CONSORTIUM INTERNATIONAL E-MIAGE

MODULE C215

BASES DE DONNEES AVANCEES

Ch3 – Bases de données et Multimédia

MASTER METHODES INFORMATIQUES APPLIQUEES A LA GESTION DES ENTREPRISES

02 janvier 2013

Gérard-Michel Cochard

C215 - Bases de données avancées

Ch3. : Bases de données et Multimédia

Ch3.1 : Introduction

Multimédia et bases de données

Le mot multimédia est maintenant couramment employé comme adjectif. On se demande alors pourquoi on écrit encore "base de données multimédi**a**" alors qu'il est clair que ce sont les données qui sont multimédi**as**. Pour notre part, nous accorderons le mot multimédia avec le mot données et on écrira "base de données multimédi**as**".

Ceci étant, qu'est-ce qu'une donnée multimédia ? C'est une donnée qui appartient à plusieurs formats d'information :

- L'image fixe (dont l'écriture manuscrite)
- L'image animée
- Le son
- La vidéo
- Le texte alphanumérique (oubli important de Wikipedia).

L'ensemble conjugué de ces différentes données, pour une diffusion vers des utilisateurs, s'appelle…le multimédia. Il possède des spécificités fortes que nous résumons plus loin (il s'agit d'un résumé certainement pas exhaustif).
La problématique des bases de données multimédias est aisément compréhensible si l'on prend l'exemple illustratif suivant. Un hold up a lieu dans une banque et la police recherche activement les voleurs. Pour cela elle dispose

- d'enregistrement vidéo provenant des caméras de surveillance
- de conversations téléphoniques provenant de divers suspects
- de photographies prises par des témoins
- de photographies issues de fichiers de malfaiteurs

Tous ces documents sont volumineux et "manuellement" la police va mettre beaucoup de temps pour rechercher les corrélations possibles permettant d'identifier les voleurs. Il faut donc

- pouvoir enregistrer ces documents dans un dispositif unique
- pouvoir effectuer des requêtes sur cet ensemble de documents pour rechercher par exemple des similarités.

Sur ce dernier point il y a une énorme différence entre une requête du type "Rechercher l'adresse de la personne de nom 'Dupont' " et la requête "Rechercher 'Dupont' sur la photographie X". On conçoit que pour que la deuxième requête aboutisse, il est indispensable d'indexer l'image photographique X.

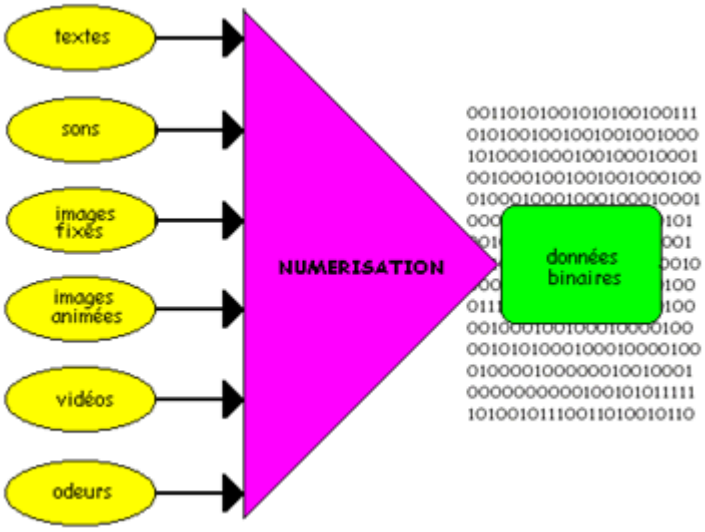
Spécificité des données multimédias

Le multimédia est l'intégration au sein d'un même système d'information de données textuelles, d'images fixes ou animées, de sons, de vidéo. Son histoire est très récente, depuis 1990 environ, et va de pair avec la tendance prononcée à la numérisation de l'information, la production de petits systèmes informatiques à possibilités très performantes et au développement de l'interconnexion de réseaux, notamment dans ce dernier cas, Internet et sa composante "multimédia" : le World Wide Web.

Selon les points de vue, le multimédia recouvre un marché en pleine expansion, un environnement de travail et de loisir (les jeux vidéo !), des applications professionnelles ou grand public, une industrie portant sur la production de matériels et logiciels. Une définition possible est celle-ci (selon Terrasson) : le multimédia est l'exploitation simultanée de données sonores, visuelles, informatiques, l'ensemble des techniques de création, de stockage, de transmission, de restitution, et aussi l'ensemble des données permettant cette exploitation simultanée.

Un premier intérêt du multimédia est l'intégration généralisée des données et des outils de manipulation de ces données. Avant le multimédia, des appareillages divers étaient nécessaires : un ordinateur pour les données informatiques, un magnétophone pour les cassettes de sons, un magnétoscope pour les vidéos, un téléphone pour le transport de la voix, du papier pour les images, …L'utilisation simultanée de données de natures différentes nécessitait donc une juxtaposition inconfortable de ces outils. L'utilisation de l'ordinateur comme outil unique est un progrès sensible.

Par ailleurs, les données multimédias sont manipulées comme des données informatiques puisque codées sous forme de 0 et de 1 et peuvent donc, en particulier, être stockées de manière unique et transmises de manière unique ce qui constitue un second intérêt



Cette intégration se fait cependant avec quelques problèmes :

- le volume de données numérisées est généralement important, ce qui nécessite des techniques de compression/décompression.
- Les traitements sont plus complexes et nécessitent des matériels et logiciels de plus en plus performants
- la transmission de données volumineuses nécessite des débits binaires importants sur les réseaux.

Néanmoins cette tendance semble irréversible et entraîne une reconfiguration du paysage autrefois audio-visuel en paysage totalement numérique.

La plupart des informations sont distribuées sous forme de textes, d'images fixes, de sons, de vidéo. Actuellement, il est possible de numériser ces médias (il n'est cependant pas encore possible de numériser les odeurs (sauf appareillages particuliers), les goûts et les sensations de toucher).

Les textes

Composés de caractères dits d'imprimerie, la numérisation s'opère simplement par codage de chaque caractère en une suite de 0 et de 1. Le code ASCII (American Standard Code for Information Interchange) sur 7 bits permet de coder 128 caractères usuels. Comme les ordinateurs travaillent usuellement sur des mots qui sont des multiples de mots de 8 bits (octets), on peut rajouter un 0 devant le code ASCII, ce qui correspond au code normalisé. Ainsi le caractère "A" est codé 01000010. On peut aussi rajouter un 1 et définir un second jeu de 128 caractères (code ASCII étendu) ; malheureusement ce second jeu n'est pas normalisé et varie d'une plate forme informatique à l'autre.

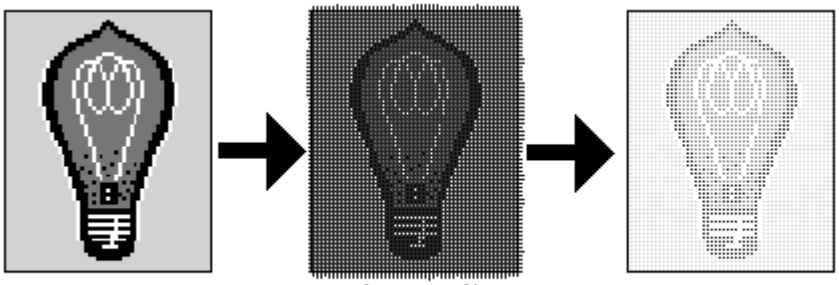
Il est également possible de coder la mise en forme du texte : gras, italique, souligné, la taille des caractères, l'alignement, etc. Là encore, il n'y a pas de normalisation et les logiciels manipulant du texte possèdent encore leur propre système.

Pour un texte comportant 2000 caractères (espaces compris), sans style, la taille du fichier numérisé sera donc de 2000 octets ce qui correspond à un volume faible. Le livre numérique des "Trois mousquetaires" d'Alexandre Dumas correspond à un ficher de 1371 Ko (abu.cnam.fr).

Cependant, l'ASCII est bien insuffisant pour coder tous les signes utilisés sur la planète (caractères chinois, arabes, arméniens, latins, bengalis, Braille, cherokees, etc.). Un code universel a donc été inventé : l'Unicode dont les caractères sont représentés par 16 bits ce qui donne 216 possibilités, soit 65536 possibilités (pas mal, non ?). De plus l'Unicode englobe l'ASCII. Mais l'inconvénient est justement la longueur du code (2 octets) alors que l'ASCII ne prenait qu'un seul octet. Si on considère un grand texte en français codé en Unicode il prendra le double de volume de sa version codée en ASCII. Mais il y a une solution : l'UTF-8. Cette solution consiste à utiliser l'ASCII (sur un octet par caractère) pour les caractères "anglophones" et utiliser Unicode seulement pour les caractères spéciaux (é, à, ê, ….) en prévenant évidemment que l'on a affaire à un caractère spécial. Cette méthode réduit très sensiblement les volumes de données. Aujourd'hui UTF-8 est lisble par tous les (bons) navigateurs.

Les images fixes

Pour les applications multimédias, les images sont converties en matrices de points. Chaque point est codé suivant sa "couleur" et les codes résultants sont placés séquentiellement dans un fichier, ligne par ligne, colonne par colonne.



Pour une image binaire, c'est à dire "bicolore", noir et blanc, chaque point est codé sur 2 bits (0 pour noir, 1 pour blanc. Pour une image avec 256 nuances de gris allant du noir total au blanc total, chaque point sera codé sur 8 bits (de 00000000 pour noir à 11111111 pour blanc). Pour une image couleur, on exprime cette couleur comme une superposition d'une dose de Rouge, d'une dose de Vert et d'une dose de Bleu (système RVB) ; si on permet une variation de 0 à 255 pour une dose, une couleur

quelconque sera codée comme la juxtaposition de trois octets, soit 24 bits (et donc 16 777 216 couleurs possibles).

Considérons une image pour laquelle la grille de codage se compose de 640*480 points. Si l'image est binaire, sa taille est de 38 400 octets, si l'image est à 256 nuances de gris, sa taille est de 307 200 octets ; si l'image possède des couleurs codées sur 24 bits, sa taille est de 921 600 octets. Si l'on considère qu'une vieille disquette 3p1/2 contient usuellement 1 400 000 octets, on comprend que ce support est quelque peu limité pour le stockage de nombreuses images. Par ailleurs, des techniques de compression permettent de gagner de la place mais au prix d'une décompression au moment de l'affichage des images ; ce que l'on gagne en volume est perdu en vitesse d'affichage.

La taille du fichier obtenu après numérisation et avant compression est fortement influencée par le pas de la grille de codage qui est appelé résolution et exprimé usuellement en points par pouce (dpi : dots per inch). Pour une même image une résolution de 600 dpi donnera un fichier image 36 fois plus volumineux que le fichier image résultant d'une numérisation à 100 dpi.

Les techniques de compression conduisent à des formats d'images, soit normalisés, soit imposés par les industriels. Trois formats d'image sont couramment employés pour les applications multimédias en ligne :

- le format GIF qui correspond à des images compressées sans perte d'information avec 256 nuances de couleurs
- le format PNG, une alternative à GIF (PNG's Not Gif) consistant en une amélioration de GIF.
- le format JPEG qui correspond à des images compressées avec perte d'information avec 16 millions de couleurs.

Ces trois formats ont l'immense avantage d'être interprétés par la plupart des navigateurs du Web.

Les sons

Les technologies d'acquisition du son ont été longtemps analogiques : le son était représenté par les variations d'une grandeur physique, une tension électrique par exemple. Les techniques actuelles permettent d'obtenir directement un son numérisé : c'est notamment le cas des magnétophones produisant un enregistrement sur cassettes D.A.T. (Digital Audio Tapes, digitalisation du son à une fréquence de 48 KHz) .

Pour numériser un son enregistré de manière analogique, on procède en trois étapes :

1. Echantillonnage : l'amplitude du signal analogique est mesurée à une fréquence d'échantillonnage f. On obtient ainsi une collection de mesures.
2. Quantification : une échelle arbitraire allant de 0 à 2n-1 est employée pour convertir les mesures précédentes. Une approximation est faite de manière à ce que chaque mesure coïncide avec une graduation de l'échelle (cette approximation, qui modifie légèrement le signal, est appelée bruit de quantification).
3. Codage : suivant sa grandeur dans cette nouvelle échelle, chaque mesure est codée sur n bits et placée séquentiellement dans un fichier binaire.

On comprendra que les valeurs de f et de n sont critiques pour la taille du fichier résultant. Usuellement 3 qualités de numérisation sont employées :

- la qualité Hifi ou CD audio : f=44 KHz, n=16 bits, stéreo (2 signaux sonores)
- la qualité "radio" : f= 22 KHz, n=8 bits, mono ou stéréo
- la qualité "téléphonique" : f=11 KHz, n=8 bits, mono

Le volume d'un fichier son est alors donné par la relation

$$V = f \times t \times n \times p$$

où f est la fréquence d'échantillonnage (en Hz), t la durée du son (en secondes), n le nombre de bits de codage d'un échantillon, p le nombre de piste (p = 1 pour un son mono, 2 pour un son stéréo, 4 pour un son quadriphonique) et V le volume du fichier (en bits). Ainsi, 1 heure de son correspondra à des fichiers de 630 Mo, 158 Mo et 40 Mo pour les qualités CD, radio et téléphonique respectivement. Nous verrons plus loin que les standards de CD ROM correspondent à une capacité d'environ 640 Mo et que les DVD vont jusqu'à 8,5 Go. Les normes de codage numérique du son portent sur l'enregistrement non compressé des mesures quantifiées (PCM : Pulse Code Modulation), sur l'enregistrement des différences entre deux mesures successives, ce qui permet une réduction de volume de données (DPCM : Delta PCM), ou encore sur l'utilisation de techniques de prédiction des mesures (ADPCM : Adaptative Differential PCM).

Etant donné les volumes importants que présentent les fichiers son, il est quasiment impératif de compresser ces fichiers. Parmi les formats de compression, le format MP3 (« ***MPEG-1 Audio layer 3*** ») est le plus populaire. Il emploie une méthode avec perte de données car il élimine les fréquences inaudibles pour l'oreille humaine et peut remplacer deux signaux voisins par un seul signal.

La vidéo

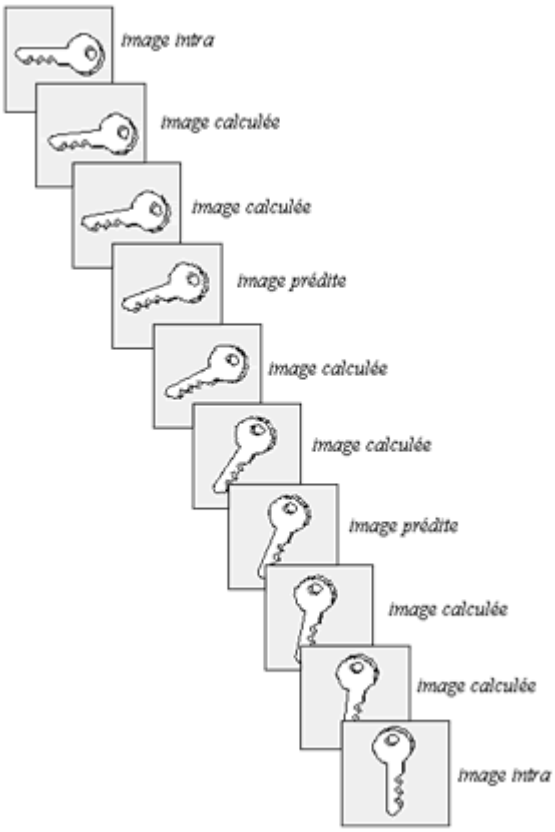
Comme pour le son, la vidéo s'exprime par des signaux de nature analogique qu'il faut numériser pour une utilisation multimédia. Il existe ici quatre signaux à numériser : trois pour l'image et un pour le son. Les signaux d'image se répartissent en signal de luminance Y et deux signaux de chrominance Db et Dr, reliés aux composantes R,V,B (Rouge, Vert, Bleu) par les relations linéaires

$$Y = 0.30R + 0.59V + 0.11B$$
$$Db = B - Y$$
$$Dr = R - Y.$$

La composante Y est échantillonnée à une fréquence de 13,5 MHz, tandis que les composantes de chrominance sont échantillonnées à une fréquence de 6,75 MHz. La quantification pour les trois signaux d'image s'opère sur 8 bits.

Les données peuvent être simultanément ou ultérieurement compressées. Plusieurs systèmes sont actuellement utilisés. La compression Indéo d'Intel est utilisée dans le logiciel Vidéo For Windows (le résultat de la numérisation et de la compression conduit à des fichiers vidéo AVI). Pour des plates-formes Macintosh, mais aussi Windows, le logiciel Quick Time d'Apple joue un rôle analogue à partir de fichiers vidéo MOOV ; un grand intérêt de Quick Time est sa possibilité d'intégration de données initiales diverses (images, sons, textes) et de proposer plusieurs types de compression. Les logiciels Video For Windows et Quick Time réalisent également la synchronisation du son avec les images.

