Vers un passage à l'échelle dans un SGBD d'images

José Martinez — **Patrick Valduriez**

Équipe ATLAS, INRIA & LINA (FRE CNRS 2729)
2, rue de la Houssinière – B.P. 92208 – 44322 Nantes Cedex 03
{prénom.nom}@lina.univ-nantes.fr

RÉSUMÉ. *Un système de gestion de bases d'images doit* a priori *s'appuyer sur un système de gestion de bases de données (SGBD). Dans cet article, nous examinons expérimentalement les limitations des SGBD relationnels*[1]*. Nous identifions ainsi un certain nombre d'écueils et apportons aussi des réponses, pour certaines encore partielles. Cinq points clés ont été identifiées : réduction physique de la taille des métadonnées, introduction d'informations redondantes, exploitation du parallélisme, classification et répartition des données, indexation physique différenciée.*

ABSTRACT. *An image database system should use a database management system (DBMS). In this paper, we experiment relational DBMS limitations for such a purpose. We identify a number of pitfalls and provide some solutions too. Some are still partially supported. However, five key points have been identified so far: reducing the metadata size, introducing some redundant data, taking advantage of parallelism, classifying and distributing metadata, using different indexing schemes.*

MOTS-CLÉS : *Base d'images, passage à l'échelle, réduction des données, parallélisme.*

KEYWORDS: *Image database, scale-up, data reduction, parallelism.*

1. Ce travail a été partiellement financé par le projet MDP2P dans le cadre de l'ACI « Masses de Données ».

1. Introduction

Les recherches concernant les bases d'images ont porté sur l'extraction de descripteurs pertinents, l'utilisation de distances appropriées, l'ajout de descriptions « sémantiques » pour faire face au fossé entre les descripteurs numériques et le sens commun, éventuellement la prise en compte du principe d'interrogation par rétro-action de la recherche d'information, voire de la navigation hyper-textuelle. Si nombre de questions restent ouvertes, un certain nombre de résultats permettent d'ores et déjà d'envisager l'exploitation de bases d'images pour différents besoins, soit immédiatement, soit à terme :

– la recherche d'images disséminées sur le disque dur comme utilitaire,

– l'album de famille pour le grand public,

– l'indexation des couvertures (et à terme de toutes les illustrations) dans les bibliothèques numériques et pour les librairies en ligne,

– enfin, le besoin de recherche d'images similaires apparaît dans un cadre ouvert comme la Toile (recherche de détournement de logos, par exemple) où les moteurs de recherche n'utilisent pour le moment que l'information textuelle à proximité des images.

Du point de vue des systèmes de gestion de bases de données (SGBD), le cas des applications familiales n'est pas complexe. L'ordre de grandeur de la base sera de quelques milliers à dizaines de milliers d'images. Ce sont plutôt les techniques de reconnaissance de formes qui sont à améliorer, les utilisateurs souhaitant retrouver les images qu'ils ont prises du « petit dernier », des « vacances à la neige », etc.

Pour les applications professionnelles, l'usage de données multimédias implique de grands volumes, de l'ordre du million à la dizaine de millions d'images, même si peu de fournisseurs d'images peuvent se vanter d'offrir un tel éventail d'images à l'heure actuelle. Corbis™ dispose de quelques 40 millions d'images, indexées majoritairement par mots clés et par catégorisation et très grossièrement par le contenu (orientation de l'image – horizontal, vertical – et principales couleurs – rouge, vert, etc. –). Google™ revendique l'indexation de 425 millions d'images, mais indexées par le texte environnant et non par le contenu.

Plusieurs approches sont candidates à l'exploitation de bases d'images massives :

– les systèmes propriétaires, capables de s'adapter à la nature particulières des images,

– les SGBD relationnels, cheval de bataille de l'industrie,

– les SGBD natifs XML, susceptibles de manipuler plus aisément des données pour lesquelles le standard MPEG-7 a été créé [NAC 99a, NAC 99b].

Dans ce travail, nous avons choisi d'explorer la deuxième approche. Soulignons tout d'abord que nous nous plaçons dans l'optique de l'intégrateur de solutions et non dans le rôle d'un fournisseur de SGBD dédié. Contrairement aux modèles propriétaires, il est aisé de mixer plusieurs types de données, notamment des données

textuelles pour fournir de la sémantique aux images sous la forme de mots clés, au minimum, et d'informations plus structurées, mais aussi d'autres données numérisées pour aboutir, à terme, à une véritable SGBD multimédia. Contrairement aux modèles basés sur XML, les SGBD relationnels sont matures, disponibles, optimisés et susceptibles d'autres améliorations ou d'extensions, dont les approches objets-relationnelles [DAT 98]. Bien sûr, utiliser des sur-couches nuit gravement aux performances.

Mais, les SGBD relationnels ont été optimisés pour des applications utilisant des données simples, pouvant être indexées efficacement. Les applications multimédias imposent de nouvelles contraintes [KLA 95, CHA 98]. Les descripteurs multimédias sont complexes : histogrammes, transformées, etc. Le but est donc de trouver un compromis entre la finesse des descripteurs mathématiques pertinents et les performances élevées attendues des applications multimédias utilisant un SGBD [CHE 97].

