Un système d'indexation automatique

par Dominique Maniez

Date de publication : 03 octobre 2009

Dernière mise à jour :

I - Introduction

Le programmeur de base de données en lisant le titre de cet article va être induit en erreur. En effet, le terme d'indexation automatique fait référence, dans le cas présent, à une technique documentaire et non pas à la commande qui permet d'indexer un champ d'une table (bien que ces deux opérations représentent des similitudes conceptuelles, indexer, signifiant étymologiquement, montrer du doigt, c'est-à-dire identifier en vue d'une reconnaissance rapide, attirer l'attention sur).

L'indexation pour un documentaliste consiste à repérer les mots les plus significatifs d'un texte (les descripteurs) afin que ceux-ci servent de clefs d'accès. Regroupé au sein d'une collection, le document ainsi indexé pourra aisément être retrouvé grâce à une interrogation sur un de ses mots-clefs. De la même manière, les bons ouvrages comportent un index des termes importants permettant ainsi un accès rapide à l'information.

Les techniques d'indexation en documentation, qui font partie de la technique du catalogage, sont des techniques très complexes qui obéissent à des règles et des normes maintenant bien codifiées. Le problème le plus crucial reste néanmoins celui de l'indexation matière, c'est-à-dire le fait de trouver les descripteurs d'un texte. Cette opération qui fait appel à l'intelligence et à la culture du documentaliste est une opération très coûteuse en temps humain et difficilement formalisable. Cependant de nombreux chercheurs, avec l'accroissement de la puissance des ordinateurs se sont penchés sur la possibilité d'automatiser cette indexation.

Nous allons transposer le problème des documentalistes dans un cadre informatique et essayer de trouver une solution dont nous dévoilerons uniquement l'algorithme, laissant au lecteur le choix de coder cet algorithme dans le langage qui lui convient.

Il nous arrive souvent d'avoir à traiter des bases de données ayant des champs textuels relativement importants par leur taille (de l'ordre de 2000 caractères) et leur contenu ; un problème immédiat se pose : comment retrouver une information stockée dans un champ caractère ou bien dans un champ mémo ? Les techniques prévues par les SGBD ne sont guère satisfaisantes car on ne peut pas indexer un champ mémo, ni un champ caractère si celui-ci est trop long. De plus, l'indexation d'un champ caractère ne sera valable que si la recherche a lieu sur les premiers caractères de ce champ, ce qui est rarement le cas dans la pratique.

Il existe cependant des fonctions qui permettent la recherche d'une sous-chaîne dans une chaîne de caractères, mais leur emploi est trop pénalisant sur de gros volumes de données. Devant toutes ses limitations, il faut donc envisager un autre système si l'on veut exploiter efficacement les champs textuels. La seule solution consiste à créer un index (au sens documentaire du terme), c'est-à-dire la liste des mots-clef du champ caractère. Cette opération technique nécessite normalement l'intervention d'un documentaliste qui emploie des règles bien établies faisant appel à l'analyse textuelle. Cette hypothèse nous semblant irrecevable dans le cadre de nos applications, il nous faut alors envisager une autre solution plus économique en temps, d'où l'idée d'un index documentaire automatique.

Les linguistes ont depuis longtemps découvert deux particularités intéressantes de tout document textuel :

1. tout texte contient les descripteurs nécessaires à sa propre indexation.
2. un texte est formé pour moitié de mots grammaticaux qui en eux-mêmes n'ont aucun sens

tout texte contient les descripteurs nécessaires à sa propre indexation.

La solution proposée ici consiste donc à extraire tous les mots d'un champ caractère. Évidemment, cette technique est gourmande en mémoire disque, mais étant donné la taille actuelle des disques, ce facteur n'est pas trop pénalisant. D'autre part, afin de réduire la taille de notre index et d'augmenter sa pertinence, nous allons exclure de notre index, toute une série de termes qu'on appelle mots vides : on retrouve dans cette liste les articles, les pronoms, les conjonctions, les prépositions, certains adverbes, etc.