Les standards MPEG (toute une série), spécialement élaborés pour la compression de vidéo, tendent à s'imposer. Le principe de compression s'appuie sur trois types d'images : les images "intra" sont des images peu compressées qui servent de repère (une image intra pour 10 images successives) ; les images "prédites" sont des images obtenues par codage et compression des différences avec les images intra ou prédites précédentes (une image prédite toutes les trois images) ; les images "interpolées" sont calculées comme images intermédiaires entre les précédentes. L'utilisation de vidéos numériques MPEG nécessite la présence d'une carte de décompression dans le micro-ordinateur d'exploitation.

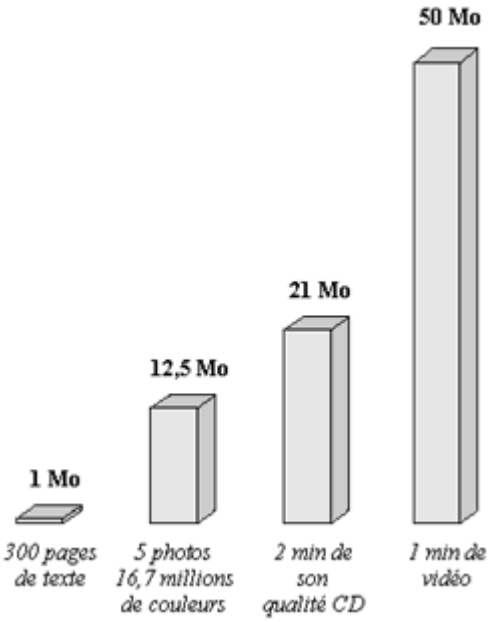


Un des standards MPEG mérite notre attention pour la suite : le standard MPEG-7 (qui porte aussi le nom de Multimedia Content Description Interface) qui permet de rendre possible une recherche d'informations dans un flux vidéo.

Spécificité des supports

Les supports des données ou des applications multimédia doivent satisfaire deux contraintes essentielles :

- posséder des capacités importantes de stockage
- posséder des taux d'entrée/sortie suffisants pour l'exploitation en temps réel.



Les supports classiques que sont les disquettes 1,4 Mo sont évidemment à éliminer du fait de leur faible capacité. Les disques durs magnétiques pourraient être de bons candidats pour l'hébergement d'une application multimédia de taille raisonnable ; en particulier, leur débit d'entrée/sortie est satisfaisant ; mais leurs coûts et leurs technologies qui les rendent indissociables de l'ordinateur (peu adapté au stockage) les éliminent, quant il s'agit de supports de diffusion commerciale, au profit de la famille des CD-ROM qui, née avec le multimédia, est actuellement, avec les DVD-ROM, le support quasi-universel des applications et des données multimédias distribués off line.

Par contre, pour des applications mettant en jeu des bases de données, accessibles en local ou à distance, l'utilisation des disques durs est encore la meilleure solution, sauf, peut-être, pour des données dont les modifications sont rares ; dans ce dernier cas, l'utilisation de dispositifs de type juke-box de CD-ROM/DVD-ROM peut être une solution.

Spécificité des outils et méthodes de développement

Le développement d'une application multimédia implique des outils spécifiques adaptés au type des données à traiter ou à leur intégration.
Les outils qui sont liés au type de donnée sont

- pour l'image : outils d'acquisition, de dessin, d'édition et de retouche.
- pour le son : outils de numérisation, d'édition et de retouche
- pour la vidéo : outils de numérisation, d'édition et de retouche, d'animation
- pour le texte : outils d'édition, de reconnaissance optique de caractères.

Suivant les cas, ces outils sont des composants matériels et/ou logiciels.
Les outils d'intégration sont logiciels et correspondent soit à la structuration des données, soit à leur manipulation et en particulier à la réalisation de l'interface utilisateur. Les outils de structuration des données sont principalement des systèmes de gestion de bases de données dotés d'extensions permettant la gestion de données multimédias. Les outils de manipulation peuvent être des langages auteurs orientés multimédias (ils réalisent alors directement l'intégration de données multimédias de types différents) ou des langages de programmation usuels (c'est au programmeur de réaliser tous les traitements).

Spécificité des outils de traitement

Le multimédia nécessite, pour son exploitation, des ressources supplémentaires par rapport à celles des ordinateurs classiques. La station multimédia est un ordinateur, le plus souvent micro-ordinateur, doté

- de composants matériels performants: processeurs rapides (200 MHz), mémoire vive de grande capacité (32 Mo), écran 640x480 au moins ;
- d'extensions spécifiques : carte son, baffles ou casque, microphone, carte vidéo, carte réseau, modem, lecteur de CD-ROM/DVD-ROM,… ;
- de modules logiciels particuliers pour l'exploitation du son, de la vidéo, de l'image, de la communication (notamment navigateur www).

Heureusement ces ressources se sont généralisées et les micro-ordinateurs du marché sont de plus en plus multimédias dans leur configuration de base et les prix continuent de baisser pour la quasi-totalité des constructeurs. En fait, il semble que les smartphones et tablettes graphiques soient bien en avance sur les ordinateurs portables classiques.

Bien entendu, ces considérations ne s'appliquent qu'aux "postes" d'exploitation ; les dispositifs de stockage et de diffusion que constituent les serveurs peuvent être, au contraire, d'une très grande complexité technique et d'un coût nettement plus important.

Spécificité des outils de diffusion

Dans le domaine du multimédia, on parle souvent d'applications "on line" et d'application "off line" pour indiquer que l'exploitation des données multimédias s'effectue en connexion avec un réseau informatique ou en local sans réseau. Plus précisément, ces deux modes d'exploitation correspondent aux deux modes de diffusion actuels que sont la diffusion par CD-ROM/DVD-ROM et la diffusion par le World Wide Web.

Le CD-ROM est le moyen de stockage le plus pratique pour le multimédia et son utilisation première a été pour l'enregistrement du son sur le CD Audio en remplacement du disque vinyle. D'une capacité globale de 640 Mo, le CD-ROM permet l'enregistrement de 200 000 pages de texte brut, ou de 2000 images en 256 couleurs, ou de 250 images en 16 millions de couleurs, ou de 12 minutes de vidéo, ou de 60 minutes de son, ou, bien sûr, un mixage de différents médias. Ces chiffres montrent à la fois les avantages et les inconvénients du CD-ROM : sa capacité est grande mais ses limites sont vite atteintes. L'apparition du DVD (Digital Versatile Disk) permet de multiplier par 10 les capacités de stockage ; et prend peu à peu la place des "bons vieux CD".

L'autre mode de diffusion consiste en la composante interactive d'Internet : le Web. Basé sur une architecture de type client-serveur ce service assure la délivrance par un serveur sur un poste client de documents multimédias encodés par un langage de description appelé HTML (HyperText Mark up Language). Ces documents HTML sont interprétés par un logiciel spécifique, le navigateur, installé sur le poste client. Le navigateur (browser, en anglais) permet la communication avec le serveur et l'affichage du document. Le protocole HTTP (HyperText Transfer Protocol) qui régit les communications entre le serveur et le client est l'outil de base pour la diffusion sur réseau de documents multimédias. On connaît le succès de cette technologie (de 12 millions d'internautes en France en 2001 à 49 millions en 2012 soit 71,6% des Français de 11 ans et plus selon Médiamétrie).

Ch3.2 : Le multimedia dans l'objet-relationnel

Les types SQL3 pour le multimedia

SQL3 a défini des types de données "classiques" :

- CHAR(n) : chaîne de carractères de longueur n
- CHAR VARYING(n) : chaîne de caractères de longueur maximale n [Oracle nomme ce type VARCHAR2(n)]
- INTEGER : nombre entier [Oracle nomme ce type NUMBER]
- NUMERIC(p,q) ; nombre décimal de p chiffres avec q chiffres décimaux [Oracle nomme ce type NUMBER(p,q)]
- FLOAT: nombre flottant [pas d'équivalent chez Oracle]
- DATE : date
- TIME : heure
- TIMESTAMP : date et heure [pas d'équivalent chez Oracle]

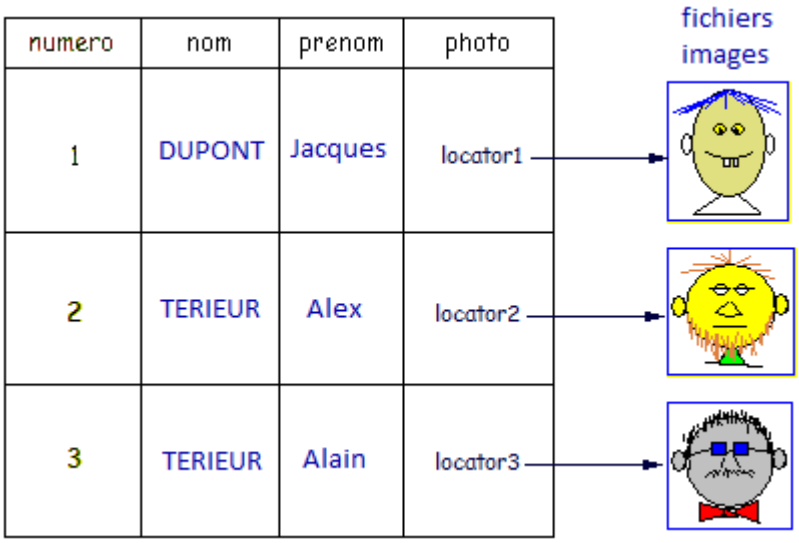
Il a aussi introduit des types de données se référant à des objets multimédias. Ces types sont repris dans Oracle :

- BLOB : objet codé en binaire (Binary Large Object)
- CLOB : objet codé en caractères (Character Large Object)
- NCLOB : objet codé en caractères "nationaux" (National Character Large Objet)
- BFILE : fichier binaire externe (Binary File)

BLOB, CLOB, NCLOB (les "LOB") correspondent à des documents multimédias. BLOB est utilisé pour des fichiers binaires de type image, audio ou vidéo. CLOB et NCLOB sont utilisés pour des données texte volumineuses. Dans tous les cas, le stockage comprend un "LOB-locator", qui est un pointeur vers les données multimédias, et les données multimédias elles-mêmes, "LOB-value". Dans Oracle, si la taille du LOB est inférieure à 4Ko, le LOB-locator et la LOB-value sont stockés directement dans la table qui les réfère. Sinon la table ne contient que le LOB-locator et la LOB-value est stockée en dehors de la table (mais dans la base de données cependant). La taille d'un LOB peut aller jusqu'à 128 To, mais tout dépend du SGBD utilisé (4Go pour Oracle)

A contrario BFILE désigne des fichiers représentant des objets multimédias stockés à l'extérieur de la base de données. Dans la base de données BFILE correspond à un pointeur (BFILE-locator) permettant de trouver l'objet extérieur à la base de données (sous forme d'un fichier binaire).

Exemple 1 : trombinoscope : on souhaite créer une table répertoriant des personnes avec leur nom, leur prénom et leur photographie. On ajoutera un numéro identificateur pour distinguer les personnes. On a donc une relation PERSONNE (numero, nom, prenom, photo) dont la vue conceptuelle est la suivante :

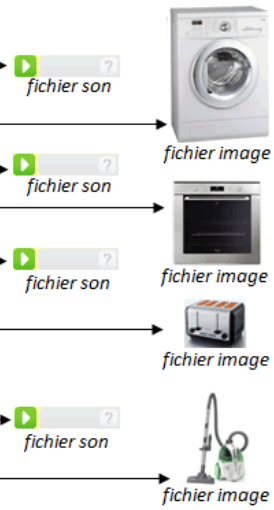



```
CREATE TABLE personne
(
        numero NUMBER PRIMARY KEY,
        nom VARCHAR2(20),
        prenom VARCHAR2(20),
        photo BLOB
);
```

On a ici choisi de stocker les photographies dans la base de données. Bien entendu, dans la table, on n'a pas les photos directement (comme dans l'image précédente) mais en général des LOB-locators.

Exemple 2 : catalogue de produit --> table PRODUIT (<u>numero</u>, nom, description, commentaire, photo) où description est un long texte, commentaire une plage sonore et photo une image.

numero	nom	description	commentaire	photo
1	lave linge	12 programmes. Jusqu'à 6 mouvements de tambours différents pour une meilleure qualité de lavage. Départ différé, affichage du temps restant. Garantie 2 ans.	locatorC1	locatorP1
2	four	Four multifonction pyrolyse inox abwhirl 876Hp67	locatorC2	locatorP2
3	grille pain	Pour tous types de pains, Thermostat 7 positions, Parois froides, 4 fentes ajustable extra larges et profondes Boutons lumineux : réchauffage, décongélation, Arrêt automatique Ramasse miettes frontal. 1800 watts	locatorC3	locatorP3
4	aspirateur	Puissance 2000 W - Dépression 29 kPa - Débit d'air 29 dm3/s - Décibel 79 dB(A) - Capacité de la cassette 2 litres - Rayon d'action 8 m - XYZ 8	locatorC4	locatorP4




```
CREATE TABLE produit
(
        numero NUMBER PRIMARY KEY,
        nom VARCHAR2(30),
        description CLOB,
        commentaire BLOB,
        photo BFILE
);
```

On a ici choisi de stocker les photos à l'extérieur de la base de données mais pas les sons, ni les textes. Comme dans l'exemple précédent, dans les champs description, commentaire, photo on trouve des LOB-locators qui pointent vers les objets concernés (sauf si la taille des CLOB et BLOB est inférieure à 4 Ko).

exemple avec Oracle XE :

Nom De Colonne	Type De Données	Valeur Nullable	Valeur Par Défaut	Clé Primaire
NUMERO	NUMBER	No	-	1
NOM	VARCHAR2(30)	Yes	-	-
DESCRIPTION	CLOB	Yes	-	-
COMMENTAIRE	BLOB	Yes	-	-
PHOTO	BFILE	Yes	-	-
				1 - 5

Manipulation des données multimédias (Oracle)

Considérons la création de table précédente :

```
CREATE TABLE produit
(
        numero NUMBER PRIMARY KEY,
        nom VARCHAR2(30),
        description CLOB,
        commentaire BLOB,
        photo BFILE
);
```

et examinons les possibilités de manipulation des données par type de donnée en supposant que les champs numero et nom sont déjà renseignés..

CLOB, NCLOB

Commençons par initialiser à "vide" le champ description avec la fonction EMPTY_CLOB(). Par exemple,

```
UPDATE produit
SET description = EMPTY_CLOB( )
WHERE nom = "aspirateur";
```

La fonction EMPTY_CLOB( ) initialise un champ vide pour un CLOB ou un NCLOB. Pour les BLOB, on a une fonction équivalente EMPTY_BLOB( ).

Pour insérer un CLOB ou un NCLOB dont la taille est inférieure à 4ko, l'utilisation de SQL est très simple : il suffit de mettre à jour le champ correspondant au CLOB.

Par exemple

```
UPDATE produit SET description = 'Puissance 2000 W - Dépression 29 kPa - Débit d air 29
dm3/s - Décibel 79 dB(A) - Capacité de la cassette 2 litres - Rayon d action 8 m - XYZ 8'
WHERE numero = 4;
```

et si l'on voulait insérer un nouvel enregistrement avec une valeur pour le CLOB :

```
INSERT INTO produit (numero, nom, description) VALUES (5, 'mixeur',
'Blablablablablablabla');
```

exemple avec OracleXE :

EDIT	NUMERO	NOM	DESCRIPTION	COMMENTAIRE	PHOTO
	1	lave-linge	12 programmes ; 6 mouvements de tambours différents pour une meilleure qualité d elavage ; départ différé ; affichage du temps restant ; garantie 2 ans	[datatype]	[datatype]
	2	four	Four multifonction pyrolyse inox abwhirl 876Hp67	[datatype]	[datatype]
	3	grille-pain	Pour tous types de pains; Thermostat 7 positions ;Parois froides ; 4 fentes ajustables extra larges et profondes ; Boutons lumineux : réchauffage, décongélation, arrêt automatique ; ramasse-miettes frontal ; 1800 Watts	[datatype]	[datatype]
	4	aspirateur	Puissance 2000 W - Dépression 29 kPa - Débit d air 29 dm3/s - Décibel 79 dB(A) - Capacité de la cassette 2 litres - Rayon d action 8 m - XYZ 8	[datatype]	[datatype]
	5	mixeur	Blablablablablablabla	[datatype]	[datatype]
					row(s) 1 - 5 of 5

Pour lire un CLOB dont la taille est inférieure à 4 ko, un simple SELECT suffit car le type est équivalent à un VARCHAR :

SELECT description FROM produit WHERE numero = 1;

exemple avec OracleXE :

DESCRIPTION
12 programmes ; 6 mouvements de tambours différents pour une meilleure qualité d elavage ; départ différé ; affichage du temps restant ; garantie 2 ans

Cependant, dans le cas général, Pour accéder à un LOB il faut utiliser les fonctions du paquetage DBMS_LOB dont voici un extrait :

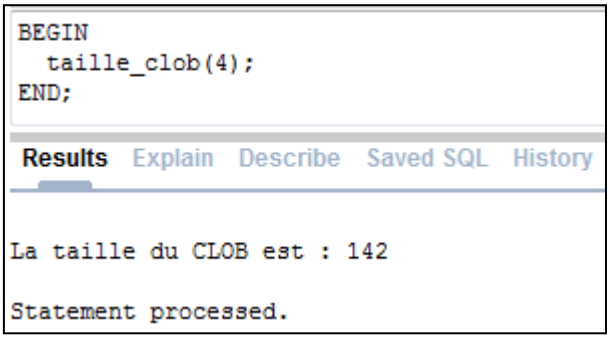
description	**utilisation**
Lecture d'un LOB à partir d'une position	DBMS_LOB.READ(lob-locator, taille, position, buffer)
Ecriture d'un LOB à partir d'une position	DBMS_LOB.WRITE(lob-locator, taille, position, texte)
Ajout de données à la fin d'un LOB	DBMS_LOB.WRITEAPPEND(lob-locator, taille, texte)
Obtention d'une partie d'un LOB à partir d'une position	SELECT DBMS_LOB.SUBSTR(attribut,position, longueur)
Obtention de la position d'une partie d'un LOB	SELECT DBMS_LOB.INSTR(attribut, valeur)
Obtention de la longueur d'un LOB	SELECT DBMS_LOB.GETLENGTH(attribut)

ainsi que la fonction d'écriture DBMS_OUTPUT.PUT_LINE().