Dans le travail décrit dans cet article, nous nous sommes intéressés aux difficultés prévisibles, en les quantifiant, lorsque l'on décide de stocker une base d'images dans un SGBD *relationnel* (SGBDR). Au-delà des constats, des éléments de solution sont apportées ; d'autres éléments de réflexion restent plus ouverts. En tout état de cause, une solution semble devoir être un assemblage de plusieurs facteurs indépendants d'amélioration des performances. Nous en avons déjà identifié cinq dont les trois premières ont été expérimentées :

1) le *compactage* et/ou la *réduction* des dimensions de la taille des métadonnées est un élément extrêmement important des performances d'un SGBD puisque ce dernier manipule des informations stockées sur disque [CHE 97, KAN 99, MOO 01] ;

2) la matérialisation d'informations redondantes permet d'indexer ces dernières, donc d'accélérer certaines requêtes ;

3) le *parallélisme* est un outil indispensable pour diminuer les temps de réponse *in fine*, en complément de toutes les autres approches [KAN 96] ;

4) le placement de classes d'images sur diverses machines permet de mieux paralléliser les requêtes en n'exploitant que la puissance nécessaire, en diminuant les temps de consolidation des résultats, en offrant un véritable parallélisme entre requêtes accédant à des données disjointes ;

5) l'indexation – multi-dimensionnelle – physique différenciée.

Cette article est structuré en quatre parties. Dans la section 2, nous introduisons le modèle des métadonnées qui décrivent les images ainsi que les requêtes qui ont servi à étalonner et comparer les deux SGBDR utilisés. Dans la section 3, nous présentons les conditions d'expérimentations et les performances des différentes exécutions d'insertion dans les SGBDR retenus. La problématique du « passage à l'échelle » nous amène, dans la section 3, à paralléliser les requêtes sur une grappe de machines. Enfin, nous récapitulons les conclusions de cette étude expérimentale qui ouvre de nombreuses voies de recherche.

2. Modèle expérimental

Pour conduire des expérimentations, nous avons opté pour un schéma de description des images simple, largement exploité dans la littérature, mais qui permet néanmoins de mettre en évidence des problèmes sérieux. L'indexation du contenu des images se fait essentiellement *via* des histogrammes de couleurs.

L'interrogation s'appuie largement sur SQL (*Structured Query Language*). En effet, les performances des SGBDR sont catastrophiques pour des traitements ponctuels répétés. Par ailleurs, si la majeure partie du travail venait à être réalisée à l'extérieur du SGBD, ce dernier serait relégué au rôle passif de stockage.

2.1. *Modèle de métadonnées*

Les métadonnées associées à des images sont nombreuses, ouvertes, extensibles, partiellement standardisées par MPEG-7 [NAC 99a, NAC 99b]. Le schéma relationnel retenu est illustré en figure 1 où les attributs soulignés forment la clé de chaque relation. Détaillons les points clés de ce schéma[1].

La relation *HistogrammesRégions* est un exemple de données complexes puisqu'elle représente un ensemble d'images dont le contenu est traduit par un ensemble de régions décrites chacune par trois histogrammes, qui sont des vecteurs d'entiers naturels. La clé de la relation est donc composée de l'identifiant de l'image, de celui de la région, du canal dans un espace de couleur donné (RVB, L*a*b*...) et de l'index dans l'histogramme correspondant. Les régions peuvent être obtenues par segmentation de l'image ou déterminées par découpage heuristique de l'image (centre / contour, points-forts / bords, etc., voire totalité de l'image) [MAR 00].

Les relations *MomentsInertie* et *Couleurs* sont des exemples de réduction *et* de passage forcé en première forme normale (1FN).

Les deux relations sont des approximations de plus en plus grossières de la relation *HistogrammesRégions*. Toutes deux décrivent les couleurs de la *totalité* de l'image. La relation *MomentsInertie* décrit encore les trois canaux indépendamment. La relation *Couleurs* va plus loin dans la réduction en introduisant des couleurs perceptuelles, sans dimension, qui couvrent la totalité de l'espace de couleurs à trois dimensions.

Dans les deux cas également, les descriptions ont été forcées en 1FN. Il aurait été possible d'ajouter le numéro d'un moment d'inertie en tant qu'attribut supplémentaire, ce qui donnerait le schéma de relation *MomentsInertie*(IID, canal, moment, valeur). De même, le schéma de la relation *Couleurs* serait (IID, couleur, valeur). Nous verrons en section 3.2.3 quels sont les avantages du schéma retenu en figure 1.

1. D'autres relations ont été créés pour les métadonnées structurées (auteur, date de prise de vue...), le type d'images (couleurs, niveaux de gris ; naturelle ou graphique), les trois couleurs dominantes de l'image, etc. Celles retenues ici suffisent à notre propos.