II - Mots vides

Les programmeurs qui débutent en informatique documentaire ont souvent du mal à constituer leur liste de mots vides et ils peuvent dans ce cas-là s'inspirer de celles que l'on trouve sur Internet. Il en existe également une autre qui est à portée de main et que bon nombre d'utilisateurs ignorent totalement : celle qui est fournie par Windows. En effet, peu de gens le savent, mais Windows inclut une fonctionnalité d'indexation des documents qui se trouvent sur notre disque dur. Il existe ainsi un service de Windows baptisé Service d'indexation. Si l'on prend la peine de consulter l'aide en ligne de ce service, on verra que Windows exclut de ses index tout une série de mots dont voici la liste :

a	afin	ai	aie	aient	aies
ait	alors	après	as	au	aura
aurait	auraient	aurais	auras	aurez	auriez
aurions	aurons	auront	autant	aussi	autre
autres	aux	auxquelles	auxquels	avait	avaient
avais	avant	avec	avez	aviez	avions
avoir	avons	ayant	ayez	ayons	beaucoup
bien	car	ce	ceci	cela	celle
celles	celui	cependant	certains	certaines	ces
cet	cette	ceux	chez	clic	comme
comment	concernant	dans	de	depuis	des
devaient	devais	devait	devant	devez	deviez
devions	devoir	devons	devra	devrai	devraient
devrais	devrait	devras	devrez	devrions	devrons
devront	doit	dois	doive	doivent	doives
donc	dont	du	dû	elle	elles
en	es	et	eu	eue	entre
eues	eux	jusque	la	le	a b c d e
il	ils	les	leur	leurs	f g h i j
lorsque	lui	là	ma	me	k l m n o
moi	mon	mes	ne	notre	p q r s t
nous	ont	or	ou	en outre	u v w x y
où	par	parce que	pas	peut	z $
peuvent	peux	plus	pour	hors	1 2 3 4 5
pourras	pourra	pourrai	pourraient	pourrais	6 7 8 9 0
pourrait	pourrez	pourriez	pourrions	pourrons	
pourront	pouvaient	pouvait	pouvais	pouvant	pouvez
pouviez	pouvions	pouvoir	pouvons	pu	puis
puisque	puisses	puisse	puissent	puissiez	puissions
quand	quant	que	quel	quelle	quelles
quels	qui	quoi	sa	se	sera
serai	seraient	serais	serait	seras	serez
seriez	serions	serons	seront	ses	si
sien	sienne	siennes	siens	soient	soit
sois	sommes	son	sont	sous	soyez
soyons	suis	sur	ta	te	tel
telle	telles	tels	tes	tien	tienne
tiennes	tiens	toi	ton	tous	tout
toute	toutes	tu	un	une	vers
veuille	veuillent	veuilles	veuillez	veuillons	veulent
veut	veux	vos	votre	voudra	voudrai
voudraient	voudrais	voudrait	voudrais	voudras	voudrez
voudriez	voudrions	voudrons	voudront	voulaient	voulais
voulait	voulant	voulez	vouliez	voulions	vouloir
voulons	voulu	voulue	voulus	voulues	vous
y	à	étaient	étais	était	étant
étiez	étions	êtes	être	les deux	venir
viens	vient	venons	venez	viennent	venu
venus	venue	venues	vienne	viennes	venions
veniez	viennent	faire	fais	fait	faisons
faites	font	fasse	fasses	fassions	fassiez
fassent	fait	faite	juste	pareil	la plupart
jamais	maintenant	désormais	re	de nouveau	dire
dis	dit	disons	disent	dites	dise
dises	disions	disiez	voir	vois	voit
voyons	voyez	voient	voie	voies	voyions
voyiez	vu	vue	encore	prendre	prends
prend	prenons	prenez	prennent	prenne	prennes
prenions	preniez	pris	prise	haut	manière

Le choix des termes composant la liste des mots vides est très important et il est souvent préférable de laisser la possibilité à l'utilisateur de la base de données de modifier lui-même cette liste. En effet, dans le cas, par exemple, d'une base de données juridiques, les termes décret, arrêté, loi seront tellement utilisés qu'on retrouvera leur occurrence dans chaque enregistrement de la base de données ; il conviendra donc d'incorporer ces termes à la liste des mots vides. Un autre problème réside dans le fait que les textes sont saisis soit en minuscules soit en majuscules ; ainsi la recherche pourra ne pas aboutir si l'utilisateur ne frappe pas son critère de recherche exactement comme dans le texte saisi. La solution consiste donc à indexer tous les champs en majuscules et à convertir le critère de recherches en majuscules.