Comme exemple, identifions la taille des CLOB de la table produit. Il faut pour cela créer une procédure en PL/SQL :

```
CREATE OR REPLACE PROCEDURE taille_clob(num IN INTEGER)
AS
v_clob CLOB;
BEGIN
SELECT description INTO v_clob FROM produit WHERE numero=num;
DBMS_OUTPUT.PUT_LINE('La taille du CLOB est : ' || DBMS_LOB.GETLENGTH(v_clob));
END;
```

On obtiendrait avec Oracle XE :



Pour lire le "champ" description de la table produit pour numero = 4, il faut aussi créer des procédures avec le langage PL/SQL. Tout d'abord, il faut récupérer le LOB-locator avec la procédure req_clob(v_clob, num) qui à partir de la valeur num du numéro de produit fournit le LOB_locator v_clob :

```
CREATE OR REPLACE PROCEDURE req_clob(v_clob IN OUT CLOB, num IN INTEGER)
IS
BEGIN
        SELECT description INTO v_clob FROM produit WHERE numero =num;
END;
```

Puis avec le LOB-locator, on accède au CLOB avec la procédure Read_description (num)

```
create or replace PROCEDURE Read_description(num IN INTEGER)
IS
        loc_clob CLOB;
        buffer VARCHAR2(400);
        position INTEGER := 1;
        taille INTEGER := 400;
BEGIN
        req_clob(loc_clob,num);
        buffer :=DBMS_LOB.GETLENGTH(loc_clob);
        DBMS_LOB.READ(loc_clob, taille, position, buffer);
        DBMS_OUTPUT.PUT_LINE('buffer = ' || buffer);
        DBMS_OUTPUT.PUT_LINE('taille = ' || taille);
END;
```

Nous avons choisi un buffer de taille 400 car la longueur des chaînes est inférieure à cette valeur. Pour manipuler les données issues de la base de données, on utilise les fonctions

On obtiendrait avec OracleXE :

```
BEGIN
 Read_description(4);
END;
```

Results | Explain | Describe | Saved SQL | History

```
buffer = Puissance 2000 W - Dépression 29 kPa - Débit d air 29 dm3/s - Décibel 79 dB(A) - Capacité de la cassette 2 litres - Rayon d action 8 m - XYZ 8
taille = 142

Statement processed.
```

Si l'on prend un très grand texte (entre 4Ko et 4Go) il sera stocké dans la base, mais pas dans la table. Nous ne considérerons pas ce cas ici car il nous entrainerait dans des spécificités complexes d'Oracle.

BFILE

Rappelons que le type BFILE correspond à des fichiers binaires enregistrés à l'extérieur de la base de données. La première chose à faire est de créer un alias du répertoire où se trouve le fichier binaire.

```
CREATE DIRECTORY MEDIAS AS 'C:\MEDIAS';
```

Bien entendu, il faut les droits pour créer le répertoire et y accéder (GRANT ..... TO.....). Ceci peut être effectué avec une ligne de commande SQL en se connectant comme administrateur de la base (on suppose que son login est SYSTEM et que son mot de passe est SYSTEM ; par ailleurs l'utilisateur est par exemple TARTARIN) :

```
SQL> connect SYSTEM/SYSTEM
Connected
SQL> GRANT CREATE ANY DIRECTORY TO TARTARIN;
Grant succeeded
```

Après avoir créé la DIRECTORY MEDIAS, on peut ensuite insérer les images (définies par l'attribut photo) dans la table produit :

```
UPDATE produit SET photo=BFILENAME('MEDIAS','lave-linge.png') WHERE numero=1;
UPDATE produit SET photo=BFILENAME('MEDIAS','four.png') WHERE numero=2;
UPDATE produit SET photo=BFILENAME('MEDIAS','grille-pain.png') WHERE numero=3;
UPDATE produit SET photo=BFILENAME('MEDIAS','aspirateur.png') WHERE numero=4;
```

Pour manipuler les objets de type BFILE on utilise aussi des fonctions DBMS_LOB spécifiques qui consistent à ouvrir un fichier, charger un fichier et fermer un fichier. Toute opération doit ouvrir et fermer un fichier. L'opération de fermeture peut être utile car le nombre maximal de fichiers ouverts est limité (par défaut à 10 dans Oracle).

description	utilisation
Ouverture de fichier	DBMS_LOB.OPEN(lob-locator, DBMS_LOB.LOB_READONLY)
Chargement d'un objet BFILE dans un objet BLOB	DBMS_LOB.LOADFROMFILE(lob-cible, lob-source, nb_octets, position_de, position_à)
Fermeture de fichier	DBMS_LOB.CLOSE(lob-locator)

Nous allons utiliser ces fonctions dans le paragraphe suivant.

BLOB

Les BLOBs se manipulent comme les CLOB. Pour insérer un BLOB qui n'est pas un texte (dans notre exemple c'est un fichier son de 8218 Ko), il faut passer par BFILE. On utilise par exemple la procédure suivante :

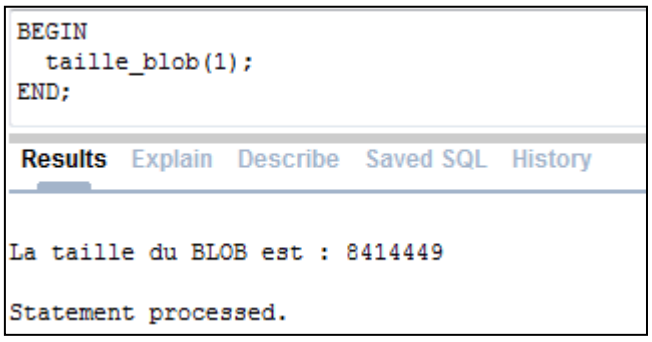
```
CREATE OR REPLACE PROCEDURE Insert_blob(num IN INTEGER)
IS
        src_blob BFILE := BFILENAME('MEDIAS','son1.mp3');
        dest_blob BLOB;
BEGIN
        UPDATE produit SET commentaire= EMPTY_BLOB() WHERE numero = num;
        SELECT commentaire INTO dest_blob FROM produit WHERE numero = num;
        DBMS_LOB.OPEN(src_blob, DBMS_LOB.LOB_READONLY);
        DBMS_LOB.LoadFromFile(dest_blob,src_blob,DBMS_LOB.GETLENGTH(src_blob));
        DBMS_LOB.CLOSE(src_blob);
        COMMIT;
END;

BEGIN
        Insert_blob(1);
END;
```

Bien entendu, on ne peut entendre le son avec la base de données (il faudrait faire un programme incorporant la recherche SQL). Toutefois, on peut vérifier que le BLOB est bien là en affichant sa taille avec la procédure

```
CREATE OR REPLACE PROCEDURE taille_blob(num IN INTEGER)
IS
v_blob BLOB;
BEGIN
SELECT commentaire INTO v_blob FROM produit WHERE numero=num;
DBMS_OUTPUT.PUT_LINE('La taille du BLOB est : ' || DBMS_LOB.GETLENGTH(v_blob));
END;
```

exemple avec OracleXE



Le résultat est concordant car on obtient 8414449/1024 = 8127 Ko.

A noter que l'on peut obtenir plus rapidement ce résultat avec :

```
SELECT DBMS_LOB.GETLENGTH(commentaire) FROM produit WHERE numero = 1
```

Results | Explain | Describe | Saved SQL | History

DBMS_LOB.GETLENGTH(COMMENTAIRE)
8414449

Ch3.3 : Structures de données multidimensionnelles

Les arbres k-dimensionnels (k-d Trees)

Imaginons que l'on veuille représenter les villes d'une carte géographique comme celle présentée ci-dessous. On peut alors utiliser la technique des arbres qui est d'essence hiérarchique et permet d'effectuer des recherches. Plusieurs types d'arbres sont à considérer. Commençons par les arbres k-dimensionnels destinés à représenter des données dans un espace à k dimensions. La carte géographique proposée correspond à un espace à 2 dimensions. On utilisera donc un "2-d Tree" dont les nœuds comporteront des enregistrements possédant au moins 2 valeurs x et y représentant les coordonnées d'un point sur la carte. Ces enregistrements peuvent posséder aussi d'autres attributs (par exemple le nom de la ville pour reprendre l'exemple illustratif). Comme il s'agit d'un arbre il faut aussi indiquer des pointeurs vers les nœuds suivants. Pour un arbre 2-d, un nœud se présentera donc sous la forme suivante :

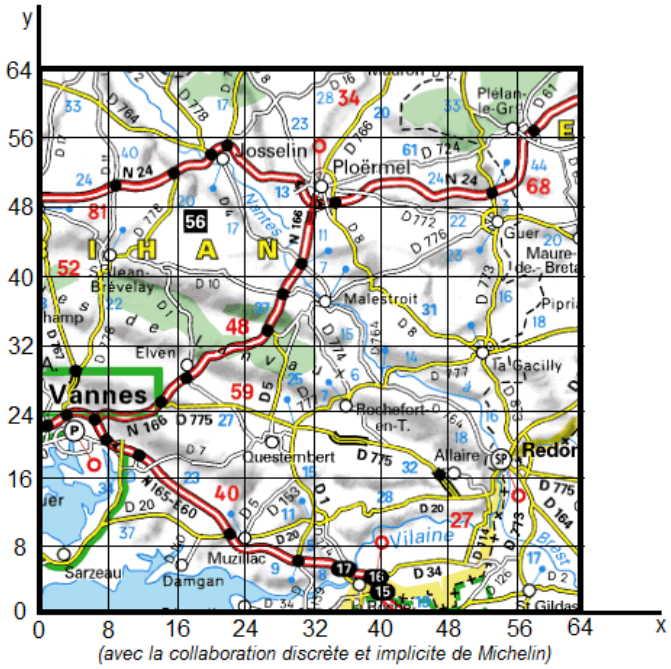
information	x	y
@gauche	@droite	

Cela signifie qu'un nœud de l'arbre ne peut avoir que deux descendants au plus, l'un vers la gauche et l'autre vers la droite. La racine de l'arbre correspond au niveau 0. Ses descendants immédiats (2 au plus) définissent le niveau 1, et ainsi de suite.

Par ailleurs, les nœuds sont placés suivant la règle suivante :

- Si un nœud N appartient à un niveau pair, tous les descendants du sous-arbre pointé par @gauche ont x < x(N) et tous les descendants du sous-arbre pointé par @droite ont x ≥ x(N).
- Si un nœud N appartient à un niveau impair, tous les descendants du sous-arbre pointé par @gauche ont y < y(N) et tous les descendants du sous-arbre pointé par @droite ont y ≥ y(N).

Supposons que l'on ait à construire cet arbre pour la carte géographique proposée.

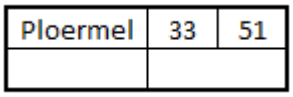


(avec la collaboration discrète et implicite de Michelin)

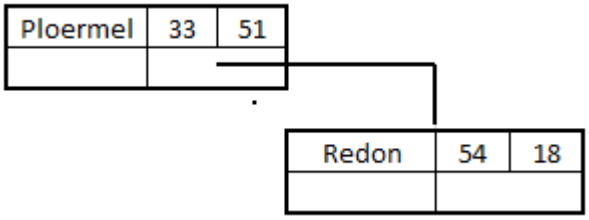
Les villes à prendre en considération sont listées dans le tableau suivant :

information	x	y
Ploërmel	33	51
Redon	54	18
Vannes	4	22
Questembert	26	20
Muzillac	24	9
Josselin	22	54
La Gacilly	52	31

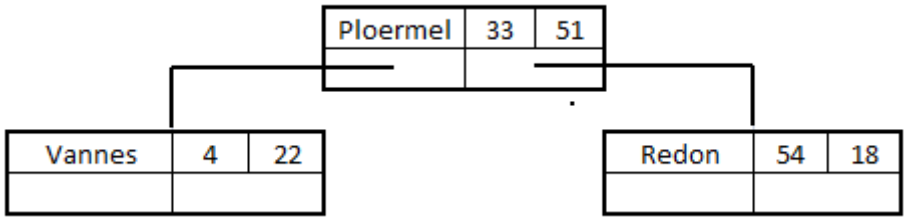
On commence par placer le premier nœud de l'arbre correspondant à la première ville de la liste (niveau 0) :



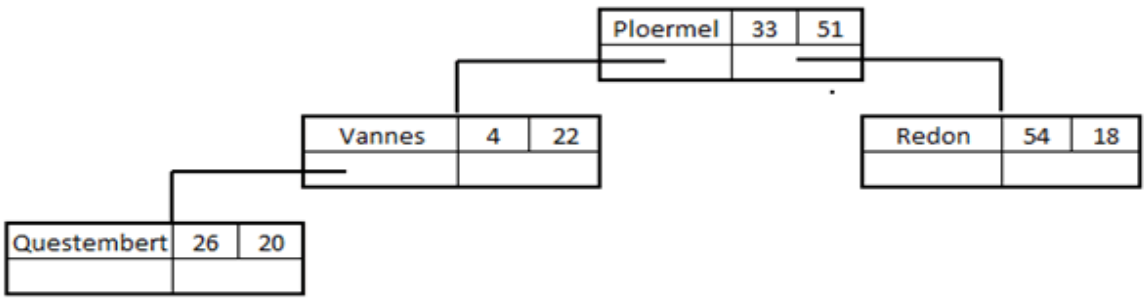
On passe ensuite à la deuxième ville de la liste, Redon (54,18). Comme 54 > 33, le nouveau nœud sera à droite du précédent :



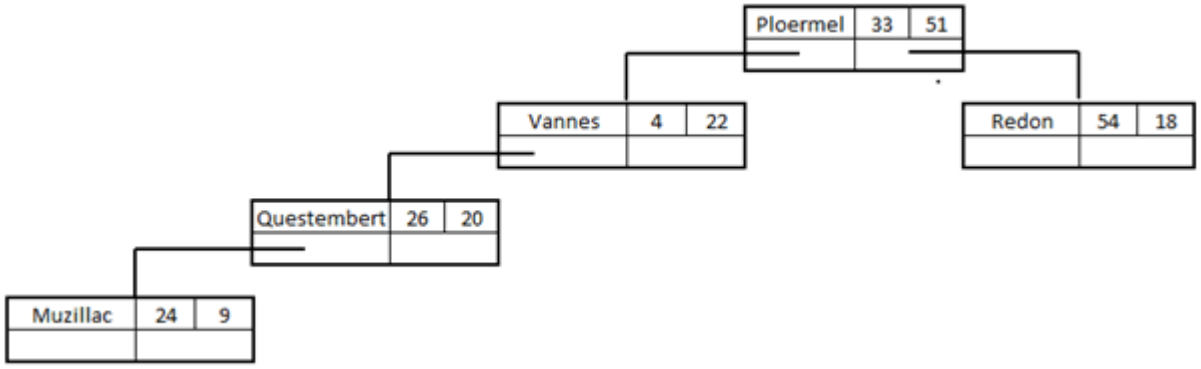
Passons à la troisième ville, Vannes (4,22). On compare d'abord 4 à 33 (de Ploermel) ce qui indique que l'on est à gauche de Ploermel. Comme il n'y a rien à gauche, on place le nœud "Vannes"