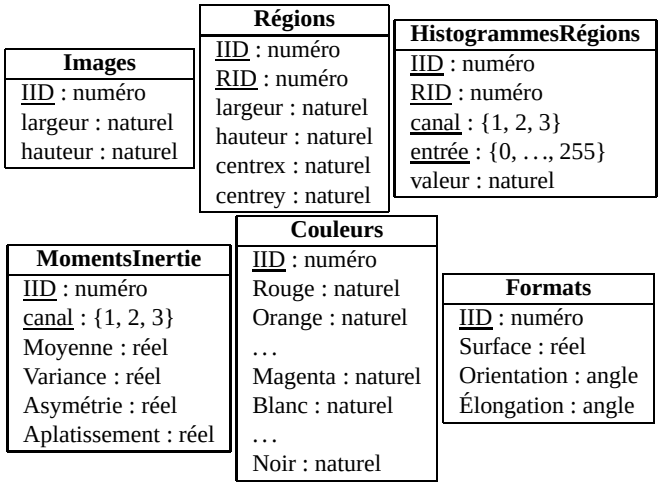

Figure 1. *Schéma relationnel d'une base d'images*

Une instance de ce schéma de base de données est créée par un processus aléatoire. En effet, les expérimentations vont porter sur 1 000, 10 000 et 100 000 images avec un objectif initial de 1 000 000 d'images. De plus, nous souhaitions pouvoir contrôler les lois de répartition des descripteurs d'images. Enfin, les temps d'extraction des descripteurs, non négligeables, seraient venus s'ajouter à ceux des insertions.

2.2. *Requêtes type*

Les requêtes que l'on doit envisager dans un SGBD d'images sont variées. Nous en retenons deux, outre les jointures, qui posent une difficulté notable, au sens des performances.

2.2.1. *Requêtes « sémantiques »*

Les utilisateurs d'un SGBDR manipulent *a priori* des informations structurées. On peut être amené à créer des vues au-dessus des informations de base, afin de faciliter la rédaction des requêtes. Prenons l'exemple du *format* des images. L'information est portée par les attributs *largeur* (l) et *hauteur* (h). Mais plusieurs informations sont intéressantes à dériver : par exemple, la surface $s = l \times h$, l'orientation $o = \frac{1-\cos(2\alpha)}{2}$ ou l'élongation $e = \left|\frac{4\alpha}{\pi} - 1\right|$ avec $\alpha = \arctan\frac{l}{h}$. Ces données peuvent être rendues perceptuelles en les discrétisant[2]. L'intervalle $[1, +\infty[$ de la surface peut être découpé en sous-intervalles auxquels on associera les termes *minuscule*, *petite*, *moyenne*, *grande*, *papier peint*, *immense*. Pour l'orientation, comprise dans $]0, \frac{\pi}{2}[$, on pourra distinguer

2. En particulier, les variables linguistiques de la logique floue peuvent être employées.

Nom	Processeur	Mémoire centrale	Mémoire secondaire	Système d'exploitation	SGBD relationnel
PG–	Pentium 133 MHz	128 Mo	ATA 33 5 000 tr/min	Linux Mandrake 8.2	PostgreSQL 7.3
ORA	bi-Céléron 500 MHz	512 Mo	ATA 66 7 500 tr/min	Windows 2000	ORACLE 9*i Personal*
PG	bi-Céléron 500 MHz	512 Mo	ATA 66 7 500 tr/min	Linux Mandrake 9.0	PostgreSQL 7.3
PG+	bi-Pentium III 800 MHz	1 Go	ATA 100 7 200 tr/min	Linux Red Hat 9.0	PostgreSQL 7.3

Tableau 1. *Configurations matérielles et logicielles utilisées*

les images *paysage*, *carrée* et *portrait*. Enfin, l'élongation nous permet de proposer *aucune*, *standard*, *panoramique*, *allongée*, voire *barre*.

L'intérêt de ces informations redondantes est qu'*en cas de matérialisation de l'information dérivée*, aussi bien numérique que linguistique, des index peuvent être mis en place afin d'accélérer les recherches. Pour les valeurs numériques, les arbres-B classiques suffisent ; pour les valeurs linguistiques, des index *bitmap* peuvent être employés. Certains SGBD, c'est le cas de PostgreSQL, sont capables de créer des index sur des fonctions, ici $largeur \times hauteur$, ce qui permet de créer l'index sans avoir à matérialiser la relation. Mais la rédaction et la reconnaissance de formules complexes comme celle de l'élongation est plutôt aléatoire.

Il faut souligner qu'il n'est pas possible de créer des index sur les informations de base en vue de répondre efficacement aux mêmes requêtes. En effet, les transformations ayant produit s, o et e utilisent *deux* attributs, combinés de manière non-linéaire de surcroît. Ainsi, indexer la largeur et la hauteur ne permet pas d'accélérer les recherches de sélection des images panoramiques. Sur une requête, cela se traduit par la possibilité, ou pas, d'effectuer une comparaison de la forme $variable\ \Theta\ constante$. L'opérateur Θ est soit l'égalité, et par généralisation l'appartenance à un ensemble de constantes, ce qui permet d'utiliser le hachage et les index *bitmaps*, soit les diverses inégalités, ce qui nécessite d'utiliser des arbres-B car ils ordonnent les valeurs des clés.

