C \in W$. Ces arcs sont des arcs de transitivité de (A, B) et (B, C), si B désigne un attribut appartenant à Y. La règle DF3 ne modifie donc pas la connexité du graphe de contribution ;

— soit par les règles DF1, DF2 et DF4 autorisant respectivement d'éliminer des DF triviales, d'éliminer une DF devant une autre plus réduite, de décomposer ou de composer une DF. L'application de ces règles, n'adjoignant aucun arc au graphe de contribution, ne peut que réduire sa connexité.

Dans le cas où F est une représentation réduite, alors son ordre de contribution n'est pas modifié. Il est donc identique à l'ordre de contribution de toutes les DF réduites de F^*.

En particulier, si la classe d'équivalence de A précède la classe d'équivalence de B dans l'ordre de contribution de MAX F, alors c'est a fortiori vrai dans l'ordre de contribution de F :

$$[A]_{\text{MAX}F} \geqslant_{\text{MAX}F} [B]_{\text{MAX}F} \text{ entraîne } [A]_F \geqslant_F [B]_F . \quad \square$$

Exemple : Compliquons l'exemple précédent :

$$U = \{ A, B, C, D, E, H, I, M, N, P \}$$

$$F = \{ A \to B, B \to AM, AI \to B, CN \to HP, EN \to H, HP \to C, I \to N, N \to I, NP \to N \}$$

$$\text{MAX } F = \{ A \to B, B \to AM, CN \to HP, EN \to H, HP \to C, I \to N, N \to I \} .$$

Le graphe de contribution de F est représenté figure 3, celui de MAX F figure 4, l'ordre de contribution de F est représenté figure 5 et celui de MAX F figure 6.

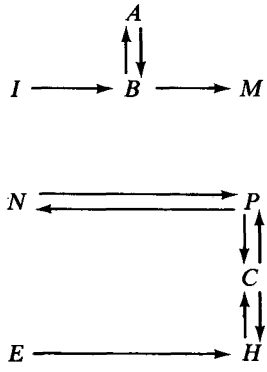

Figure 3. — Graphe de contribution de *F*.

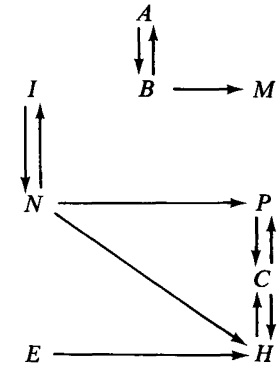

Figure 4. — Graphe de contribution de MAX *F*.

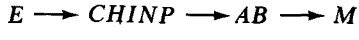

Figure 5. — Diagramme de Hasse de l'ordre de contribution de *F*.

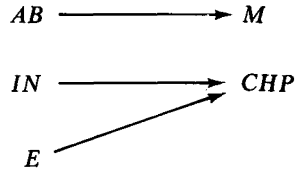

Figure 6. — Diagramme de Hasse de l'ordre de contribution de MAX *F*.

Appelons **section** T d'un ensemble F de DF la réunion des (parties de U' associées aux) sommets d'une section commençante de l'ordre de contribution de F. Autrement dit une section T de F est toute réunion de sommets de l'ordre de contribution de F telle que si T inclut un sommet S de l'ordre de contribution, alors T inclut tout sommet V précédant strictement $S(V >_F S)$.

PROPRIÉTÉ 2 : *Tout identifiant d'une section T d'un ensemble F de* DF *inclut une clé de T.*

Démonstration : Appelons X un identifiant de $T : X \to T$. Pour tout attribut A appartenant à T, ou X contient A ou il existe une partie X_A de X telle que $X_A \to A$ soit réduit. Si X_A n'était pas inclus dans T, il existerait $B \in X_A - T$ et la propriété 1 entraînerait $[B]_F \geqslant [A]_F$; ce serait en contradiction avec la propriété de T d'être section. X_A est donc inclus dans T, pour tout A ; X contient un identifiant de A inclus dans T, pour tout A, donc un identifiant de T inclus dans T, donc une clé de T. □