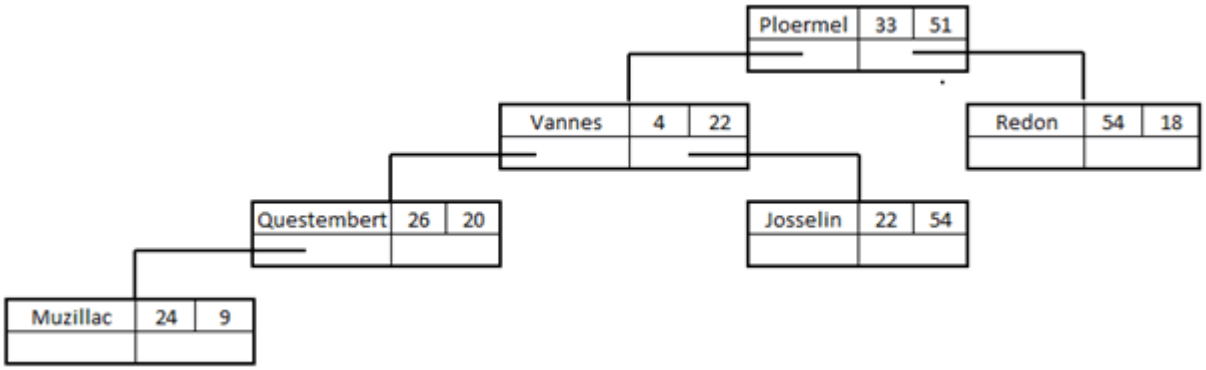
Quatrième ville de la liste, Questembert (26, 20). On compare d'abord 26 à 33 (de Ploermel) ce qui indique que "Questembert" est à gauche de "Ploermel" ce qui nous amène au nœud "Vannes". Puis on compare 20 à 22 (de Vannes) ce qui indique que "Questembert" est à gauche de "Vannes" :



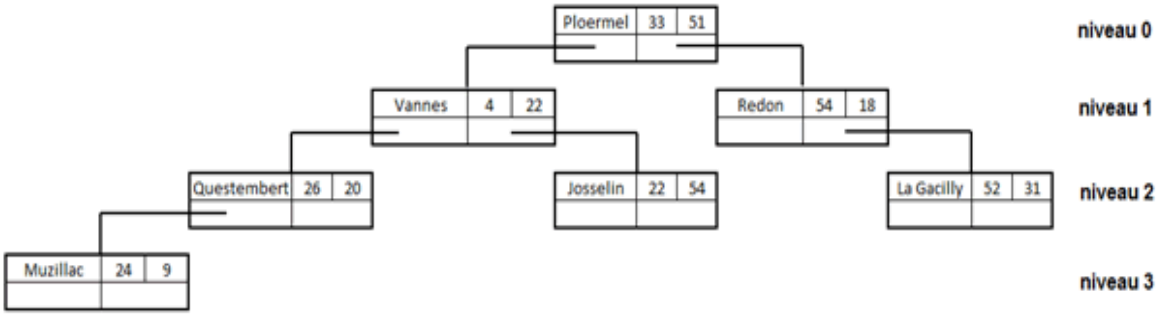
Cinquième ville de la liste, Muzillac (24,9). On compare d'abord 24 à 33 (de Ploermel) ce qui indique que l'on doit aller vers "Vannes". Puis on compare 9 à 22 (de Vannes) ce qui nous amène à "Questembert". Puis on compare 24 à 26 (de Questembert) ce qui nous indique qu'il faut placer le nouveau nœud à gauche de "Questembert" :



Sixième ville de la liste, Josselin (22, 54). On compare d'abord 22 à 33 ce qui nous amène vers "Vannes". Puis on compare 54 à 22 (de Vannes) ce qui crée à nouveau nœud à droite de "Vannes" :

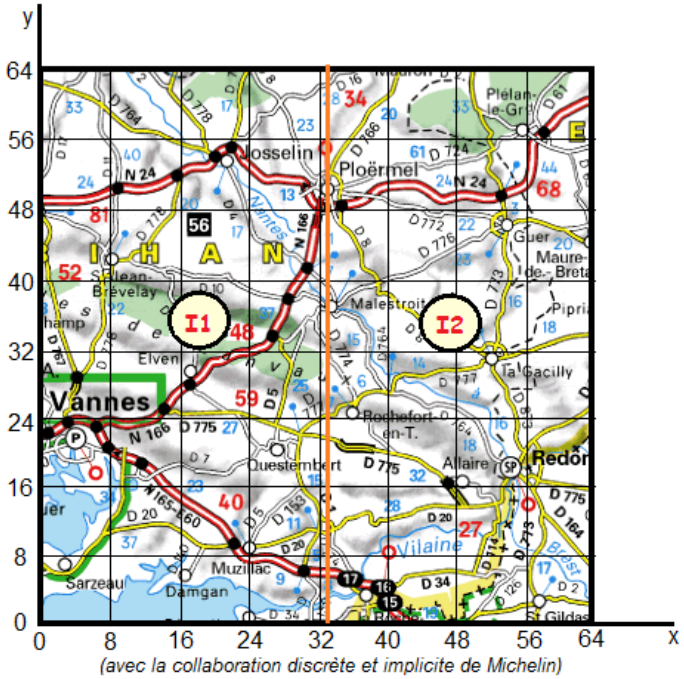


Enfin, septième ville de la liste, La Gacilly (52, 31). On compare d'abord 52 à 33 (de Ploermel) ce qui nous amène à "Redon". Puis on compare 31 à 18 (de Redon) ce qui nous crée un nouveau nœud à droite de "Redon" :



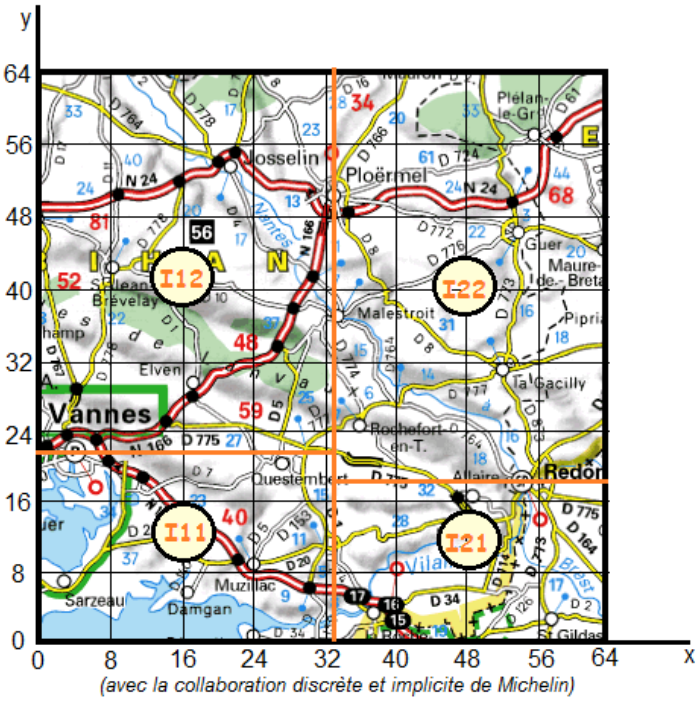
La représentation obtenue est un 2-d Tree à 4 niveaux.

Il est intéressant d'examiner la signification géométrique du processus de construction de cet arbre. Le premier nœud "Ploërmel" de niveau 0 correspond géographiquement à un point représentant l'image toute entière I. A partir de ce point on peut scinder l'image en deux parties I1 et I2 en traçant la droite verticale x = 33.



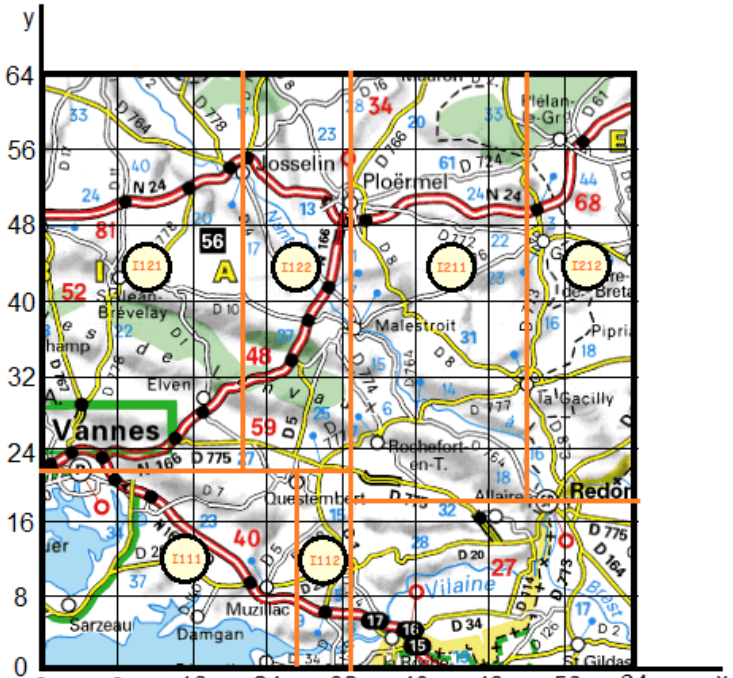
(avec la collaboration discrète et implicite de Michelin)

Les deux descendants de "Ploermel", "Redon" et "Vannes" sont au niveau 1. On peut considérer que "Redon" est représentatif de la partie rectangulaire I2 de droite ($x \geq 33$) et que "Vannes" est représentatif de la partie rectangulaire I1 de gauche ($x < 33$). Ces deux nœuds permettent une partition supplémentaire par tracé des droites $y = 18$ (I21, I22) et $y = 22$ (I11 et I12).



(avec la collaboration discrète et implicite de Michelin)

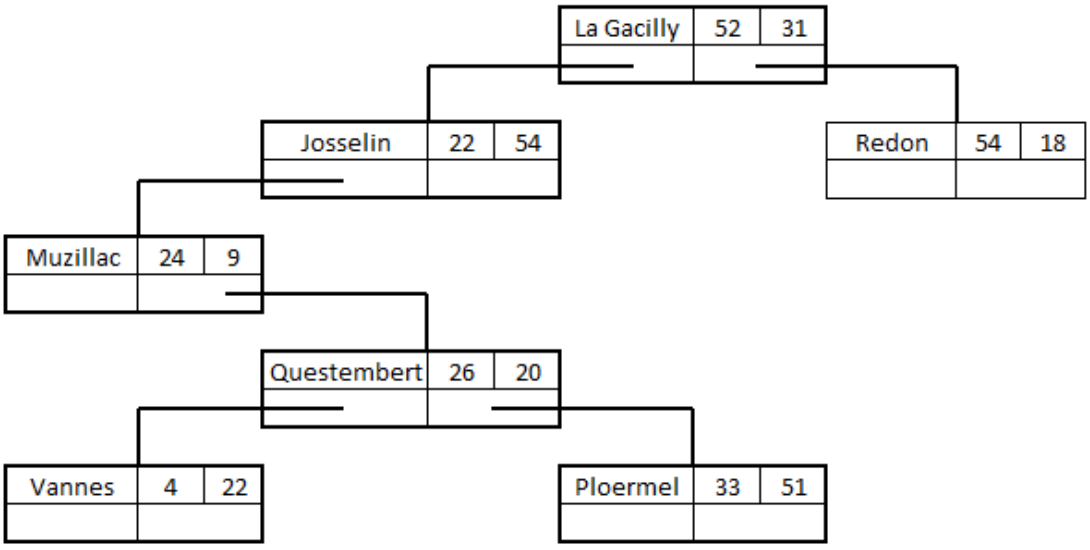
Au niveau 3, on a trois nœuds "Questembert", "Josselin" et "La Gacilly". "Questembert" et "Josselin" sont représentatifs respectivement des régions I11 et II2. "La Gacilly" est représentative de la région I22. Aucune ville ne représente la région I21.
Par chacun des points "Questembert", "Josselin", "La Gacilly", on trace des droites verticales d'équations respectives $x = 26$, $x = 22$, $x = 52$, ce qui conduit à un partitionnement supplémentaire I111, I112, I211, I212, I221, I222.



(avec la collaboration discrète et implicite de Michelin)

Il ne reste plus que le nœud "Muzillac" de niveau 3, représentatif de la région I111. On voit donc que l'arbre 2-d Tree correspond à une partition de l'image initiale dont les parties sont de dimensions différentes.

Il faut noter que la construction de l'arbre (et donc le partitionnement de l'image) dépend de l'ordre de la liste des points-villes. Si on renversait la liste précédente, on obtiendrait un 2-d Tree différent :



Quad-trees

Dans la méthode des 2-d trees, la partition de l'image s'effectue en traçant soit une ligne verticale (pour les nœuds de niveau pair) soit une ligne horizontale (pour les nœuds de niveau impair). Et chaque nœud possède deux pointeurs @gauche et @droite.

Dans la méthode des quadtrees ou arbres quaternaires, la partition s'effectue en traçant simultanément in ligne verticale et une ligne horizontale, définissant ainsi quatre régions. En conséquence, chaque nœud sera muni de 4 pointeurs @no, @ne, @so, @se, (vers nord-ouest, vers nord-est, vers sud-ouest, vers sud-est) donc chaque nœud peut avoir au plus 4 descendants :

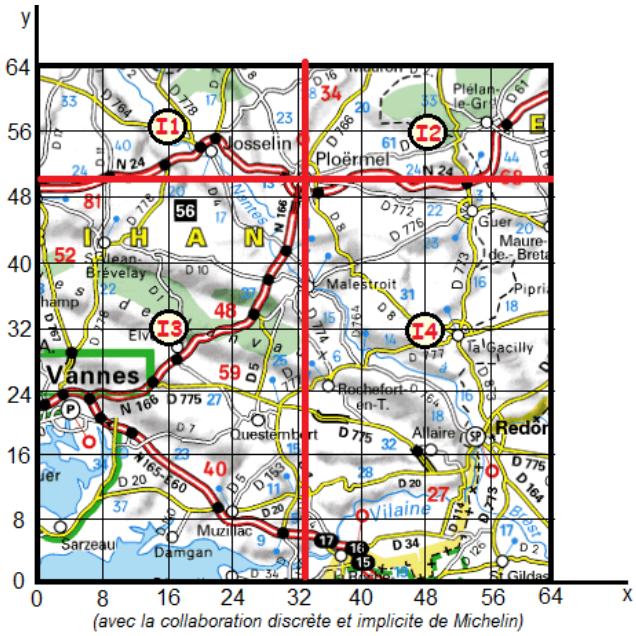
information		x	y
@no	@ne	@so	@se

Appliquons la méthode en reprenant la liste des villes précédentes.

Placement de "Ploërmel" : c'est le premier nœud, racine du quadtree

A partir du point représentatif de ce nœud, on partage l'image en 4 parties

Ploërmel	33	51	
@no	@ne	@so	@se

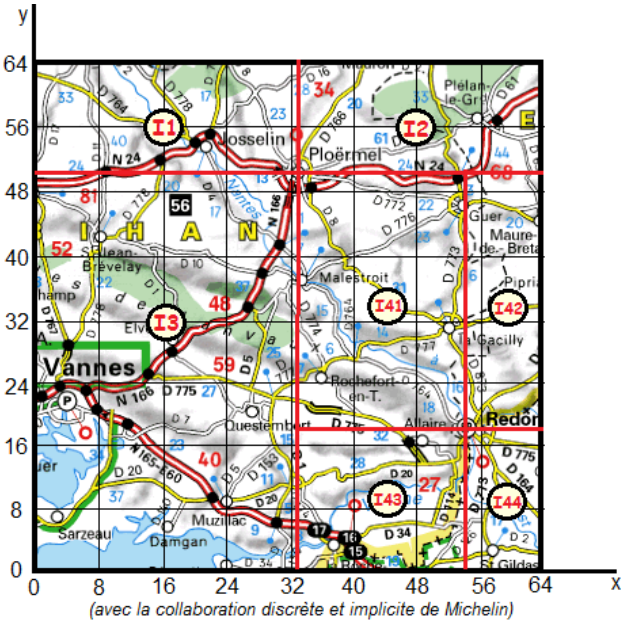


(avec la collaboration discrète et implicite de Michelin)

Placement de la deuxième ville, Redon (54, 18). Cette ville se trouve dans la région I4 (sud-est). On placera le nœud correspondant comme pointé par @se :

Ploërmel	33	51	
@no	@ne	@so	@se

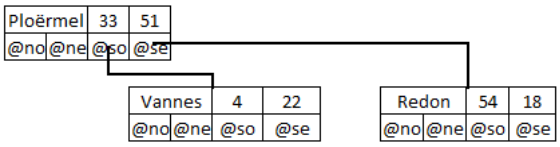
Redon	54	18	
@no	@ne	@so	@se



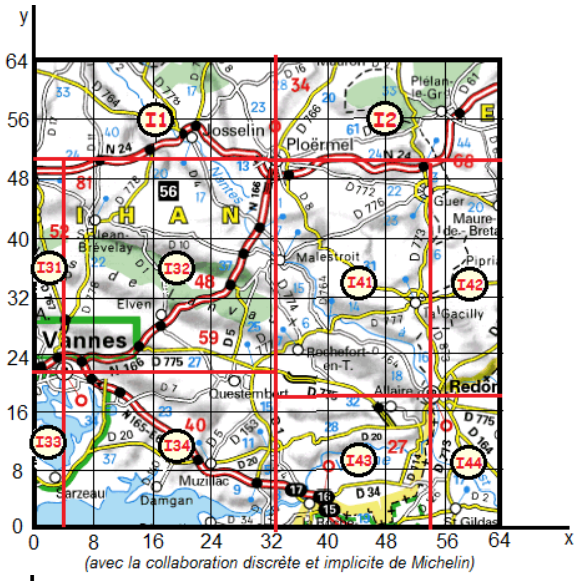
(avec la collaboration discrète et implicite de Michelin)

Le point correspondant sur la carte permet de subdiviser I4 en 4 parties I41, I42, I43, I44

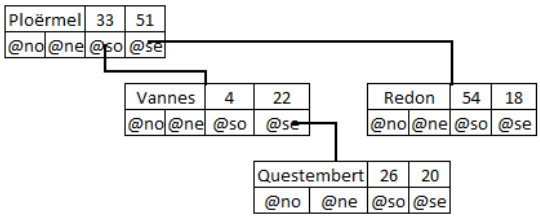
Placement de la troisième ville, Vannes (4,22). Cette ville se trouve dans la région I3. On plera donc le nœud correspondant comme descendant du nœud "Ploermel" via le pointeur @so :



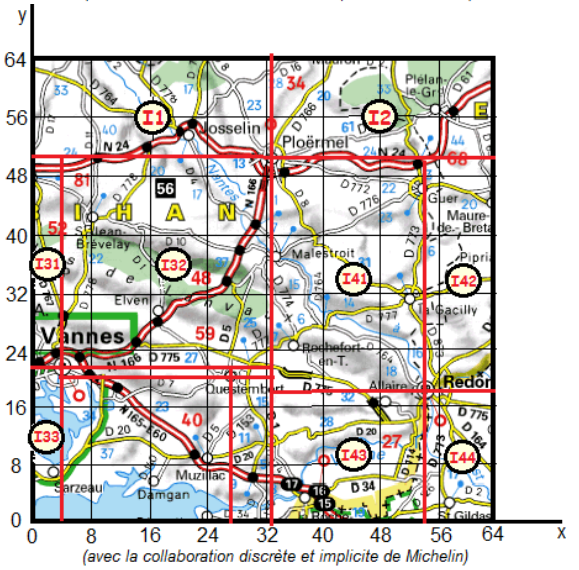
A partir du point "Vannes", on partage I3 en 4 régions I31, I32, I33, I34.