2.2.2. *Calculs de distances*

Dans les systèmes de recherche d'images par le contenu, *la* requête typique est celle qui calcule une distance entre une image exemple et l'ensemble des images de la base. Une distance simple mais commune sur des histogrammes de couleurs est la distance euclidienne $L_2(h, h') = \sqrt{\sum_{\forall i}(h_i - h'_i)^2}$ que l'on étend à deux vecteurs d'histogrammes en calculant la moyenne arithmétique des distances appliquées deux à deux. Elle est facilement traduisible en SQL.

configuration	entrées d'histogrammes par image	temps d'insertion (10 000 images)	temps d'insertion (100 000 images)
PG–	12	0 h 10 m	— h — m
ORA	12	0 h 20 m	3 h 00 m
ORA	120	— h — m	11 h 00 m
ORA	480	10 h 30 m	— h — m
PG	12	0 h 08 m	1 h 40 m
PG	120	— h — m	5 h 00 m

Tableau 2. *Insertions des métadonnées (aléatoires) dans le SGBD*

3. Expérimentations sur mono-processeurs

Les expérimentations ont été réalisées sous plusieurs configurations matérielles et logicielles, nommées et présentées dans le tableau 1. Dans les deux sections qui suivent, nous décrivons tout d'abord les évaluations de performances dans un cadre centralisé puis nous en tirons des conclusions d'une portée générale.

3.1. *Étalonnage*

3.1.1. *Insertions*

La première expérience est l'alimentation de la base en métadonnées. Pour chaque image, ce sont tous les n-uplets du schéma relationnel de la figure 1 qui sont générés et insérés[3]. Le tableau 2 présente les temps observés lors de l'insertion de ces métadonnées. Les index sur les clés primaires ont été désactivés pour accélérer ces insertions massives d'environ 4 Go de données.

Le tableau 2 confirme clairement que la puissance de calcul n'est pas un facteur prépondérant dans l'évaluation des performances d'un SGBD, les temps d'insertion comparés des configuration PG–/12 et PG/12 étant plus proches du ratio observés pour les disque respectifs que pour les processeurs.

De plus, ce tableau montre que le respect de la première forme normale pour gérer des données volumineuses n'est pas réaliste. Les temps d'insertion exhibés par ORA/480 (5 régions par image, 3 canaux par région et 32 entrées d'histogrammes par canal) sont déjà prohibitifs. La configuration 120 (5 régions, 3 canaux et 8 entrées) dégrade significativement le rendu des images. La configuration 12 (1 région – la totalité de l'image –, 3 canaux et 4 entrées) sert à montrer, dans les tableaux suivants, que même les représentations les plus dégradées souffrent encore de mauvaises performances.

3. Et plus (*cf.* la note de bas de page numéro 1) !

.

configuration	1 000 images	10 000 images	100 000 images
ORA	389 ms	1 000 ms	7 000 ms
PG	282 ms	422 ms	2 000 ms
PG+	47 ms	165 ms	—

Tableau 3. *Performances du parcours séquentiel de la relation* Images

configuration	100 images	1 000 images	10 000 images	100 000 images
ORA/12	29 s	38 s	2 m 12 s	20 m
PG/12	5 m 34 s	10 m 06 s	1 h 12 m	10 h
ORA/120	1 m 34 s	3 m 02 s	18 m	3 h
PG/120	15 m	1 h	10 h	—

Tableau 4. *Performances de la jointure de toutes les relations*

3.1.2. *Parcours séquentiels*

Candidement, les temps de parcours des relations ont été évalués. Le tableau 3 donne les résultats pour la requête `select * from Images`.

La conclusion qui s'impose est que dès que l'on dépasse quelques dizaines de milliers de n-uplets (ou plus exactement l'équivalent en blocs sous-jacents sur le disque) ce type de parcours ne peut plus être employé dans une recherche interactive. On dépasse le seuil psychologique de la seconde.

3.1.3. *Jointures*

Dans une requête, il apparaît le plus souvent une ou plusieurs jointures. Ici, nous nous plaçons dans le cas symptomatique où une requête exploiterait toutes les métadonnées et devrait donc joindre toutes les relations du schéma de la figure 1. Le tableau 4 donne donc les les plus mauvais résultats auxquels on peut s'attendre. Ils ont été obtenus sans utiliser d'index, la stratégie des différentes jointures étant toujours le parcours parallèle de relations triées. La complexité de la requête est dominée par la cardinalité de la relation *HistogrammesRégions*. Il ne faut donc pas s'attendre à des résultats nettement améliorés dès lors que l'on exploite les histogrammes. De même, l'activation des index n'améliore les résultats que lorsque le nombre de données est faible.

Encore une fois, l'utilisation de métadonnées trop fines dans les requêtes SQL est désastreuse pour les performances.