III.3. Projections, parties et clés conditionnelles

Pour décomposer le calcul des clés, définissons le type de projection de DF suivant : la **projection conditionnelle de F sur X**

$$FIXI = \{ (G \cap X) \to (D \cap X) \mid (G \to D) \in F \} \tag{5}$$

désigne l'ensemble des DF obtenues par projection sur X (des parties gauche et droite) de chaque DF appartenant à F. La projection conditionnelle n'a un sens que si $G - (G \cap X)$ est identifié par ailleurs ; elle sera appliquée aux parties V de U telles qu'aucun de leurs attributs ne soit identifié par $U - V$; ces parties seront appelées par analogie **parties conditionnelles** de $R(U, F)$

$$V \cap (U - V)^+ = \varnothing . \tag{6}$$

Une **clé conditionnelle de V** est toute partie K_V de V qui détermine conditionnellement (c'est-à-dire à l'aide de $FIVI$) V et telle qu'il n'existe pas $K' \subset K_V$ avec K' déterminant conditionnellement V. La projection conditionnelle a les propriétés suivantes :

PROPRIÉTÉ 3 : *Tout identifiant d'une partie conditionnelle V d'un schéma relationnel $R(U, F)$ inclut une clé conditionnelle de V.*

En effet aucun attribut appartenant à V est identifié par $U - V$. Supposons $U - V$ identifié. $FIVI$ définit la filiation entre attributs appartenant à V. Tout identifiant de V inclut évidemment une clé conditionnelle de V. □

PROPRIÉTÉ 4 : *Soit $R(U, F)$ un schéma relationnel et V une partie conditionnelle de R. Alors*

$$F \equiv H \text{ entraîne } FIVI \equiv HIVI. \tag{7}$$

Démonstration : Au cours de la démonstration, A désignera un attribut de $U - V$, B et C des attributs de V, X une partie de $U - V$ et X_V une partie de V.

$F \equiv H$ entraîne l'existence d'une suite $L(0)$, $L(1)$, ..., $L(n)$, $L(n+1)$, ..., $L(N)$ d'ensembles de DF fonctionnellement équivalents vérifiant :

— $L(0) = F$,

— $L(N) = H$,

— $L(n+1)$ se déduit de $L(n)$ ($n = 0, 1, 2, ..., N-1$) par l'une des opérations suivantes sur les DF :

(i) si $(X \cup X_V \cup Y \cup Y_V \to B)$, $(\{B\} \cup Y \cup Y_V \cup Z \cup Z_V \to C) \in L(n)$, avec $X_V \cup Y_V \neq \varnothing$ d'après (2), adjonction de

$$X \cup X_V \cup Y \cup Y_V \cup Z \cup Z_V \to C,$$

(ii) si $(X \cup X_V \cup Y \cup Y_V \to A)$, $(\{A\} \cup Y \cup Y_V \cup Z \cup Z_V \to C) \in L(n)$, avec $Y_V \cup Z_V \neq \varnothing$ d'après (2), adjonction de

$$X \cup X_V \cup Y \cup Y_V \cup Z \cup Z_V \to C,$$

(iii) et (iv) si $(X \cup X_V \to B) \in L(n)$, avec $X_V \neq \varnothing$ d'après (2), suppression de $(X \cup X_V \cup Y \cup Y_V \to B)$ s'il appartient à $L(n)$ ou adjonction s'il n'appartient pas à $L(n)$,

(iv) adjonction ou suppression d'une DF de partie droite $\subseteq U - V$.

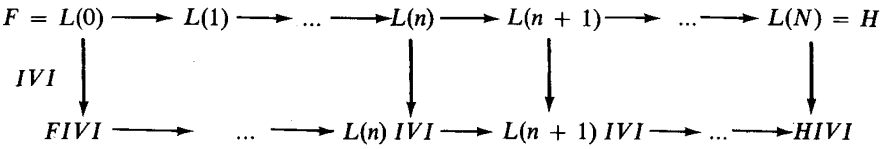

Figure 7. — Equivalence fonctionnelle et projection conditionnelle.

Pour démontrer (7), d'après la figure 7, il suffit de montrer que

$$(L(n))\, IVI \equiv (L(n+1))\, IVI$$

sont fonctionnellement équivalents. Examinons les cinq cas possibles :