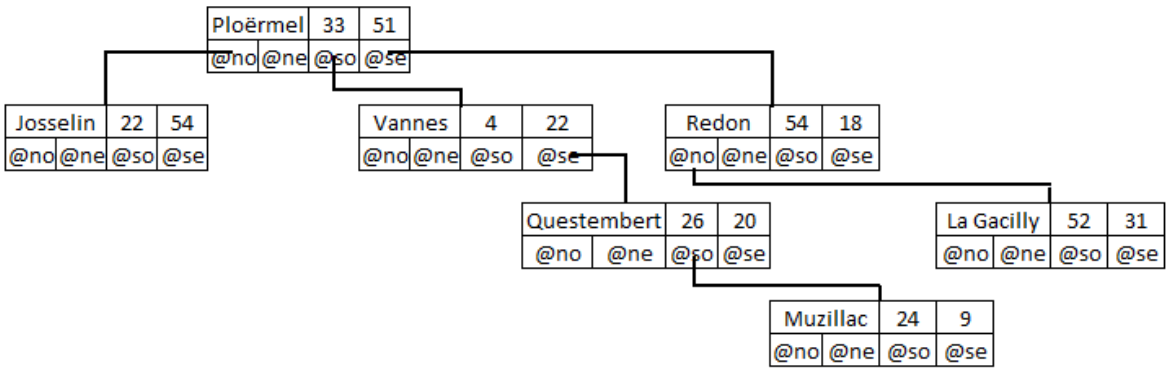
Placement de la quatrième ville, Questembert (26, 20). Cette ville est située dans la région I34, au sud est de Vannes. Le nœud correspondant sera donc un descendant de Vannes via le pointeur @se :

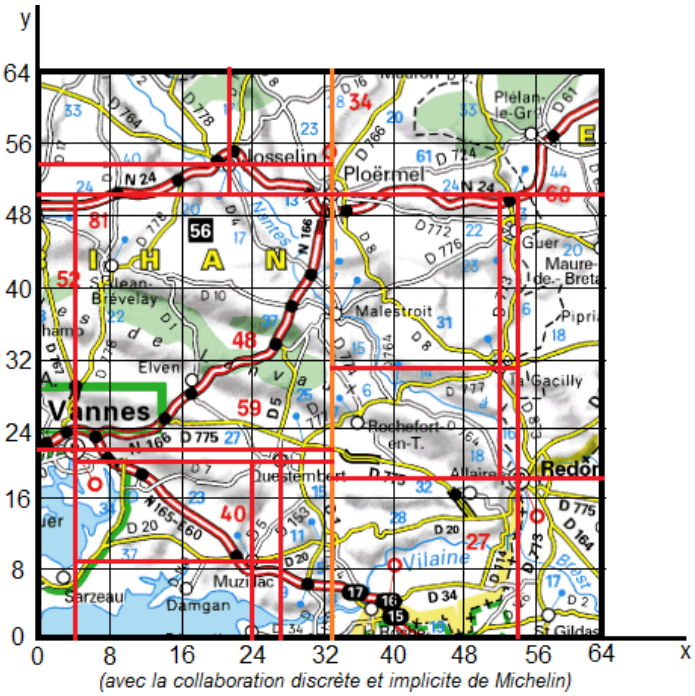


Le point correspondant de la carte partage à son tour le région I34 en 4 sous-régions.



En continuant ainsi pour les villes restantes, on aboutit à l'arbre suivant et au découpage correspondant de la carte :





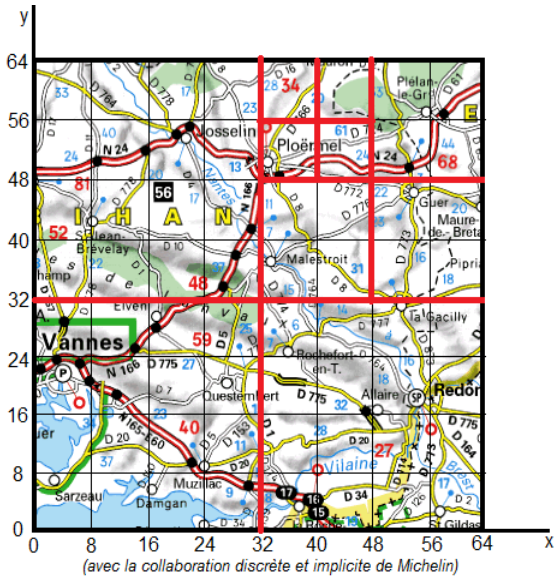
(avec la collaboration discrète et implicite de Michelin)

Cette méthode présente le même inconvénient que le 2-d Tree : si l'on change l'ordre des villes on obtient un arbre et un découpage d'image différents.

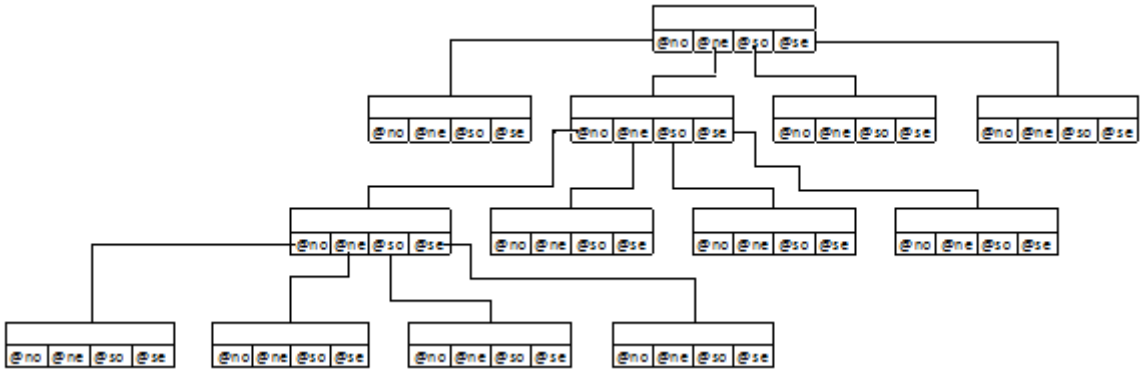
MX-QuadTrees

Comme on a pu l'observer dans les deux types d'arbres précédents, le découpage géographique en région est "irrégulier" et dépend de l'ordre de placement des points dans la liste. Suivant l'ordre utilisé, l'arbre peut être plus ou moins "haut" ou "profond".

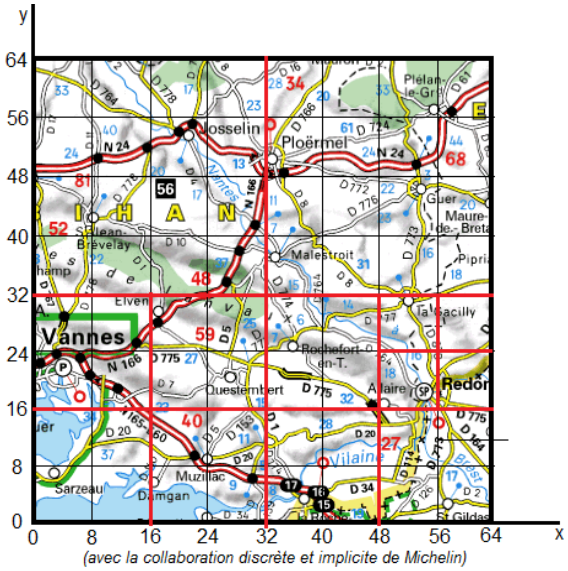
Les arbres MX-Quadtrees correspondent au contraire à un découpage régulier mais nécessitent de partir d'une image carrée dont les dimensions sont $2^k x 2^k$. Le choix de k influe sur la granularité du découpage. Le nœud racine correspond à l'image toute entière. Celle-ci est divisée en 4 parties égales en traçant les droites $x = 2^k/2 = 2^{k-1}$ et $y = 2^k/2 = 2^{k-1}$. Chacune de ces 4 parties correspond à un nœud descendant du nœud racine. A leur tour, elles sont divisées en 4 parties et ainsi de suite :



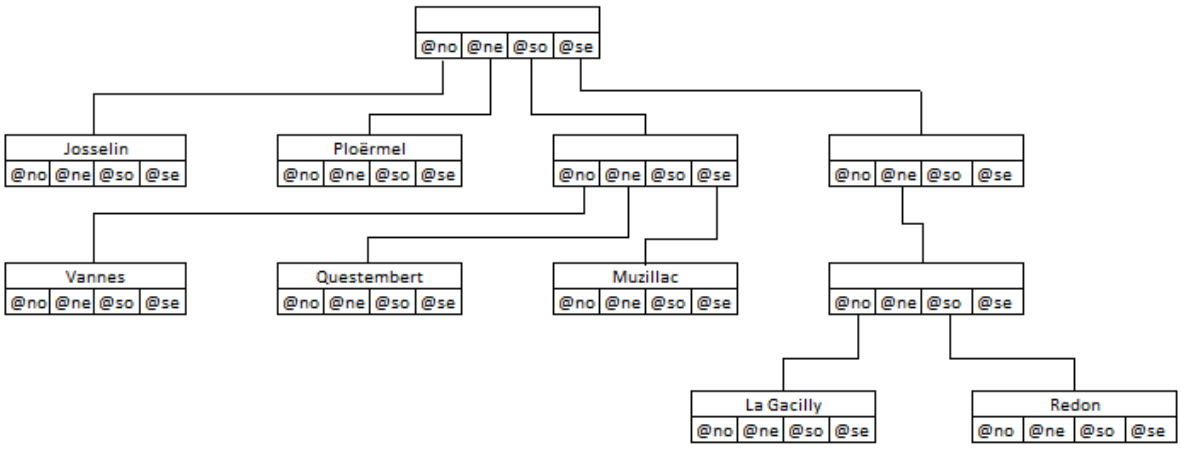
(avec la collaboration discrète et implicite de Michelin)



Pour localiser les villes utilisées dans la liste précédente, il faudra effectuer le découpage suivant :



(avec la collaboration discrète et implicite de Michelin)



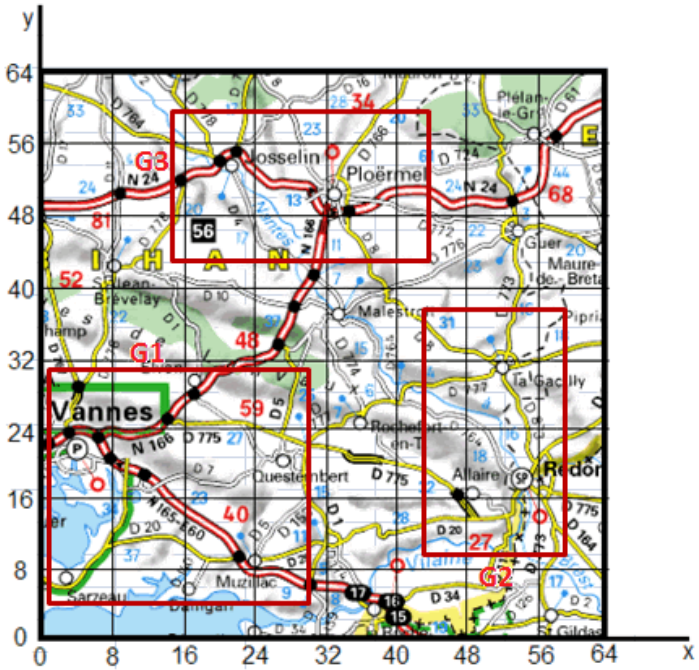
Implicitement dans le schéma précédent, nous avons restreint les régions à la présence d'un point. Par exemple Josselin est dans une grande région par rapport à Redon. Si on voulait ajouter la ville Saint-Jean Brevelay, il faudrait alors découper la région de Josselin en 4 parties de manière à ce qu'une région ne corresponde qu'à une seule ville. Bien entendu, on pourrait effectuer le découpage total en carrés de côté $2^0 = 1$, mais cela correspondrait à stocker des nœuds vides et gaspillerait de la place mémoire.

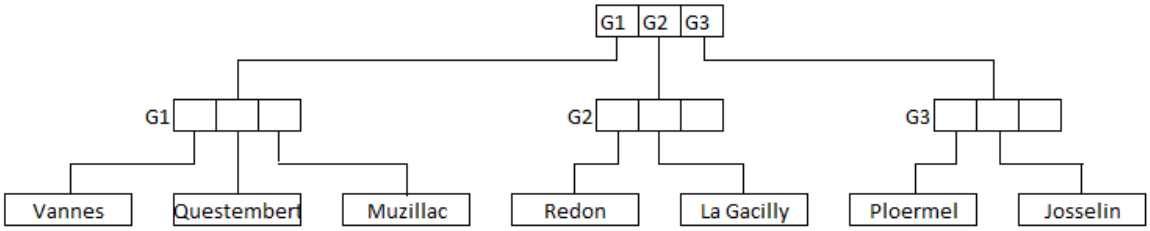
Le codage d'un MX-Quad-Tree est assez simple. Il faut d'abord se fixer une règle : NO = 0, NE = 1, SE = 2, SO = 3. Il suffit alors de mettre à la suite les numéros de quadrants pour obtenir le codage.

On normalise ensuite en mettant X là où il n'y a pas de point significatif. Pour l'exemple précédent, on obtient : 0X (Josselin), 1X (Ploermel), 30X (Vannes), 31X (Questembert), 32 (Muzillac), 210 (La Gacilly), 213 (Redon).

R-Trees

Les arbres R-Trees sont très différents des précédents. Ils consistent à stocker des rectangles pouvant contenir à leur tour d'autres rectangles. Les feuilles de l'arbre sont des rectangles. Les nœuds intermédiaires sont des groupes de rectangles.





Les R-trees obéissent toutefois aux règles suivantes :

- chaque nœud doit être au moins à moitié plein, c'est à dire avoir entre K/2 et K descendants où K est l'ordre fixé de l'arbre.
- la racine et les feuilles peuvent faire exception à la règle précédente
- l'arbre est équilibré (les données sont au même niveau).

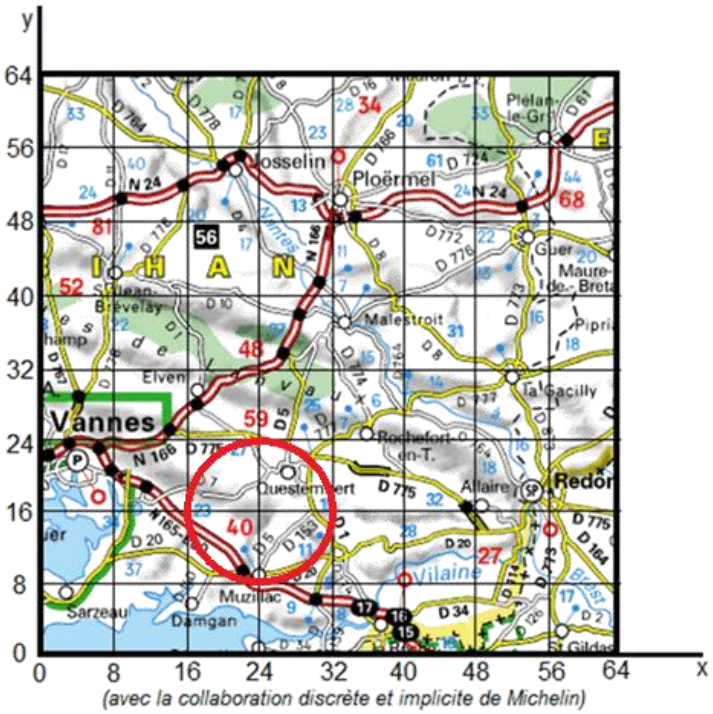
L'arbre du schéma précédent est d'ordre K = 3.