3.1.4. *Calculs de distances*

Les deux raisons précédentes se combinent dans le cas de la requête de calcul de la distance euclidienne. Le tableau 5 en témoigne. Par ailleurs, il apparaît ici nettement que l'emploi des index sur les clés est préjudiciable aux performances. Les parcours séquentiels bénéficient du pré-chargement des pages successives sur le disque,

(a)

configuration	100 images	1 000 images	10 000 images	100 000 images
ORA/120	47 s	51 s	1 m 29 s	11 m 43 s
PG/120	1 m 21 s	1 m 33 s	3 m 56 s	28 m 00 s

(b)

configuration	100 images	1 000 images	10 000 images	100 000 images
ORA/12	31 ms	328 ms	3 s	35 s
ORA/120	2 s	15 s	16 m	1 h 40 m
ORA/480	1 m	10 m	55 m	3 h
PG/12	44 s	45 s	52 s	124 s
PG/120	1 m 21 s	1 m 33 s	3 m 56 s	28 m
PG+/120	2 s	7 s	82 s	—

Tableau 5. *Performances du calcul de la distance euclidienne (a) sans index et (b) avec index*

ce qu'aucun facteur de sélectivité ne permet de compenser lors des descentes aléatoires dans les pages des arbres-B.

Une étude détaillée du plan d'exécution donne une complexité dominée par le terme $O(|H|.\log_2|H|)$. Mais les termes de rang inférieur ne sont pas négligeables et sont dotés de mauvaise constantes multiplicatives qui traduisent de nombreux cycles de lectures / écritures de relations entières sur le disque.

Il faut souligner qu'une complexité en $O(n.\log n)$ est le maximum supportable lorsque les données sont sur un disque [SKI 97]. Mais, ce qui n'est pas mauvais pour un traitement en lot, reste inacceptable pour des recherches interactives. Pour ces applications-là, l'objectif à atteindre est une complexité qui ne dépendrait de la taille de la réponse, $n' \ll n$.

3.2. *Conclusions*

De toute évidence, l'implémentation naïve d'un schéma de base de données d'images dans un SGBDR est inutilisable en dehors de jeux d'essais très réduits.

3.2.1. *Occupation de la mémoire relationnelle*

D'après les performances relevées, la réduction de la taille des métadonnées est un facteur *extrêmement* important, *du point de vue pratique*, de l'amélioration des performances.

Malheureusement, il faut noter que la contrainte de 1FN ne permet pas de compacter autant qu'on le souhaiterait les métadonnées. Par exemple, un histogramme mono-dimensionnel pourrait être stocké en utilisant seulement $256 \times 4 = 1\,024$ octets. En reprenant le schéma et le principe de stockage d'ORACLE, l'occupation mémoire correspondante est de 4 864 octets, soit près de cinq fois plus. Dans les deux configu-

rations effectivement expérimentées, 120 et 480, l'occupation totale des histogrammes pour une image composée de cinq régions s'élèvent déjà à 9 120 octets et 2 280 octets.

En revanche, *du point de vue théorique*, la réduction de la taille des métadonnées ne constitue qu'une amélioration de la *constante multiplicative* dans la complexité des requêtes.

Par conséquent, cette optimisation est entièrement à la charge du SGBD utilisé. Lors de la création des relations, il suffit de veiller à utiliser les types de données les plus adaptés du SGBD et à ne pas perdre de place dans l'occupation des blocs sur les disques. En effet, les métadonnées sont utilisées en écriture, pour alimenter la base, puis en lecture, pour répondre aux requêtes, mais pas en modification ; il n'y a donc pas de risque de débordement des blocs.

3.2.2. *Réduction de la taille des métadonnées*

Pour réduire encore la consommation d'espace sur les disques et donc les temps de réponse des requêtes, il faut procéder à une réduction plus drastique de la taille des métadonnées. Les relations *MomentsInertie* et *Couleurs* sont des exemples de telles réductions.

Dans le cas de *MomentsInertie*, les histogrammes sont remplacés par leurs quatre premiers moments d'inertie. Par ailleurs, le nombre de moments retenus étant constant et faible, la contrainte de 1FN peut être facilement violée. Les différents noms cachent des indices : moyenne arithmétique ($\overline{m} = m_0$), écart-type (m_1), asymétrie (m_2), aplatissement ou KURTOSIS (m_3), c'est-à-dire un vecteur. Les valeurs peuvent d'ailleurs se calculer à partir d'une même formule, variations autour des distances de MINKOWSKI : $m_i = \sqrt[i]{\frac{\sum_{j=0}^{255} |h_j - m_0|^i}{i}}, \forall i > 0$. En outre, cette transformation conserve plus d'information que la réduction drastique du nombre d'entrées dans les histogrammes.

Avec ORACLE, l'espace occupé par cinq régions et trois « histogrammes » est alors ramené à 465 octets, si les valeurs des moments d'inerties, normalisées avec la formule proposée, ne conservent que quatre chiffres après la virgule, soit 20 % de la configuration 120.

Sans même utiliser d'index, le tableau 6 (a) rapporte déjà de bien meilleures performances pour la recherche d'images contrastées, c'est-à-dire dont l'écart-type et l'aplatissement sont importants.

La relation *Couleurs* permet d'aller encore plus loin. Les trois histogrammes sont ramenés à un seul par une transformation heuristique associant chaque pixel à une « couleur » dans le langage courant, en l'occurrence les principales couleurs de l'arc-en-ciel (rouge, orange, jaune, vert, bleu clair, bleu et mauve) et quatre niveaux de gris. De plus, ces onze couleurs sont regroupées au sein d'un unique n-uplet. L'occupation mémoire est ainsi ramenée à 42 octets : 1,8 % de la configuration 120.