(i) $L(n+1) = L(n) \cup \{ X \cup X_V \cup Y \cup Y_V \cup Z \cup Z_V \to C \}$ avec

$$X \cup X_V \cup Y \cup Y_V \to B,$$

$$\{ B \} \cup Y \cup Y_V \cup Z \cup Z_V \to C \in L(n),\ X_V \cup Y_V \neq \varnothing;$$

d'où $(L(n+1))\ IVI = (L(n))\ IVI \cup (X_V \cup Y_V \cup Z_V \to C)$ s'obtient à partir de $(L(n))\ IVI$ en appliquant DF3 à

$$X_V \cup Y_V \to B \quad \text{et} \quad \{ B \} \cup Y_V \cup Z_V \to C.$$

(ii) $L(n+1) = L(n) \cup \{ X \cup X_V \cup Y \cup Y_V \cup Z \cup Z_V \to C \}$ avec

$$X \cup X_V \cup Y \cup Y_V \to A,$$

$$\{ A \} \cup Y \cup Y_V \cup Z \cup Z_V \to C \in L(n),\quad Y_V \cup Z_V \neq \varnothing;$$

d'où $(L(n+1))\ IBI = (L(n))\ IBI \cup \{ X_V \cup Y_V \cup Z_V \to C \}$ s'obtient à partir de $(L(n))\ IBI$ en appliquant DF2 à $(Y_V \cup Z_V \to C)$.

(iii) $L(n+1) = L(n) - \{ X \cup X_V \cup Y \cup Y_V \to B \}$ avec

$$(X \cup X_V \cup Y \cup Y_V \to B),\quad (X \cup X_V \to B) \in L(n),\quad X_V \neq \varnothing;$$

d'où

$(L(n+1))\ IVI = (L(n))\ IVI$ si $(X_V \cup Y_V \to B$ est la projection sur V d'une DF $\in L(n+1)$

$= (L(n))\ IVI - \{ X_V \cup Y_V \to B \}$ sinon, c'est-à-dire s'obtient à partir de $(L(n))\ IVI$ en éliminant $(X_V \cup Y_V \to B)$ devant $(X_V \to B)$ à cause de DF2.

(iv) $L(n+1) = L(n) \cup (X \cup X_V \cup Y \cup Y_V \to B)$ avec

$$(X \cup X_V \to B) \in L(n),\quad X_V \neq \varnothing;$$

d'où

$(L(n+1))\ IVI = (L(n))\ IVI$ si $(X_V \cup Y_V \to B)$ est la projection sur V d'une DF $\in L(n+1)$

$= (L(n))\ IVI \cup (X_V \cup Y_V \to B$ sinon, c'est-à-dire s'obtient à partir de $(L(n))\ IVI$ en adjoignant $(X_V \cup Y_V \to B)$ redondant devant $(X_V \to B)$.

(v) $(L(n+1))\ IBI = (L(n))\ IBI$. □

III.4. Principe de la décomposition des clés

PROPRIÉTÉ 5 : *Toute clé K de schéma relationnel $R(U, F)$ contient et contient seulement :*

— *l'ensemble $U - U'$ des attributs n'appartenant pas à F*

— *et, pour chaque composante fortement connexe S du graphe de contribution de F, une clé conditionnelle $K_{S'}$ de $S' = S - (U - S)^+$:*

$$K = (U - U') \cup \left(\bigcup_S K_{S'} \right). \tag{8}$$

Démonstration : Raisonnons par induction sur la suite suivante de réunions de $U - U'$ et des sections $\varnothing$, W, ..., T, $T \cup V$, ..., U' de F croissantes par rapport à l'inclusion ensembliste

$U - U'$

$(U - U') \cup W$ si W désigne une cfc du graphe de contribution de F sans prédécesseur

...

$(U - U') \cup T$

$(U - U') \cup T \cup V$ si V désigne une cfc du graphe de contribution de F non incluse dans T et si les prédécesseurs stricts de V dans l'ordre de contribution de F, sont inclus dans T

...

U

La clé de $U - U'$ est $U - U'$ puisque chaque attribut appartenant à $U - U'$ n'appartient à aucune partie droite de DF appartenant à F. Supposons que toute clé de $(U - U') \cup T$ est obtenue par la formule (8), S désignant chaque cfc du graphe de contribution de F incluse dans T. Montrons que la formule (8) reste vraie pour $(U - U') \cup T \cup V$, c'est-à-dire que chacune de ses clés est la réunion de n'importe quelle clé de $(U - U') \cup T$ et de n'importe quelle clé conditionnelle de $V - (U - V)^+$.

Tout identifiant X de $(U - U') \cup T \cup V$ inclut $U - U'$, puisque chaque attribut appartenant à $U - U'$ n'appartient à aucune partie droite de DF appartenant à F. X, identifiant T, inclut aussi une clé K_T de T d'après la propriété 2, donc $(U - U') \cup K_T$, donc une clé de $(U - U') \cup T$. Identifiant la partie conditionnelle $V' = V - (U - V)^+$, il inclut aussi une clé conditionnelle $K_{V'}$ de V' d'après la propriété 3.