Maintenance des arbres

Nous examinons ici comment effectuer des recherches dans un arbre comme ceux des types précédents. La mise à jour des structures précédentes constitue, par ailleurs, un point crucial de choix. Par mise à jour on entend ici l'insertion et la suppression de nœuds.

k-d Trees

Le recherche d'un nœud s'effectue simplement en suivant les règles qui ont permis de construire l'arbre. Toutefois, le type de recherche le plus intéressant est celui où on veut sélectionner les points qui se trouvent à une distance inférieure ou égale à une valeur r d'un point donné (x_0, y_0). Par exemple, on recherche les villes qui se trouvent à proximité du point (24,16) dans un rayon de 8 unités. Elles se trouvent dans le cercle rouge ci-dessous : on constate que Questembert et Muzillac répondent à la requête.



(avec la collaboration discrète et implicite de Michelin)

Pour arriver de manière automatique à ce résultat, on est amené à compléter la structure d'un enregistrement en ajoutant les bornes des régions représentées par les points. On ajoutera ainsi dans chaque nœud N, $x_{min}(N)$, $x_{max}(N)$, $y_{min}(N)$, $y_{max}(N)$. Par exemple pour Redon et Vannes, on aura les enregistrements suivants :

Questembert		26	20
$x_{min} = 0$	$x_{max} = 33$	$y_{min} = 0$	$y_{max} = 22$
@gauche		@droite	

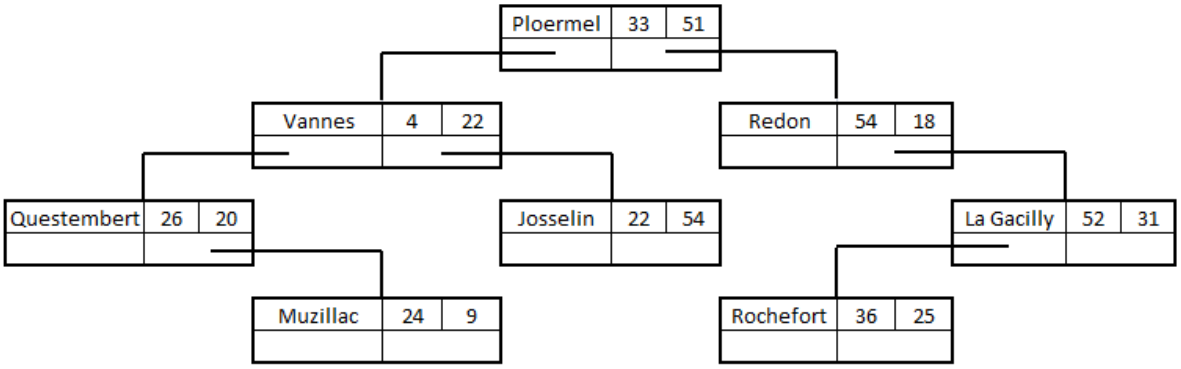
Muzillac		24	9
$x_{min} = 0$	$x_{max} = 26$	$y_{min} = 0$	$y_{max} = 22$
@gauche		@droite	

En se basant sur ces paramètres supplémentaires, la requête consiste à rechercher les intersections des régions représentées par les nœuds avec le cercle de rayon r. On parcourt le 2-d Tree pris comme exemple initial en commençant par la racine (Ploërmel) qui ne correspond pas à un point du disque d'équation $(x - 24)^2 + (y - 16)^2 \leq 64$. On passe ensuite à ses descendants :

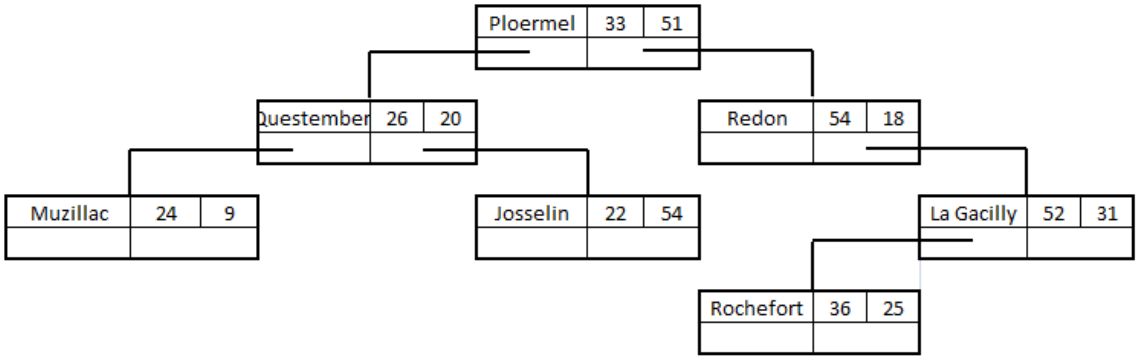
- Redon représente la région x_{min} = 33, x_{max} = 64, y_{min} = 0, y_{max} = 64. Le cercle est inscrit dans un carré x_{min} = 16, x_{max} = 32, y_{min} = 8, y_{max} = 24. Il n'y a donc pas d'intersection possible.
- Vannes représente la région x_{min} = 0, x_{max} = 33, y_{min} = 0, y_{max} = 64. Il y a ici une intersection notable puisque le disque est contenu dans la région. On vérifie que Vannes (4, 22) n'est pas dans le disque. On passe donc aux descendants de Vannes :
    - Josselin représente la région x_{min} = 0, x_{max} = 33, y_{min} = 22, y_{max} = 64. Il y a intersection mais on vérifie que Josselin n'est pas dans le disque. Josselin n'a pas de descendant
    - Questembert représente la région x_{min} = 0, x_{max} = 33, y_{min} = 0, y_{max} = 22. Il y a intersection. On vérifie que Questembert est bien dans le disque. Questembert a un descendant :
        - Muzillac représente la région x_{min} = 0, x_{max} = 26, y_{min} = 0, y_{max} = 22. Il y a intersection. On vérifie que Muzillac est bien dans le disque. Muzillac n'a pas de descendant.

Le résultat de la requête est donc {Questembert, Muzillac}.

L'insertion d'un nouveau nœud ne pose pas de problème. Par exemple ajoutons la ville Rochefort-en-Terre (36, 25). En comparant les x et les x aux nœuds présents, on constate que le nouveau nœud prend sa place comme descendant gauche de La Gacilly :



La suppression est plus complexe lorsque le nœud n'est pas une feuille de l'arbre, mais un nœud intermédiaire dans la mesure où ce nœud peut avoir des descendants qu'il faut, eux, conserver. Ces descendants, et leurs descendants, constituent des sous-arbres qu'il va falloir raccrocher à des nœuds restants. Supposons que l'on supprime le nœud "Vannes". Nous avons à replacer deux sous-arbres, celui qui a pour racine "Questembert" et celui qui a pour racine "Josselin". L'arbre de gauche, de racine "Questembert", sera raccroché à Ploërmel à gauche. On ne peut évidemment faire de même avec le sous-arbre de droite constitué du seul nœud "Josselin".Il faut réorganiser l'arbre ce qui est couteux en temps :



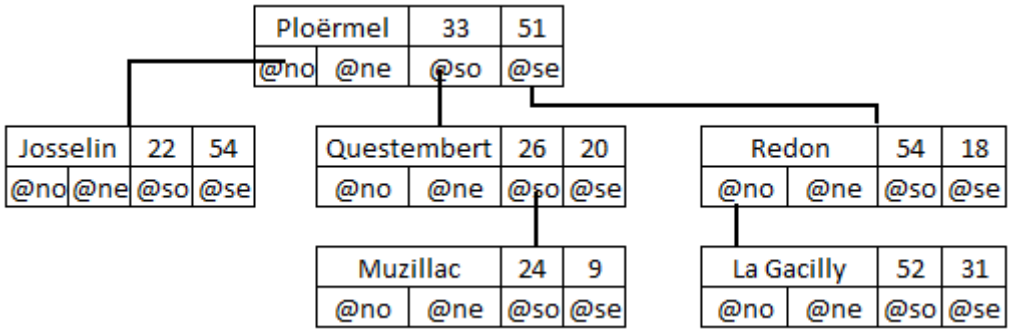
QuadTrees

La recherche de points au voisinage d'un point donné se traite de la même façon que pour les 2-d Trees : on élimine les sous-arbres dont la racine représente une région qui n'a pas d'intersection avec le disque cible. Plus précisément on utilise l'algorithme récursif suivant :

```
procedure Recherche( T nœud, C cercle)
{
    Si region(T)∩C = ∅
       alors stop;
       sinon
            si (x(T), y(T) ∈ C alors lister le nœud;
            procedure Recherche (@no(T), C);
            procedure Recherche (@ne(T), C);
            procedure Recherche (@so(T), C);
            procedure Recherche (@se(T), C);
    FinSi
}
```

Pour l'insertion, le procédé a été vu lors de la construction d'un arbre.

Pour la suppression d'un nœud, le procédé est aussi délicat que pour le cas des 2-d Trees. Par exemple si l'on veut supprimer Vannes de l'arbre obtenu précédemment, on aura la reconstruction suivante :



MX-Quadtrees

Nous avons vu l'insertion des nœuds en construisant précédemment l'arbre.

La recherche de localisation de villes par rapport à un point donné est analogue aux cas précédents avec deux différences :

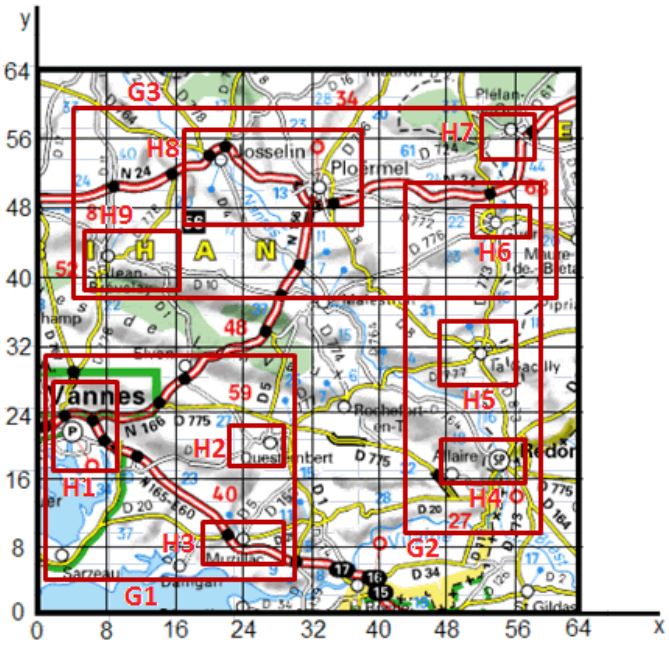
- les valeurs de xmin, xmax, ymin, ymax sont fixées par le découpage.
- les nœuds sont tous au niveau des feuilles de l'arbre et le calcul, permettant de savoir si un point est dans le cercle, ne s'applique donc qu'aux feuilles.

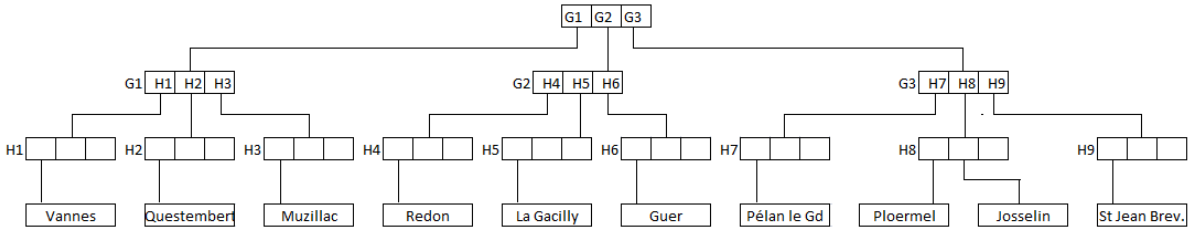
Cette dernière remarque explique aussi pourquoi la suppression est ici ultra simple : les nœuds représentatifs des points n'ont pas de sous-arbre. On peut donc les supprimer sans avoir à reconstruire l'arbre.

R-Trees

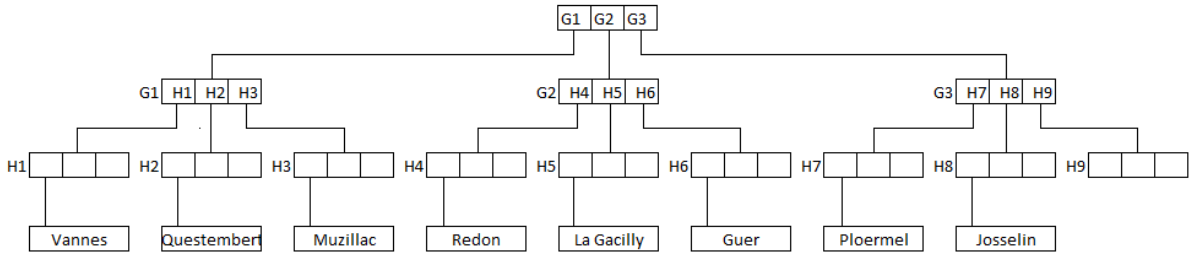
La situation est ici radicalement différente des précédentes.

Pour l'insertion d'un nouveau rectangle, il faut tenir compte de la règle suivant laquelle un rectangle doit contenir au moins K/2 rectangles et au plus K rectangles. Cela veut dire que lorsqu'il y a encore de la place dans un rectangle on peut incorporer éventuellement le nouveau rectangle. Par exemple ajoutons Plélan le Grand, Guer et Saint-Jean Brévelay. Il n'y a que deux places disponibles. Il faut donc ajouter un niveau et répartir les villes de manière à ce qu'un nœud soit au moins à moitié rempli. On pourra adopter le découpage suivant qui correspond à l'apparition d'un nouveau niveau (pour k = 3).





La suppression doit aussi tenir compte de la règle de remplissage : entre K/2 et K rectangles. Par exemple si l'on supprime Plélan le Grand et Saint jean Brévelay, le rectangle G3 ne contient plus assez de rectangles intérieurs. Il faut alors reconstruire l'arbre (la règle n ≥ k/2 ne s'applique pas aux feuilles) en essayant de lui donner un aspect équilibré.



Ch3.3 : Données médias

Données images

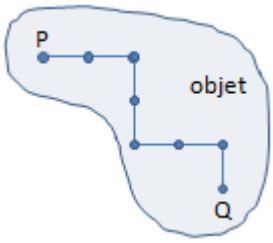
Description d'une image

Une image contient des objets qui peuvent être la cible de recherches faites dans une collection d'images. A un objet on peut faire correspondre

- le descripteur de région qui indique dans quelle région de l'image se situe l'objet
- le descripteur de caractéristiques qui fournit, au plus bas niveau, les caractéristiques des pixels composant l'objet : position (x, y), couleur (R,V,B) et au plus haut niveau des indications relatives à la composition en couleur comme l'histogramme de l'objet, à la texture ou aux formes.

Un objet comportant beaucoup de pixels, il peut être utile de travailler par groupes de pixels ou cellules obtenues par un quadrillage de l'image. Si une image est de dimension a x b, une cellule aura les dimensions m x n telles que a mod m = 0 et b mod n = 0. Une cellule contient donc m x n pixels. La division de l'image en cellules définit la résolution de l'image. Dans ce cas le descripteur de caractéristiques de bas niveau portera sur les cellules et non plus sur les pixels ; il portera sur moins de données : position moyenne de la cellule, couleur moyenne de la cellule.

Un objet est un ensemble de points (ou de cellules) tel que si P et Q sont deux points (ou cellules) de l'objet, on peut passer de P à Q par incrémentation des coordonnées..



Si l'objet est rectangulaire, la définition est plus simple : P(x,y) ∈ objet si $x_{min} \leq x \leq x_{max}$ et $y_{min} \leq y \leq y_{max}$.