(a)

configuration	100 images	1 000 images	10 000 images	100 000 images
ORA	31 ms	125 ms	1 s 250 ms	2 s 500 ms
PG	2 s 687 ms	2 s 703 ms	2 s 766 ms	3 s 687 ms
PG+	253 ms	525 ms	4 s 000 ms	—

(b)

configuration	100 images	1 000 images	10 000 images	100 000 images
ORA	16 ms	16 ms	16 ms	16 ms
PG	18 ms	32 ms	31 ms	16 ms
PG+	26 ms	4 ms	4 ms	—
// (PG+)	—	8 ms	11 ms	58 ms

Tableau 6. *Performances de la sélection d'images contrastées (a) sans index et (b) avec index « multi-dimensionnel »*

3.2.3. *Indexation multi-dimensionnelle*

Les métadonnées multimédias présentent des difficultés importantes d'indexation physique. Indépendamment, elles sont trop peu sélectives, comme « vue intérieure / extérieure » pour une image ou « parole / musique » pour l'audio. Inversement, la prise en compte conjointe d'un grand nombre de propriétés se heurte à la « malédiction de la dimensionnalité », terme qui recouvre tout à la fois la difficulté de regrouper correctement les données et de retrouver ultérieurement un sous-ensemble significatif [SKI 97, BEL 61]. Malgré de nombreuses propositions (arbres-TV [LIN 94], -SR [KAT 97], -X [BER 96], -A [SAK 00]…), le problème de l'indexation des données multimédias, naturellement multi-dimensionnelles, reste donc mal résolu [BER 02].

Toutefois, la relation *MomentsInertie* se limitant aux quatre premiers moments d'inertie, il est possible de simuler une indexation multi-dimensionnelle en créant six arbres-B : $m_0m_1m_2m_3$ (pour les recherches sur « moyenne », « moyenne et écart-type », « moyenne, écart-type et asymétrie », ou « moyenne, écart-type, asymétrie et aplatissement »), $m_1m_2m_3$ (pour les recherches sur « écart-type », « écart-type et asymétrie », ou « écart-type, asymétrie et aplatissement »), $m_3m_0m_1$, m_2m_3, m_1m_3, m_0m_2. Le tableau 6 (b) démontre une amélioration extrêmement importante des performances. Tous les résultats restent largement en dessous de la seconde. La dernière ligne de ce tableau anticipe sur les performances des expérimentations parallèles.

Un unique arbre-R pourrait avantageusement remplacer ces six arbres-B [4]. En revanche, les arbres-R restent inutilisables pour la relation *Couleurs*, ces derniers s'effondrant au-delà de cinq dimensions [BER 96]. Une dernière limitation des propositions actuelles est que l'indexation multi-dimensionnelle devrait pouvoir s'appliquer (i) à plusieurs n-uplets d'une même relations (cas de la relation *HistogrammesRégions*) et (ii) à des n-uplets de différentes relations (entre les relations *Couleurs* et *Formats*, par exemple).

4. Le calcul de L_2 est optimisable avec un arbre-R mais pas avec une combinaison d'arbres-B (il faut définir un intervalle pour chaque dimension puis trier les distances calculées). Malheureusement, ils sont en option avec ORACLE et limités à deux dimensions avec PostgreSQL.

requête	100 images	1 000 images	10 000 images	100 000 images
jointure/120	19 s	2 m 56 s	26 m	4 h 45 m
L_2/120	5 s	6 s	20 s	5 m 25 s

Tableau 7. *Performances des requêtes sur une grappe de* $4 + 1$ *machines*

4. Expérimentations multi-processeurs

La réduction de la taille des données ne suffit pas à exécuter les requêtes aussi rapidement qu'on le souhaiterait. De plus, en l'absence de mécanismes spécifiques intégrés au SGBD, certaines requêtes, comme les calculs de distances, nécessitent toujours de parcourir l'ensemble des n-uplets d'une relation. Une approche complémentaire est d'exploiter le parallélisme.

4.1. *Configuration*

Les expérimentations suivantes ont été réalisées sur une grappe de machines comportant quatre machines bi-processeurs et une machine mono-processeur, point d'entrée de la grappe. La configuration des quatre principales machines est celle de la ligne PG+ du tableau 1 (la machine maître ne disposant que de 512 Mo de mémoire centrale).[5]

4.2. *Répartition naïve*

La répartition naïve a consisté à insérer *uniformément* les métadonnées de 25 000 images sur chacune des quatre machines esclaves. Le schéma de la base étant dupliqué sur les quatre sites, les requêtes peuvent être exécutées en parallèle sans aucune dépendance, à part l'inter-classement final effectué par le maître pour le calcul de L_2.

Les requêtes étant largement indépendantes, un gain de performance appréciable est obtenu pour les opérations les plus coûteuses, c'est-à-dire la jointure complète et la distance euclidienne. En comparant les résultats du tableau 7 avec ceux des tableaux 4 et 5, seule la ligne PG+/120 est à prendre en compte.