Inversement, la réunion d'une clé $K_{(U-U') \cup T}$ de $(U - U') \cup T$ et d'une clé conditionnelle $K_{V'}$ de $V' = V - (U - V)^+$ est une clé de

$$(U - U') \cup T \cup V .$$

En effet $K_{(U-U')\cup T} \cup K_{V'}$ identifie $(U - U') \cup T \cup V$, puisque tout attribut $B \in V - V'$ appartient à $(U - V)^+$, donc à T^+ d'après la propriété 1.

$$K_{(U-U')\cup T} \cup K_{V'}$$

est une clé de $(U - U') \cup T \cup V$, sinon

— soit il existerait un attribut $C \in K_{V'}$ tel que

$$(K_{(U-U')\cup T} - \{C\}) \cup V \to C, \quad K_{(U-U')\cup T} - \{C\} \to C$$

d'après la propriété 1, soit $K_{(U-U')\cup T} - \{C\}$ identifiant $(U - U') \cup T$, en contradiction avec $K_{(U-U')\cup T}$ clé de $(U - U') \cup T$

— soit il existerait un attribut $C \in K_{V'}$ tel que

$$(U - U') \cup T \cup (K_{V'} - \{C\}) \to V',$$

soit $K_{V'} - \{C\}$ détermine conditionnellement V', en contradiction avec $K_{V'}$ clé conditionnelle de V'. □

IV. ALGORITHME DE DÉCOMPOSITION DE CLÉS

IV.1. Algorithme trouvant les clés du schéma relationnel $R(U, F)$

données U : ensemble d'attributs, F : ensemble de DF
résultat CLES : ensemble d'ensembles d'attributs (* les clés de U *)
procédures REPIR, CLES
variable
  PRODCLES : produit cartésien d'ensembles de parties de U
algorithme
  PRODCLES ← $(U - U')$ (* singleton de l'ensemble des attributs appartenant à U et non à F *)
  REPIR (F, card F) (* F devient une représentation réduite et irredondante *)
  construire le graphe de contribution de F et ses composantes fortement connexes S, puis l'ordre de contribution de F
  pour chaque S sans prédécesseur ou de cardinalité $\geqslant 2$ *faire*
    si S admet un prédécesseur *alors*
      $S' = S - (U - S)^+$
      si card $S' \geqslant 2$ *alors*
        FIS' I
        CLES (*FIS' I*, S', CLES)
        PRODCLES ← PRODCLES × CLES
    sinon
      FISI
      CLES (*FISI*, S, CLES)
      PRODCLES ← PRODCLES × CLES
  CLES ← ensemble des réunions des composantes de chaque uplet de PRODCLES

IV.2. Commentaires et complexité

La procédure REPIR, en construisant une représentation réduite de F, permet de construire la décomposition la plus fine possible, conformément à la propriété 1.

Toute cfc précédée par au moins une autre cfc dans le graphe de contribution et de cardinalité 1 n'est pas considérée puisque son attribut est identifié par les attributs que le précèdent.

Toute cfc de cardinalité $\geqslant 2$, voit ses attributs qui sont identifiés par d'autres cfc, éliminés. Au lieu de construire $S' = S - (U - S)^+$, on ne construira que $S' = S - (T_{<S})^+$, où $T_{<S}$ désigne l'ensemble des attributs précédant strictement S dans le graphe de contribution de F.

Pour toute cfc indécomposable, on peut modifier la procédure CLES du paragraphe I.2 pour construire toutes les clés, par un test d'arrêt afin qu'elle ne construise qu'un certain nombre de clés. On peut aussi utiliser la procédure CLE pour ne construire qu'une clé. Cependant le coût de la décomposition, de sa mise en œuvre, est ainsi évaluable en nombre d'opérations élémentaires :

$0((\text{card } U * \text{card } F)^2)$ pour REPIR

$0((\text{card } U)^3)$ pour la construction de cfc

$0((\text{card } U)^2 * \text{card } F)$ pour la construction des S'.

Il est indépendant du nombre de clés à construire et est d'autant plus amorti qu'il est utilisé pour la construction de nombreuses clés. De coût polynomial en card U et card F, la décomposition restreint :

— l'ensemble des attributs dont on cherche les clés, de U à des S ou S'

— et l'ensemble des DF définissant les clés, de F à des $FISI$ ou $FIS'I$.