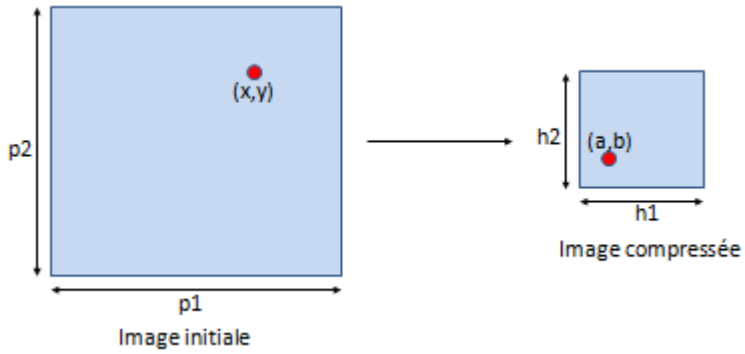
Une base de données image, compte tenu des définitions précédentes, sera caractérisée par

- un ensemble d'images, elles-mêmes décomposées en cellules mxn
- chaque cellule est définie par ses caractéristiques
- chaque image contient des objets, que l'on supposera inscrits dans des rectangles.
- un objet possède des caractéristiques globales

Compression des images

Une image comporte un nombre important de pixels ou même de cellules. Il est utile de réduire le nombre de données à manipuler en compressant une image par un algorithme de transformation qui

fera passer d'une image de dimensions (p1, p2) à une image compresées de dimension (h1, h2). L'algorithme de transformation ne doit pas demander un temps de calcul trop long.



Citons deux méthodes usuelles d'obtention d'images compressées :

- la transformée de Fourier discrète DFT

$$C(a,b) = DFT(x,y) = \frac{1}{p1p2}\sum_{x=0}^{p1-1}\sum_{y=0}^{p2-1} I(x,y)e^{-i(\frac{2\pi xa}{p1}+\frac{2\pi yb}{p2})}$$

- la transformée en cosinus discrète DCT

$$C(a,b) = DCT(x,y) = \frac{1}{\sqrt{p1p2}}\alpha(x)\alpha(y)\sum_{x=0}^{p1-1}\sum_{y=0}^{p2-1} I(x,y)cos\frac{(2x+1)\pi a}{2p1}cos\frac{(2y+1)\pi b}{2p2}$$

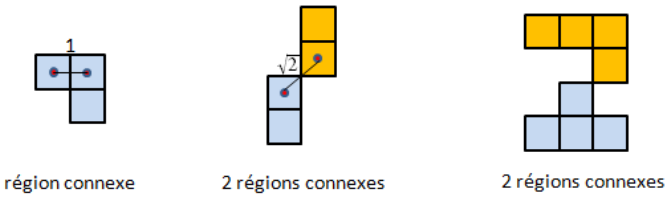
$$\text{avec } \alpha(x),\alpha(y) = \frac{1}{\sqrt{2}} \; si\; x = 0, y = 0 \text{ et 1 sinon.}$$

Ces deux transformations sont inversibles. toutefois, les algorithmes de compression comprennent généralement d'autres traitements comme la quantification qui n'est pas une opération inversible. La transformation inverse ne redonnera pas l'image initiale mais une image avec perte d'information. Une autre propriété des DFT et DCT est la conservation de la distance euclidienne.

Segmentation

Revenons sur le concept de région dans une image. Comment détermine-t-on ces régions ? Elles peuvent être obtenues par un procédé de segmentation (on les appelle alors des segments) basé sur une propriété d'homogénéité (par exemple, pixels dont le niveau de rouge est compris entre 0 et 127 dans une échelle allant de 0 à 255).

Pour comprendre le procédé de segmentation, il faut définir ce que l'on entend par région connexe. C'est un ensemble de cellules tel que la distance entre deux cellules contigues (partageant une arête) de cet ensemble est 1.



Une propriété d'homogénéité utilisée dans le procédé de segmentation fournit une réponse booléenne : oui ou non. On pourra alors définir un segment de la manière suivante :

1 - On part d'un point (cellule) qui vérifie la propriété d'homogénéité (réponse : oui). Il constitue le départ d'un segment R.
2 - On cherche si ses voisins Nord, Est, Sud, Ouest vérifie la propriété d'homogénéité. Si oui ils sont ajoutés à R.
3 - On recommence 1 avec l'un des nouveaux points de R. Quand on ne trouve plus de points, on a définit un segment R

En partant d'un autre point non sélectionné, on peut ainsi définir un autre segment et ainsi de suite jusqu'à ce que tous les points de l'image aient été sélectionnés.

exemple :

4	220	78	56	240
3	44	68	82	210
2	210	142	160	156
1	12	57	91	130
	1	2	3	4

Propriété d'homogénéité : niveau de rouge compris entre 0 et 127 (les niveaux sont indiqués sur la figure)

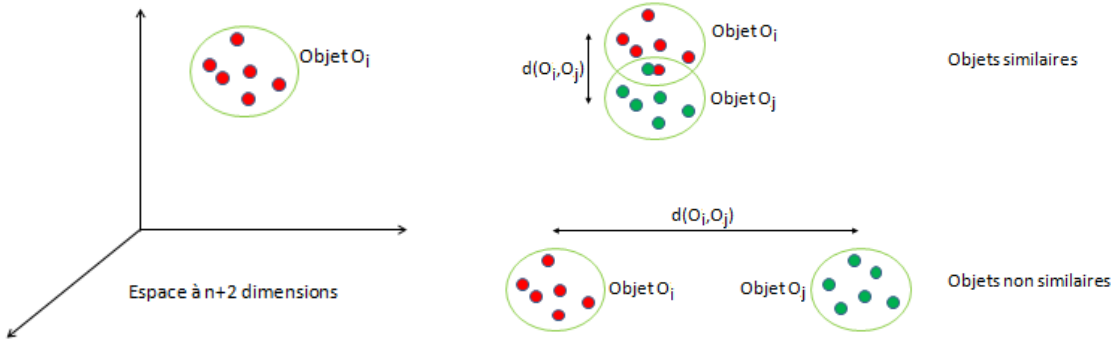
4 segments :
R1 = {(1,1), (2,1), (3,1)}
R2 = {(1,2), (2,2), (3,2), (4,1), (4,2), (4,3), (4,4)}
R3 = {(1,3), (2,3), (2,4), (3,3), (3,4)}
R4 = {(1,4)}

Les segments obtenus forment une **partition** de l'image : leur réunion donne l'image complète et les segments sont disjoints deux à deux.

Recherche de similarité

Comment, dans une base de données images, trouver les images qui "ressemblent le plus" à une image donnée ? Il y a deux démarches générales pour résoudre cette requête : la démarche métrique (très utilisée) et la démarche par transformation.

Dans la démarche métrique, on considère un ensemble W d'objets Oi ayant les caractéristiques de cellules p1, p2, ...pn. Une cellule j d'un objet Oi aura les caractéristiques (xj, yj, v1j, v2j, ..., vnj). Si l'objet Oi est de dimansions w x h (on le supposera rectangulaire), il y a wh caractéristiques qui correspondent chacune à un point dans un espace à n+2 dimensions. un objet est donc représenté par un ensemble de points dans cet espace.



Dans cet espace, on définit une distance. Rappelons qu'une distance d(x,y) est un nombre réel non négatif satisfaisant les propriétés suivantes

d(x,y) = d(y,x)
d(x,x) = 0
d(x,y) ≤ d(x,z) + d(z,y)

Il y a plusieurs façons de définir une distance. La plus connue est la distance euclidienne définie à partir du théorème de Pythagore $d^2 = a^2 + b^2$. Appliquons la distance euclidienne à des objets en couleur dont les propriétés caractéristiques sont le niveau de rouge, le niveau de vert et le niveau de bleu. Pour un objet Oi, on aura donc wh caractéristiques du type (x, y, nivR(x,y), nivV(x,y), nivB(x,y)) où nivX(x,y) prennent des valeurs de 0 à 255. En supposant que la dimension d'un objet est w x h, la distance entre deux objets Oi et Oj pourra être définie par

$$d(O_i,O_j) = \sqrt{\sum_{x=0}^{w-1}\sum_{y=0}^{h-1}[(niv_i^R(x,y) - niv_j^R(x,y))^2 + (niv_i^V(x,y) - niv_j^V(x,y))^2 + (niv_i^B(x,y) - niv_j^B(x,y))^2]}$$

Evidemment un tel calcul peut être long. C'est pourquoi on utilise généralement une fonction fe d'extraction de caractéristiques qui vise à remplacer l'ensemble de points (dans un espace à n+2 dimensions) représentant un objet par un seul point dans un espace à s dimensions (s << n+2). C'est une forme de compression où les transformées DFT et DCT peuvent être utilisées puisqu'elles conservent la distance euclidienne.

Dans la démarche de transformation, on applique le principe suivant consistant à transformer un objet pour le faire ressembler à un autre et à mesurer le "coût" correspondant. Plus le coût est élevé plus les objets concernés sont dissemblables. Une transformation peut être une combinaison de transformations élémentaires : translation, rotation, réduction, agrandissement mais aussi extension (ajout de formes), excision (suppression de formes), coloration (changement de couleur).

$$O_1 \xrightarrow{T_1} O_2 = T_1(O_1) \xrightarrow{T_2} O_3 = T_2(O_2) \xrightarrow{T_3} O_4 = T_3(O_3) \qquad coût = \sum_i coût(T_i)$$

Il peut y avoir plusieurs chaînes de transformations qui mènent au même résultat. Dans ce cas, la mesure de la dissimilarité est le coût minimum des coûts de T(O, O')UT(O', O) où T(O,O') est l'ensemble de toutes des transformations qui font passer de l'objet O à l'objet O'.

Reamarquons que

- l'utilisateur peut définir les transformations élémentaires à effectuer dans le sens où elles l'arrangent
- l'utilisateur doit associer un coût à chaque transformation

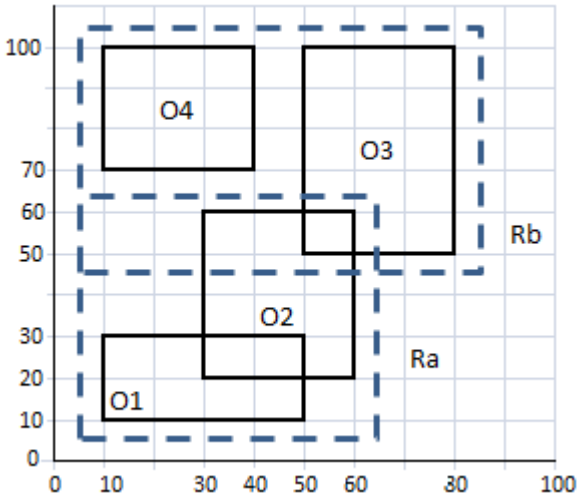
Représentation par des arbres R

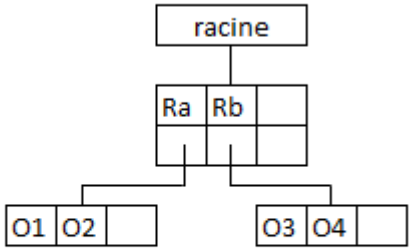
On peut indexer une base de données images avec des arbres R. ceci est justifié quand les objets sont délimités par des rectangles. On commence par créer une table relationnelle OBJET

OBJET

idobjet	image	xmin	xmax	ymin	ymax	libelle
'O1'	image1.png	10	50	10	30	'photo de Jojo'
'O2'	image1.png	30	60	20	60	'photo de Claudine'
'O3'	image2.png	50	80	50	100	'photo de Alfred'
'O4'	image3.png	10	40	70	100	'photo de Jules'

Bien que les objets délimités par des rectangles soient dans des images différentes, la construction de l'arbre R s'effectue comme si ils étaient dans la même image :





arbre R d'ordre 3

A partir des coordonnées des objets dans la table OBJET, on peut retrouver un objet en circulant dans l'arbre R depuis la racine.

Données vidéos

Structure d'une vidéo

On sait qu'une vidéo numérique se compose

- d'une séquence d'images fixes
- d'un son synchronisé sur les images

Nous ne considérerons pas ici le problème de la synchronisation du son avec les images. Nous ferons comme si la vidéo était muette. Nous reviendrons sur la question du son dans la partie suivante.

Le standard le plus utilisé en vidéo appartient à la famille MPEG. Dans MPEG-1, la séquence d'images comporte trois types d'images I, P, B :

I	B	B	P	B	I	B	P	B	B	I

I est une image "intra" obtenue par enregistrement direct de la caméra ; P est une image calculée à partir de l'image I qui la précède ; B est une image calculée par interpolation à partir des images P ou I les plus proches. Les images sont dans un format compressé (JPEG).

Quoi qu'il en soit, on peut considérer une vidéo comme une suite d'images numérotées dans le sens de la visualisation, ce qui nous suffira dans ce qui suit.

1	2	3	4	5	6	7	8	9	10	11

Description d'une vidéo

Une vidéo comporte, comme les images, des **objets** (personnes ou choses). Le fait nouveau ici est qu'elle comporte aussi des **activités**. Prenons l'exemple suivant :

la voiture sort du garage et roule sur la route

images 1 à 99

la voiture fait demi-tour

images 100 à 199

la voiture rentre au garage

images 200 à 299

Sur cet exemple, les objets peuvent être : le garage, la voiture. Les activités peuvent être "sortie du garage", "demi-tour", "rentrée au garage".

Un objet possède des caractéristiques comme dans le cas des images. cependant on peut distinguer les caractéristiques indépendantes des images de la vidéo et des caractéristiques dépendantes des images de la vidéo. Un schéma d'objet sera donc de la forme (cd, ci) où cd désigne les caractéristiques indépendantes et ci les caractéristiques dépendantes. Une caractéristique ne peut être à la fois indépendante et dépendante. Derrière cd et ci il y a en fait une série de caractéristiques ou propriétés repérées par un nom et une valeur. Un objet particulier est alors défini par un triplet (oid, os, prop). oid est l'identifiant unique de l'objet qui permet de le discerner des autres objets. os est un schéma d'objet. prop qui correspond aux propriétés de l'objet suivant le schéma os peut être défini de la manière suivante : pour chaque propriété ci, il y a au moins une propriété valuée nomprop = valeurprop ; pour chaque propriété cd, il y a au moins une propriété de la forme nomprop=valeurprop IN f où f est le numéro d'une image de la vidéo. Dans notre exemple, on aura

images	oid	objet	ci	cd
1 à 49	111	voiture	couleur=verte marque : Renault	état = sort du garage IN 1,49
	222	garage	porte = grise	porte = ouverte IN 1, 49
50 à 99	111	voiture	couleur=verte marque : Renault	état = roule en s'éloignant IN 50, 99
	222	garage	porte = grise	porte = fermée IN 50,99
100 à 199	111	voiture	couleur=verte marque : Renault	état = fait demi-tour IN 100,199
	222	garage	porte = grise	porte = fermée IN 100, 199
200 à 249	111	voiture	couleur=verte marque : Renault	état = roule en revenant IN 200, 249
	222	garage	porte = grise	porte = fermé IN 200, 249
250 à 300	111	voiture	couleur=verte marque : Renault	état : entre au garage IN 250, 300
	222	garage	porte = grise	porte = ouverte IN 250, 300

Passons maintenant aux activités en adoptant une démarche proche de la précédente. Une activité se réfère à un schéma d'activité qui donne une liste de propriétés comme (nomprop, valeur). Une activité sera alors définie par une suite de paires (nomprop, valeur). Dans notre exemple on pourra avoir pour l'activité "circuler" : (entrer-sortir, voiture), (rouler, voiture), (faire_demi_tour, voiture).

Résumons-nous. Supposons que la vidéo comporte q images. Le **contenu** de la vidéo est une série de triplets (obj, act, mapping) où obj est un objet, act une activité et mapping la correspondance entre une image de la vidéo et un couple objet-activité. Exemple de triplet : (111, rouler, 75) ce qui signifie que dans l'image 75 la voiture d'oid 111 roule. Bien entendu comme les vidéos comporte un très

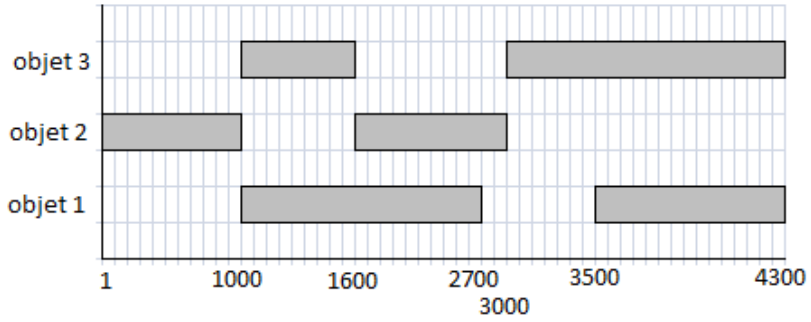
grand nombre d'images, cela fait beaucoup de triplets. On étudiera plus loin ce problème. Une base de données vidéo pourrait alors avoir la forme suivante

contenu	idvidéo	nom_vidéo	numéro image	annotations	stockage
pointeur	id	nom_fichier	num	texte	*pointeur*

où contenu est un pointeur vers un contenu comme défini ci-dessus (série de triplets), idvidéo un identifiant repérant une vidéo particulière, nom_vidéo, le nom du fichier image, numéro image, le numéro de l'image correspondant au contenu, annotations des métadonnées (auteur, sujet, date enregistrement, droits, ...) et stockage un pointeur vers les structures de stockage (disques, CD, bande, ...).

