

A1

**DEMANDE
DE BREVET D'INVENTION**

(21)

N° 80 21182

(54) Suppression de la codification de l'information, codification systématique, nouvelle structure physique de fichier informatique.

(51) Classification internationale (Int. Cl.³). H 03 K 13/00; G 06 F 5/00.

(22) Date de dépôt..... 2 octobre 1980.

(33) (32) (31) Priorité revendiquée :

(41) Date de la mise à la disposition du public de la demande..... B.O.P.I. — « Listes » n° 14 du 9-4-1982.

(71) Déposant : NIETO épouse DELL'AMICO Mercedes, résidant en France.

(72) Invention de : Mercedes Nieto, épouse Dell'Amico.

(73) Titulaire : *Idem* (71)

(74) Mandataire : Mercedes Dell'Amico,
48, rue de Champoigny, Ingré, 45140 Saint-Jean-de-la-Ruelle.

SUPPRESSION DE LA CODIFICATION DE L' INFORMATION
ET REMPLACEMENT PAR LA CODIFICATION SYSTEMATIQUE
DE L'ALPHABET POUR ORDINATEUR

5 Mon expérience professionnelle en Entreprise comme
chef de groupe Export et ma formation complémentaire en
Informatique de Gestion m'ont permis de trouver une
structure nouvelle de Fichier Informatique. Cette structure
permet d'avoir systématiquement une écriture séquentielle
de Fichier triée par ordre croissant avec accès direct à
chacune information.

10 J'ai en effet l'impression qu'il est possible de traiter
l'information sans la coder, pour le moment je pense aux
codes Clients et aux codes articles (gamme) que l'on ajoute
dans les Entreprises. J'ai donc essayé de démontrer par des
exemples simples mais concrets si cette impression pouvait
être une certitude; après l'étude que je viens de faire, je
pense que c'est une certitude car il est possible de supprimer
ces codes clients et codes articles; on obtient ainsi une
15 structure nouvelle de Fichier extrêmement performante comme
l'indique mon premier paragraphe et l'exposé qui va suivre.

Dans le Hash-Coding à l'heure actuelle :
(terme Anglo-Saxon qui signifie chercher l'information sans
avoir recours à un index)

clé → donne adresse de l'enregistrement :

20 A * 1 COMSAT : 70ème dans base des données) collision
SATCOM : 70ème dans base des données)
chaîne des homonymes avant de trouver l'information,
l'accès n'est pas direct car encombrement, collision.

B = 2

25 Z = 26

Puisque chaque lettre de l'alphabet correspond à un
chiffre, j'ai tapé sur une perfo les 26 lettres sur une carte
et j'ai fait les remarques suivantes :

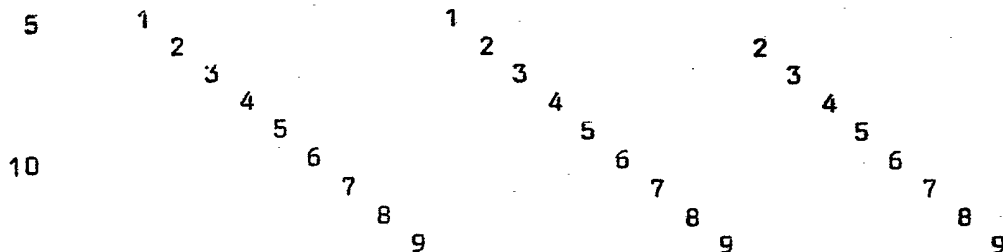
30 - chaque frappe de lettre comporte en plus du chiffre
une perforation :

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

I I I I I I I I I

I I I I I I I I I

O O O O O O O O



15 1er Rang de A à I

123456789 sont les chiffres perforés, en plus chaque lettre fait une perforation en dehors de tout chiffre, cette perforation donne le rang correspondant

20 2ème Rang de J à R

123456789 sont les chiffres perforés, en plus chaque lettre fait une perforation en dehors de tout chiffre, cette perforation indique; le rang correspondant et les perforations ont sauté d'un rang vers le bas