IV.3. Exemple

Reprenons l'exemple illustrant la propriété 1 et cherchons toutes les clés de U :

PRODCLES	$\{ D \}$				
REPIR	$\{ A \to B, B \to AM, CN \to HP, EN \to H, HP \to C, I \to N, N \to I \}$				
Graphe et ordre de contribution	voir figures 5 et 6				
S	E	IN	CHP	AB	M
$T_{<S}$			EIN		
$(T_{<S})^+$			$EHIN$		
$S' = S - (T_{<S})^+$			CP		
$FISI$ ou $FIS'I$		$\{ I \to N, N \to I \}$	$\{ C \to P, P \to C \}$	$\{ A \to B, B \to A \}$	
CLES	$\{ E \}$	$\{ I, N \}$	$\{ C, P \}$	$\{ A, B \}$	
PRODCLES	$\{ D \} \times \{ E \} \times \{ I, N \} \times \{ C, P \} \times \{ A, B \}$				
CLES	$\{ ACEI, BCEI, AEIP, BEIP, ACEN, BCEN, AENP, BENP \}$				

BIBLIOGRAPHIE

[1] W. W. ARMSTRONG, *Dependency structures of database relationships*. Proc. IFIP 1974, North Holland, 1974, 580-583.

[2] C. BEERI, P. A. BERNSTEIN, *Computational problems related to the design of normal form relational schemes*. ACM TODS, 4, 1, march 1979, 30-59.

[3] C. BEERI, P. A. BERNSTEIN, N. GOODMAN, *A sophisticate's introduction to database normalization theory*. Proc. 4th Conf. on Very Large Data Bases, West Berlin, ACM, N.Y., 1978, 113-124.

[4] A. BEKESSY, J. DEMETROVICS, *Contribution to the theory of database relations*. 1979, 1-10.

[5] P. A. BERNSTEIN, *Synthesing third normal form relations from functional dependencies*. ACM TODS, 1, 4, Dec. 1976, 277-298.

[6] E. F. CODD, *A relational model of data for large shared data banks*. Comm. ACM, 13, *switching functions*. IBM Res. Develop., 17, 1973, 374-386.

[7] E. F. CODD, *Further normalization of the data base relational model*. Courant Institute Computer Science Symposia Series, 6 : Data Base Systems, Prentice-Hall, 1971, 33-64.

[8] C. DELOBEL, M. ADIBA, *Bases de données et systèmes relationnels*. Bordas, Paris, 1982.

[9] C. DELOBEL, R. G. CASEY, *Decomposition of a data base and the theory of Boolean switching functions*. IBM Res. Develop, 17, 1973, 374-386.

[10] J. DEMETROVICS, *On the number of candidate keys*. Information processing letters, 7, 6, 1978, 266-269.

[11] R. FADOUS, J. FORSYTH, *Finding candidate keys for relational data bases*. Proc. ACM SIGMOD, 1975, 203-210.

[12] G. GARDARIN, *Bases de données*. Eyrolles, Paris, 1983.

[13] S. S. ISLOOR, *An algorithm with logical simplicity for designing third normal form relational database schema from functional dependencies*. ICMOD 78, 29-30 June 1978, FAST, Milano, Italy, 31-50.

[14] M. LEONARD, *Aides algorithmiques à la conception de bases de données*. Thèse de 3e cycle, Univ. Grenoble, juin 1976.

[15] T. W. LING, F. W. TOMPA, T. KAMEDA, *An improved third normal form for relational databases*. Department of Computer Science, University of Waterloo, Ontario, Canada, Research report, 1979, 15 p.

[16] C. L. LUCCHESI, S. L. OSBORN, *Candidate keys for relations*. Journal of Computer and System Sciences, 17, 1978, 270-279.

[17] E. PICHAT, *Algorithme construisant une base de données sous 3e forme normale irredondante ou la réconciliation des algorithmes de synthèse et de décomposition*. Actes des Journées d'études ADI « Bases de données », Toulouse, nov. 1983, 139-156.

[18] E. PICHAT, C. DELOBEL, *Designing nonredundant third normal form relational database schema*. Rapport de recherche nº 149, IMAG, Université de Grenoble, janvier 1979, 39 p.

[19] J. D. ULLMAN, *Principles of database systems*. Computer Science Press, 1982.

[20] C. T. YU, D. T. JOHNSON, *On the complexity of finding the set of candidate keys for a given set of functional dependencies*. Information Processing letters, 5, 4, October 1976, 100-101.

[21] C. ZANIOLO, *A new normal form for the design of relational database schema*. ACM TODS, 7, 3, Sept. 1982, 489-499.