Idéalement, on peut exécuter une requête p fois plus vite avec p processeurs. Les requêtes les plus coûteuses vues ci-dessus ont une complexité en $O(n.\log_2 n)$. Il n'est pas possible d'envisager l'usage de n processeurs pour aboutir à des temps d'exécution logarithmiques. En revanche, la croissance des fonctions logarithmiques étant très

5. ORACLE dispose de plusieurs versions parallèles : sur des grappes de machines, sur des processeurs symétriques, sur des super-calculateurs. N'ayant à notre disposition ni les unes ni les autres, nous ne pouvons pas ici dresser de comparaison entre un produit industriel et une solution accessible à tous.

limitée, on s'intéresse naturellement au cas où $p = \log_2 n$. Il vient que la complexité d'une exécution locale d'une requête coûteuse est en $\frac{n}{\log_2 n} \times \log_2 \frac{n}{\log_2 n}$.

Le maître doit fusionner les sous-résultats. Chaque ensemble de données provenant des différent SGBD étant déjà trié, il suffit de choisir à tour de rôle le premier parmi les $p = \log_2 n$ premières réponses concurrentes. Une implémentation efficace d'une séquence ordonnée permet de réaliser l'insertion et la recherche du premier parmi les concurrents en $O(\log_2 p)$ [COR 94]. Par conséquent, l'algorithme de fusion des résultats a, avec $p = \log_2 n$, une complexité en $\frac{n}{\log_2 n} \times \log_2 \frac{n}{\log_2 n} + n.\log_2 \log_2 n$, soit, après simplifications algébriques et asymptotiques, en $O(n.\log_2 \log_2 n)$.

Cela dit, la complexité effective est plus exactement en $O(n + n'.\log_2 \log_2 n')$ où n' est le sous-ensemble des réponses qui sont réellement lues par le maître. En effet, si l'on ne contrôle pas la façon dont les SGBD calculent leurs résultats, en revanche, le maître n'a à télécharger et fusionner que le nombre de réponses demandées par l'utilisateur. Si $n' \ll n$ (par exemple, les cent plus proches images), alors le second terme devient négligeable et l'algorithme peut être jugé linéaire. Soulignons que cette amélioration notable est due à la présence d'un post-traitement, que nous contrôlons, dans l'architecture du processus d'interrogation.

5. Conclusion et perspectives

L'étude décrite visait à quantifier les difficultés auxquelles une base d'images est confrontée lorsque l'on veut la faire « passer à l'échelle ». Une contrainte importante, largement exogène, est le choix d'un SGBD *relationnel*. En particulier, nous avons écarté, pour le moment, toute extension propriétaire comme le modèle objet-relationnel d'ORACLE ou les tableaux de PostgreSQL.

Les premières conclusions que nous en tirons sont claires.

1) Le compactage standard qui permet d'améliorer les performances d'un SGBD sur des données traditionnelles ne suffit pas. Il est indispensable de réduire la taille des métadonnées dès l'extraction.

2) En l'état actuel des SGBDR, l'exploitation du parallélisme est incontournable pour les requêtes qui nécessitent de parcourir l'ensemble des métadonnées.

3) Des techniques de placement des données semblent indispensables. D'une part, elles permettent de profiter du parallélisme dans les limites du – strict – nécessaire. D'autre part, l'indexation physique des relations peut être particularisée à des sous-ensembles des métadonnées, ce qui est extrêmement important pour lutter contre la « malédiction de la dimensionnalité ».

Ces conclusions nous amènent naturellement à développer des travaux dans les mêmes directions.

1) En ce qui concerne la réduction des données, il faut veiller à extraire des descripteurs (i) qui soient visuellement pertinents, (ii) qui puissent se réduire et (iii) qui

soient aussi simples que possible. Il s'agit donc de trouver un compromis entre les données issues du traitement et de l'analyse de l'image, souvent complexes (histogrammes, séries de FOURIER, transformées en ondelettes, points ou régions d'intérêt, etc.) et les types de base d'un SGBD : nombres et relations. L'exemple utilisé a consisté à remplacer un histogramme par ses premiers moments d'inerties et par un histogramme « sémantique » de couleurs nommées (rouge, bleu, blanc, noir, etc.).

2) Dans l'exploitation du parallélisme, qui devra suivre la croissance de la taille de la base d'images, il faut trouver un équilibre entre les temps de traitement locaux et les temps de consolidation des résultats, ainsi qu'entre les performances des disques et celle du réseau. Notre étude a été réalisée sur une grappe de machine, ce qui assure une homogénéité des traitements locaux et des temps de communication entre toute paire de machine. Nous avons vu que notre configuration était sous-optimale en nombre de processeurs ; nous serons donc amenés à étendre nos expérimentations sur des grappes plus importantes. Dans les années à venir, un véritable SGBD multimédia sera un système largement réparti. Les contraintes seront donc plus fortes.