25

3ème Rang de S à Z

23456789 sont les chiffres perforés (le 1 n'est pas perforé) en plus chaque lettre fait une perforation dans la colonne des 0, cette perforation indique le rang correspondant, ces 8 perforations ont sauté aussi d'un rang vers le bas

30

-3-

Compte tenu de cela, j'ai tapé sur 3 cartes différentes

les lettres :

A à I

J à R

S à Z

5 et comme toutes les données se tapent en colonne 1 ces 3 cartes se superposent suivant les remarques indiquées ci-dessous :

	1	2	3	4	5	6	7	8	9	
	A	B	C	D	E	F	G	H	I	(1 ^{er} RANG)
	J	K	L	M	N	O	P	Q	R	(2 ^{ème} ")
le 1 n'est pas perforé →		S	T	U	V	W	X	Y	Z	(3 ^{ème} ")

10 frappe sur les chiffres indiqués + perforation en dehors de tout chiffre sauf pour le 3^{ème} rang colonne des 0 perforée
saut d'un rang vers le bas, mêmes remarques que précédemment, donc le code des lettres de l'alphabet est toujours chiffré de 1 à 9 sauf pour le 3^{ème} rang qui est chiffré de 2 à 9.

15 En somme, c'est le saut d'un rang vers le bas qui indique à l'ordinateur si la lettre est un A ou un J, un B ou un K ou un S etc ... ainsi que le démontre le tableau alphabétique construit ci-dessus. Je constate d'après ces remarques que, quel que soit la longueur du nom toutes les combinaisons ou concatenations sont écrites avec les chiffres 1 à 9 et 2 à 9
20 pour les lettres du 3^{ème} rang.

Compte tenu de cela, un Nom étant un ensemble de lettres et puisque chaque lettre à sa frappe engendre systématiquement un chiffre, l'ensemble de ce Nom devient donc un code chiffré.

25 Je me suis alors demandé à quoi servait le code client que l'on a toujours ajouté pour distinguer un client d'un autre lors du traitement de l'information et j'ai essayé de comprendre :

exemple : le NOM COMSAT 6 lettres, à sa frappe le code engendré est : je me sers du tableau alphabétique pour le savoir :

30 COMSAT
364213

quelques homonymes de ce nom :
SATCOM TASCOR MOSCAT
213364 312364 462313

35 Tous les codes des homonymes sont bien différents malgré que le nombre de lettres soit égal, en fait c'est la position inversée de chaque lettre qui engendre un code différent donc la différence d'écriture créée systématiquement

une différence de code ceci implique que le code que l'on ajoute lors du traitement de l'information pour distinguer un client d'un autre est un code superflu.

J'ai voulu savoir ce que ce code superflu provoquait exemple :

5 code client nom client
 238050 MARTINON (ca nom à sa frappe
 41939565 engendre son propre
 code)

la clé est le code 238050 l'ordinateur cherche le nom MARTINON à l'aide de cette clé mais le code 238050 n'étant pas le code systématique engendré par la frappe du nom MARTINON 41939565 l'ordinateur sortira les homonymes du nom MARTINON, l'accès n'est pas direct (collision, chaîne d'homonymes avant de trouver l'information souhaitée provoquant un encombrement important, c'est l'inconvénient du Hash-Coding (terme Anglo-Saxon qui signifie chercher l'information sans avoir recours à un index).

15 Maintenant, je veux savoir ce qui se passe en supprimant le code clé 238050, le nom MARTINON a créé systématiquement son code à la frappe, ce code est devenu sa propre clé et si clé = MARTINON alors l'ordinateur ira chercher le nom MARTINON sans s'occuper des homonymes de ce nom, l'accès est 41939565
20 direct car je me positionne directement sur l'information que je cherche.

Si je procède de cette façon il faut prévoir le cas de deux clients qui se nomment tous les deux MARTIN, ce cas peut être fréquent et comme il faut faire une règle générale je vais donc toujours faire figurer dans mon Fichier, le nom et le prénom du client et ce prénom fera la différence de code lorsque ce cas se présentera.