III - Algorithme de notre solution

Il faut d'abord découper le champ caractère ou le champ mémo en mots unaires. Nous considérons qu'un mot est séparé par un espace ou un signe de ponctuation. Nous stockons cette liste dans une table intermédiaire, intitulée TERMES dont la structure est la suivante :

MOT : Caractère 20

Afin d'accélérer les traitements, la liste des mots vides doit être chargée dans un tableau.

Voici l'algorithme de découpage de notre variable vtexte qui est une copie d'un champ caractère ou d'un champ mémo d'une table d'un SGBD quelconque. L'algorithme se borne à découper la variable en mots simples et à ajouter ces mots dans la table TERMES en ne stockant que les mots qui n'appartiennent pas à la liste des mots vides.

```
longtexte=len(vtexte)
initiale=1 && position de l'initiale du mot dans la chaîne vtexte
parcours=1 && variable numérique qui balaye la chaîne vtexte
ponctuation = " .;,?:!()[]'-&¿¡-"+ chr(34) && code du guillemet
ouverture de la table TERMES
do while parcours < longtexte
     if substr(vtexte,parcours,1) est un signe de ponctuation
  * on stocke le mot s'il est non vide
  vmot=substr(vtexte,initiale,parcours-initiale)
          if majuscule(vmot) n'appartient pas à la liste des mots vides
    on ajoute vmot à la table TERMES
    endif
  * on va se positionner sur le prochain caractère
  * cas où plusieurs espaces ou signes de ponctuation de suite
  do while substr(vtexte,parcours,1) est un signe de ponctuation
       parcours=parcours+1
  enddo
  initiale=parcours && un nouveau mot commence
  else && pas de ponctuation
  parcours=parcours+1
  endif
enddo
```

Une fois cette liste établie, il convient d'enlever les doublons de la table TERMES.

On ajoute ensuite cette liste dans la table MOTSCLEFS en y ajoutant la référence de l'enregistrement. Il est préférable de stocker les mots en majuscules afin de faciliter la recherche.

La table des mots-clefs a la structure suivante :

MOTCLEF : Caractère 20

NUMFICHE : Numérique 5

NUMFICHE est un pointeur sur l'enregistrement qui contient le mot clef.

Lors de la recherche, l'utilisateur saisit un terme ; le programme effectue une recherche dans la table MOTSCLEFS et récupère tous les pointeurs vers les enregistrements qui contiennent le terme recherché. La recherche est immédiate et l'on n'a plus à balayer séquentiellement chaque enregistrement.

IV - Limites et améliorations de notre solution

La principale limitation est inhérente à l'algorithme de découpage des mots qui ne prend pas en compte les mots composés, la recherche devant ultérieurement s'effectuer sur des termes unaires. Si cette limitation est connue de l'utilisateur, cela ne pose pas trop de problèmes.

Plusieurs améliorations peuvent cependant être envisagées : on peut contrôler l'indexation à l'aide d'un thésaurus qui est une liste de termes admissibles pour l'indexation. On peut également mesurer la fréquence d'apparition d'un terme dans le texte et la comparer avec une table de fréquence moyenne mesurée sur un grand échantillon de textes. Si la fréquence mesurée est supérieure à la fréquence moyenne, ce terme sera un bon descripteur. Enfin, l'analyse syntaxique d'un texte permet de reconnaître les noms des adjectifs, d'identifier les formes verbales et ainsi d'affiner grandement l'indexation.

V - Conclusion

Il est clair que certaines améliorations proposées plus haut sont difficiles à mettre en □uvre, notamment l'analyse morpho-syntaxique d'un texte. D'autre part, une indexation humaine sera toujours meilleure qu'une indexation automatique. Cependant, le système proposé dans ces colonnes se révèle à l'utilisation très performant.

J'espère que cet article vous aura donné le goût de l'indexation automatique et vous aura ouvert certaines perspectives de l'informatique documentaire. En guise d'application, nous proposerons prochainement une implémentation de cet algorithme dans une base Access.