3) Le placement différencié des métadonnées sur les différentes machines permettra d'améliorer les performances globales des requêtes. Afin de l'optimiser, des techniques de classification doivent être utilisées. Une fois l'étape de classification réalisée, il reste encore à trouver un placement des clusters d'images sur une ou plusieurs machines, minimiser les éventuelles intersections, etc. Cela devra sans doute être réalisé de manière heuristique, de nouvelles expérimentations validant les stratégies de placement vis-à-vis d'une répartition uniforme.

Au-delà de ces prolongements, des questions sur le pouvoir d'expression des requêtes, leurs performances intrinsèques ou encore leur adaptation aux utilisateurs se font jour. Il nous semble que les réponses à ces nouveaux problèmes passent pour l'utilisation d'une couche intermédiaire entre le SGBDR et l'application. Ce médiateur pourra prendre en compte des traitements, coûteux ou intraduisibles en SQL, permettant de discriminer plus finement les résultats. Ce médiateur pourra aussi utiliser des relations contenant des variables linguistiques permettant (i) de fournir des descriptions textuelles au-dessus de données numériques et (ii) de différencier les appréciations des utilisateurs sur les valeurs des métadonnées.

Remerciements

Nous remercions Damien LE ROUX qui, durant un stage de DESS, a planifié, effectué et répété patiemment les expérimentations et mesures nécessaires à ce travail prospectif.

6. Bibliographie

[BEL 61] BELLMAN R., *Adaptive Control Processes : A Guided Tour*, Princeton University Press, 1961.

[BER 96] BERCHTOLD S., KEIM D. A., KRIEGEL H.-P., « The X-tree : An Index Structure for High-Dimensional Data », *22nd VLDB Int. Conf.*, Bombay, India, septembre 1996, p. 28–39.

[BER 02] BERRANI S.-A., AMSALEG L., GROS P., « Recherche par similarités dans les bases de données multidimensionnelles : panorama des techniques d'indexation », *Ingénierie des systèmes d'information*, vol. 7, n° 5-6, 2002, p. 9-44, Hermès-Lavoisier.

[CHA 98] CHANG W., MURTHY D., ZHANG A., SYEDA-MAHMOOD T. F., « Global Integration of Visual Databases », *14th IEEE ICDE*, Orlando, Florida, février 1998, p. 542–549.

[CHE 97] CHENG X., DOLIN R., NEARY M., ET AL., « Scalable Access Within the Context of Digital Libraries », *IEEE Int. Conf. on Advances in Digital Libraries*, Washington, D. C., 1997, p. 70–81.

[COR 94] CORMEN T., LEISERSON C., RIVEST R., *Introduction à l'algorithmique*, Dunod, 1994.

[DAT 98] DATE C. J., DARWEN H., *Foundation for Object/Relational Databases : The Third Manifesto*, Addison-Wesley, 1998, 496 p.

[KAN 96] KANTH K. V. R., AGRAWAL D., EL ABBADI A., SINGH A., SMITH T., « Parallelizing Multidimensional Index Structures », rapport n° 96-12, novembre 1996, Database Systems Lab, UCSB.

[KAN 99] KANTH K. V. R., AGRAWAL D., EL ABBADI A., SINGH A., « Dimensionality Reduction for Similarity Searching in Dynamic Databases », *Computer Vision and Image Understanding (CVIU)*, vol. 75, n° 1–2, 1999, p. 59–72.

[KAT 97] KATAYAMA N., SATOH S., « The SR-tree : an index structure for high-dimensional nearest neighbor queries », *ACM SIGMOD*, Tucson, Arizona, mai 1997, p. 369–380.

[KLA 95] KLAS W., ABERER K., « Multimedia Applications and Their Implications on Database Architectures », *Advanced Course on Multimedia Databases in Perspective*, University of Twente, The netherlands, 1995.

[LIN 94] LIN K.-I., JAGADISH H. V., FALOUTSOS C., « The TV-Tree : An Index Structure for High-Dimensional Data », *VLDB Journal*, vol. 3, n° 4, 1994, p. 517–542.

[MAR 00] MARTINEZ J., « Les bases d'images », *École thématique "Document & Evolution" du GdR I3*, vol. 1, Marseille, France, septembre 2000, Cépaduès Éditions, p. 129–158.

[MOO 01] MOON B., JAGADISH H. V., FALOUTSOS C., SALTZ J. H., « Analysis of the Clustering Properties of the Hilbert Space-filling Curve », *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, n° 1, 2001, p. 124–141.

[NAC 99a] NACK F., LINDSAY A., « Everything you Wanted to Know about MPEG-7 : Part 1 », *IEEE Multimedia*, vol. 4, n° 3, 1999, p. 65-77.

[NAC 99b] NACK F., LINDSAY A., « Everything you Wanted to Know about MPEG-7 : Part 2 », *IEEE Multimedia*, vol. 5, n° 4, 1999, p. 64-73.

[SAK 00] SAKURAI Y., YOSHIKAWA M., UEMURA S., KOJIMA H., « The A-tree : An Index Structure for High-Dimensional Spaces Using Relative Approximation », *26th VLDB Int. Conf.*, Cairo, Egypt, septembre 2000, p. 516–526.

[SKI 97] SKIENA S. S., *The Algorithm Design Manual*, Springer Verlag, 1997.