Mon Fichier Clients aura toujours la structure suivante :

colonne 1	Nom et Prénom	Adresse	code Pays
	26 octets	24 octets	9(3)

(le code pays est aussi le code monnaie de facturation)

30 En somme j'aurai l'identité du Client puis toutes les informations qui se rapportent à cette identité, lecture linéaire, accès linéaire, clé d'accès pour se positionner sur l'information que l'on cherche : 1er octet gauche du Nom client.

35 En somme une ligne = ensemble complet des informations concernant le profil d'un individu. Pour le tri: Ascending Key : A (terme Anglo-Saxon qui signifie clé par ordre croissant)
1er octet gauche du Nom

NB : en supprimant la codification superflue le tri aussi devient plus performant, ainsi que je l'ai démontré (voir tableau alphabétique). Puisque l'accès se fait par la 1ère lettre ou octet gauche de chaque Nom client cette lettre engendre à la frappe son rang correspondant, l'accès est plus court puisque pour les 26 lettres de l'alphabet on a seulement un total de 9 clés ou chiffres correspondant à 3 Rangs différents et cela quelle que soit la longueur du fichier :

- pour 1000 clients on aura un total de 9 clés et 3 Rangs
- " 10000 " on aura toujours un total de 9 clés et 3 Rangs différents.

Par contre, avec la codification superflue on a autant de clés ou codes différents que de nombre de clients et varie suivant la longueur de fichier :

- pour 1000 clients on utilisera 1000 codes différents
- pour 10000 " " " 10000 " "

En ce qui concerne la nouvelle structure physique de mon Fichier Clients, le Nom et le prénom seront toujours séparés par un espace et je reprends l'exemple :

- Nom à 6 lettres :

COMSAT	ARENE	
364213	9555	← même champ
SATCOM	JEAN	
213364	1515	

- Nom à 8 lettres :

MARTINON	ALBERT	
41939565	132593	← même champ
MARNOTIN	PIERRE	
41956395	795995	

suivant chaque groupe de Nom (j'appelle groupe un même nombre et un même genre de lettres) à l'intérieur de chaque groupe, l'espace partage toujours le même champ, l'égalité n'est pas détruite, l'espace marquera la borne entre le Nom et le prénom, si cette borne se déplace à droite ou à gauche, l'ordinateur comprendra que le Nom est plus long ou plus court, l'égalité ne sera pas détruite à l'intérieur de groupe car l'important est que l'espace partage toujours le même champ pour chaque groupe de Nom (groupe de Nom à 6 lettres - groupe de Nom à 8 lettres - groupe de Nom à 10 lettres etc ... comme indiqué sur l'exemple.

Je pense que cette nouvelle structure permettrait toujours

une écriture séquentielle de fichier avec accès direct pour chaque client :

- je tape une nouvelle donnée colonne 1 (un nouveau client) l'ordre croissant porte toujours sur le 1er octet gauche du Nom : exemple MARTIN et je l'inclue à la suite de mon dernier client dont le nom commence par M, si j'ai un autre client à ajouter qui se nomme DURAND, je procède de la même façon et je l'inclue à la suite de mon dernier client dont le Nom commence par D.

Mon Fichier a augmenté de 2 clients mais l'ordre reste toujours séquentiel avec accès direct pour chaque client. On peut donc avoir une écriture de Fichier séquentielle avec accès direct à chaque client, la porte étant toujours le premier octet gauche du Nom, chaque Nom se codant systématiquement de façon différente à sa frappe.

Avec cette structure il n'est plus nécessaire de laisser de la place à la fin de chaque bloc (partie non adressable) ou de faire des Fichiers d'Extension pour inclure les nouveaux clients, Fichiers qu'il faut ensuite fusionner.

La place mémoire occupée pour un Fichier de 1000 Clients avec cette nouvelle structure serait :

- 26 octets pour Nom et prénom
- 24 octets pour adresse
- 1 octet 1/2 pour code Pays qui est aussi le code monnaie de facturation

soit au total pour un client : 51 octets 1/2

- pour 1000 clients = 51500 octets soit 103000 enregistrements soit en blocs au total = 103000 : 2048 = 50 arrondi à 51 blocs.

Avantage : écriture toujours séquentielle avec accès direct pour chaque client, possibilité de création ou de suppression de client sans lire tout le fichier, la clé d'accès étant toujours la 1ère lettre ou octet gauche de chaque Nom, cette 1ère lettre donne aussi le Rang correspondant; de ce fait, on se positionne directement sur l'information que l'on désire supprimer ou ajouter.

En ce qui concerne les codes articles, ce sont les mêmes possibilités. La frappe de chaque article engendre son propre code, pour les articles d'une même gamme c'est la dimension de l'article qui crée la différence de code exemple :

- LEGUMIER ROND 20
55744959 9654 20

LEGUMIER ROND 23
35744959 9654 23

PLAT OVALE 26
7313 65135 26

5 PLAT OVALE 28
7313 65135 28

place mémoire gagnée en supprimant ces codes superflus :

- pour 1000 clients : 4 octets X 1000 = 4000 octets
- pour 1000 articles: 3 octets X 1000 = 3000 "

10 écriture séquentielle et accès direct pour Fichier Clients
comme pour Fichier articles avec cette nouvelle structure
et, comme tout s'enchaîne, les utilisateurs de ces Fichiers
n'auront plus de codes clients ou codes articles à ajouter
lors du traitement des commandes. Etant donné que la
15 codification est le genre de travail fastidieux et répétitif
qui entraîne bien souvent des erreurs qui coutent très
cher exemple :

- une erreur de code article et le Client reçoit des bols
alors qu'il a commandé des assiettes; si l'erreur porte
sur une quantité importante de cartons, c'est, soit un retour
20 de marchandise soit un Avoir important si l'article reçu
par erreur est mal vendu dans le Marché du Client. On peut
imaginer le coût de telles erreurs dans un Service Export
où les commandes sont toujours importantes et les distances
de livraison considérables.

25 Donc gain de place mémoire, performance accrue dans
la création et la recherche de l'information, dans le tri,
en somme, gain de temps et d'argent à tous les niveaux
en supprimant les codes Clients et codes articles qui est
une codification superflue. Le principe étant le même on
30 pourrait aussi appliquer cette nouvelle structure sur d'autres
sortes de Fichier tels que Fichiers tarifs, tenue des stocks,
Agents etc...

Je pense qu'il est possible d'appliquer ce nouveau
système de traitement de l'information d'ores et déjà dans
35 les Entreprises, dans les grands Magasins de Vente par
correspondance et dans les Banques. Puis, par la suite, par
tout utilisateur du Système Informatique.

REVENDEICATIONS

1. Nouveau ^{système de} traitement de l'information, constitué par la prise directe des données en machine en colonne 1, caractérisé par le fait qu'il supprime la codification actuelle de l'information et la remplace par le code et le rang engendrés systématiquement en machine à la perforation de chaque lettre de l'alphabet démontré par la construction du tableau Alphabétique pour Ordinateur.

2. Nouveau système de traitement de l'information selon la revendication 1, caractérisé par le fait qu'il implique une égalité directe entre notre langage courant (l'information) et le traitement de ce langage en machine et crée en même temps un rang qui permet un accès direct à chaque information, la porte d'accès ou clé étant le 1er octet ou lettre alphabétique gauche du Nom de client ou de l'article gamme, en colonne 1.

3. Nouveau système de traitement de l'information selon la revendication 1, caractérisé en ce que son application peut être faite sur le système actuel Informatique sans entraîner aucun changement du matériel.

4. Nouveau système de traitement de l'information selon la revendication 2, caractérisé par le fait qu'il constitue un principe de base et qu'il peut être appliqué à tous les Fichiers qui permettent le même principe.