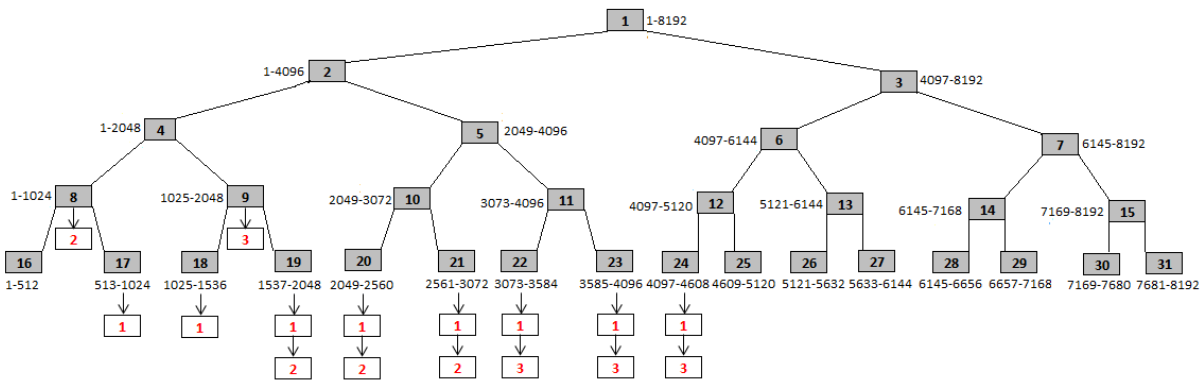
Structures d'indexation

Comme on le sait, le nombre d'images d'une vidéo est très grand. c'est pourquoi il vaut mieux parler de "morceaux" de vidéo ou **segments** en remplaçant le triplet (obj, act, mapping) par un triplet où mapping n'est plus la correspondance entre une image et une couple objet-activité mais une correspondance entre une séquences d'images et un couple objet-activité. Pour indexer ce nouveau contenu, on peut faire appel aux arbres binaires. Négligeons pour l'instant les activités et supposons que nous avons dans le temps, image par image, l'apparition/disparition des objets 1, 2, 3 ci-dessous :



objet	segment
1	1000 - 2700
1	3500 - 4300
2	1 - 999
2	1600 - 2999
3	1000 - 1599
3	3000 - 4300

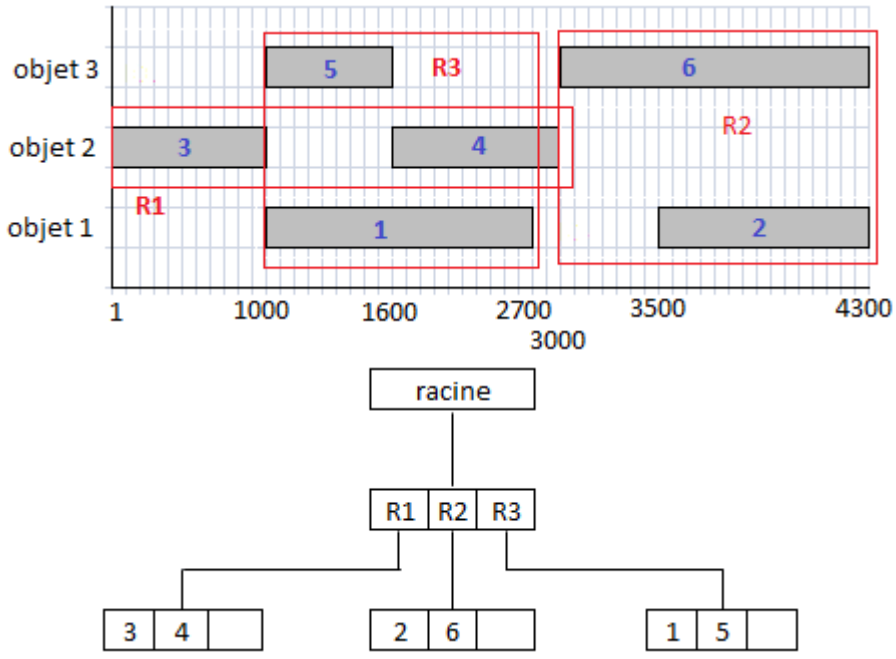
A partir de cette situation, on pourra définir les segments par dichotomie : la racine correspond au segment représentant toute la vidéo (1 - 4300) mais en divisant par 2 successivement on va tomber sur des nombres impairs (que l'on ne pourra pas diviser par 2). C'est pourquoi on ajuste le nombre d'images à la puissance de 2 la plus proche (soit ici 8192 = 2^{13}) en ajoutant des images fictives. La racine correspondra donc au segment (1-8192). Ses deux descendants correspondront aux segments (1 - 4096) et (4097 - 8192) et ainsi de suite.



Pour chaque nœud, la plage d'images est indiquée. Si un objet est présent dans une plage, un pointeur est placé vers l'objet correspondant sur le nœud dont la plage d'images le contient

(totalement ou partiellement). On a ainsi segmenté la vidéo en faisant correspondre à chaque segment les objets correspondants. On pourrait faire de même pour les activités.

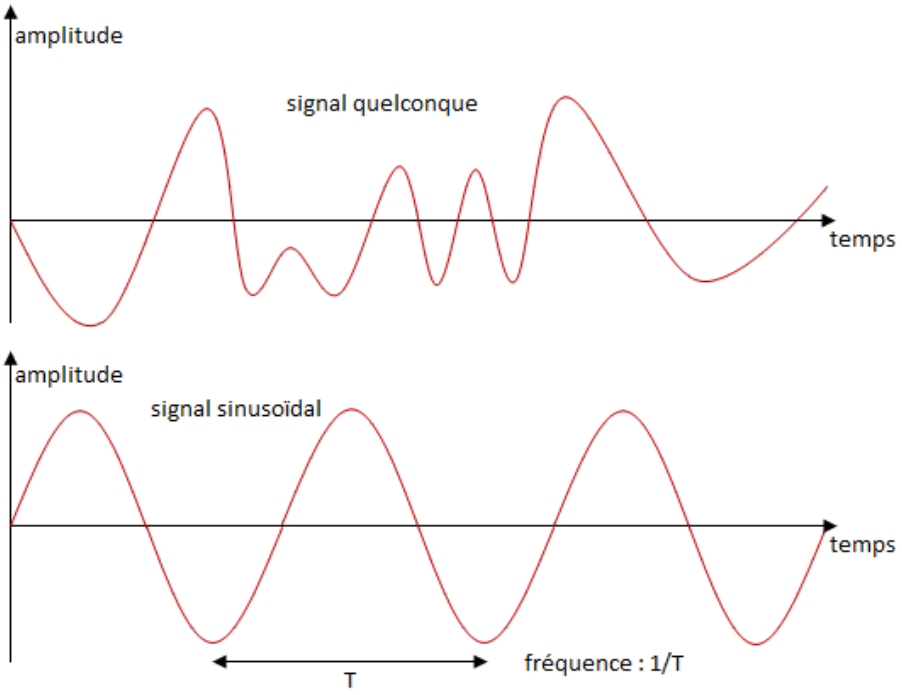
On peut aussi utiliser les arbres R pour répertorier les séquences où des objets interviennent. Les segments sont alors ces séquences, numérotées ci-dessous de 1 à 6. On peut définir des rectangles englobant certains segments, ici R1, R2, R3



Données audios

Le signal audio

Le son est représenté par une grandeur continue variant au cours du temps qui constitue le signal audio. Le théorème de Fourier a montré qu'un tel signal peut être considéré comme la superposition de signaux élémentaires sinusoïdaux se caractérisant chacun par une amplitude et une fréquence données.



Les schémas ci-dessus correspondent au son analogique. Pour passer au son numérique, il faut effectuer trois opérations :

- l'échantillonnage : on mesure l'amplitude du signal à une certaine fréquence f (fréquence d'échantillonnage), c'est à dire que toutes les 1/f secondes on mesure l'amplitude
- la quantification : on remplace l'échelle continue de l'amplitude par une échelle discrète allant de 0 à 2^n-1. On ajuste les mesures précédentes issues de l'échantillonnage pour aller au plus près des graduations de l'échelle discrète (en faisant cette opération on modifie - légèrement - le signal)
- le codage : chaque échantillon est converti dans la nouvelle échelle ; c'est donc une valeur numérique entière allant de 0 à 2^n-1.

Plus f est grand, plus le son numérisé se rapprochera du son analogique. mais cela a un coût. Une fréquence f = 44 Khz correspond à la prise de 44000 échantillons par seconde, donc 44000 nombres pour seulement une seconde de son. Par ailleurs, plus n est grand plus la mesure quantifiée sera proche de la mesure analogique. les valeurs courantes sont n = 8 (valeurs de 0 à 255 exprimées sur 8 bits) ou n = 16 (valeurs de 0 à 65535 exprimées sur 16 bits).

En définitive, le son numérisé est une suite de valeurs, un peu comparable àl a suite d'images d'une vidéo

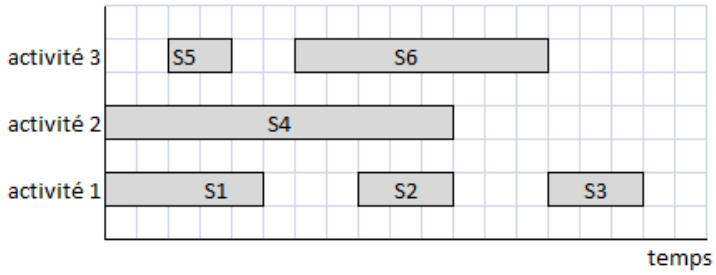
1	2	3	4	5	6	7	8	9	10	11	12	13
125	156	144	132	110	95	82	75	86	91	121	138	148

Segmentation

Comme pour la vidéo, on peut découper un signal en segments suivant plusieurs procédés :

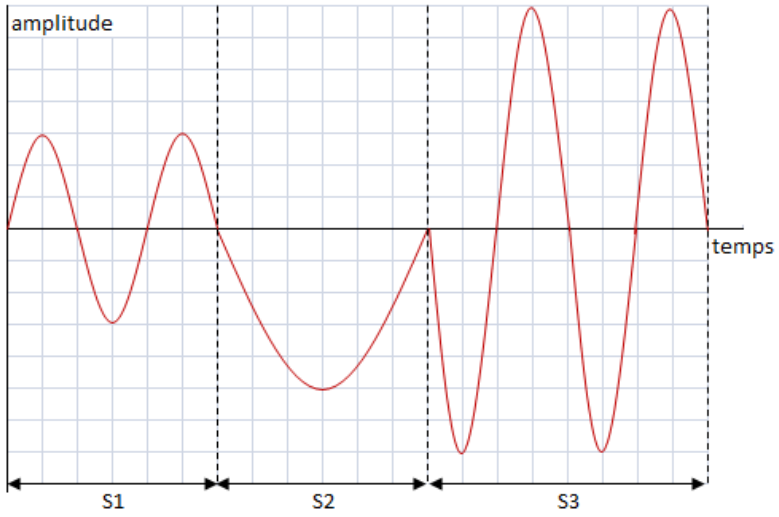
- le plus simple est un découpage régulier correspondant à un temps donné Δ. On obtient une suite de segments contigus deux à deux. Evidemment ce n'est pas le procédé le plus judicieux (c'est le prix de la simplicité)

- on peut se baser sur les activités. Par exemple tel chanteur chante tel air. On obtient alors un découpage comme celui indiqué ci-dessous :



On obtient dans ce cas des segments qui peuvent se superposer dans le temps.

- une troisième méthode est d'appliquer, comme pour la segmentation d'images une propriété d'homogénéité. Par exemple, on peut décomposer le signal en plages dans lesquelles l'amplitude et la fréquence sont relativement constantes :



Le processus de découpage peut être automatisé.

Evidemment, le nombre des segments peut être grand et on peut ici aussi appliquer une compression basée sur des transformations comme DFT et DCT vues plus haut.

Une fois la segmentation achevée, on peut utiliser des techniques d'arbres pour indexer chaque segment.

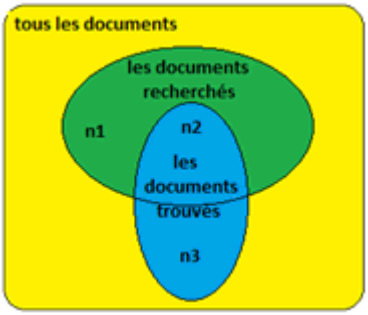
Ch3.5 : Données textuelles

Introduction

On peut considérer un document texte comme une série de chaînes de caractères. En particulier une chaîne peut être un critère qui servira aux recherches : recherche de l'occurrence de cette chaîne dans plusieurs documents. Toutefois, deux difficultés se présentent :

- une chaîne peut avoir des synonymes. En effet une même "chose" peut être désignée par des mots ou des groupes de mots différents
- inversement un mot ou un groupe de mots peuvent avoir plusieurs sens et désigner des "choses" différentes. C'est ce qu'on appelle la polysémie.

Ces deux problèmes affectent évidemment la performance d'un système de recherches. Une recherche de documents peut correspondre au schéma ci-dessous :



Il existe cependant des méthodes élaborées pour pallier ces difficultés. L'une de ces méthodes s'appelle LSI (Latent Semantic Indexing). Deux notions essentielles sont rattachées à cette méthode : la précision (precision) et le rappel (recall).

Soit n1 le nombre de documents recherchés mais non trouvés, n2 le nombre de documents recherchés et trouvés, n3 le nombre de documents trouvés mais non recherchés. Pour une recherche donnée, la précision P et le rappel R sont définis par

$$P = \frac{1+n2}{1+n2+n3} \qquad R = \frac{1+n2}{1+n1+n2}$$

On ajoute 1 au numérateur et au dénominateur pour éviter une division par zéro. Ces deux paramètres sont nécessaires car

- un algorithme de recherche qui ne trouverait rien (n2 = n3 = 0) aurait une précision de 100% et un rappel faible mais ne serait pas très utile.
- un algorithme de recherche qui trouverait tous les documents recherchés (n1 = 0) aurait un rappel de 100% et une précision faible mais ne serait pas très utile non plus.

Bien entendu, dans un texte, tous les mots ne sont pas porteurs d'une égale importance. les mots comme le, la, les, du, un, une, des, et, avec, comme, car, puis, ensuite, ... sont en quelque sorte inutiles dans une recherche (pour les recherches courantes évidemment). De même le mot "animal" dans un texte relatif à la zoologie n'est pas pertinent et le mot "ordinateur" dans un texte relatif à

l'informatique n'est pas pertinent non plus. On peut placer les mots non pertinents dans une liste spéciale appelée stop list.

Par ailleurs, on peut retrouver une partie d'un mot dans différents mots ; par exemple les mots voyage, voyages, voyageur, voyageuse, voyageurs, voyagiste, voyagistes ont une même racine "voyag" et peuvent être tous remplacés par cette racine.

Donc ayant éliminé les mots de la stop list et remplacé des mots par leur racine, on peut établir une statistique des mots rencontrés dans plusieurs documents ce qui conduit à un tableau :

mots/documents	D1	D2	D3	D4	D5	D6
t1	123	56	142	3	48	4
t2	43	4	37	67	2	208
t3	78	127	82	178	132	48
t4	206	59	199	63	61	5
t5	18	82	16	22	78	145

où D1, D2, ... sont des documents texte et t1, t2, ... sont des mots.

Recherche de similarités

Cependant les documents n'ont pas la même longueur. Pour prendre en considération cet important aspect, il faut normaliser les colonnes. On peut le faire avec la relation

$$F(i,j) = \frac{N(i,j)}{\sqrt{\sum_{k=1}^{m} N(k,j)^2}}$$

où N(i,j) est le nombre de mots ti dans le document Dj.

ce qui donnerait le résultat suivant :

mots/documents	D1	D2	D3	D4	D5	D6
t1	123	56	142	3	48	4
t2	43	4	37	67	2	208
t3	78	127	82	178	132	48
t4	206	59	199	63	61	5
t5	18	82	16	22	78	145
nb mots	1000	750	1100	750	800	900
c = somme des carrés	65822	29486	68114	40635	29537	66634
racine de c	256,6	171,7	261	201,6	171,9	258,1

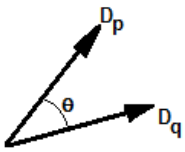
mots/documents	D1	D2	D3	D4	D5	D6
t1	0,479	0,326	0,544	0,015	0,279	0,015
t2	0,168	0,023	0,142	0,332	0,012	0,806
t3	0,304	0,740	0,314	0,883	0,768	0,186
t4	0,803	0,344	0,762	0,313	0,355	0,019
t5	0,070	0,478	0,061	0,109	0,454	0,562

On peut alors remplacer un document par la colonne correspondante considérée comme un vecteur Dj et un mot par la ligne correspondante. On désignera par la suite par f(i,j) le fréquence du mot ti dans la document Dj obtenue avec le procédé précédent.

La similarité entre deux documents est définie par l'expression Sim(d_p, d_q) :

$$Sim(D_p, D_q) = \mathbf{D}_p . \mathbf{D}_q{}^T = \sum_{i=1}^{m} f(i,p) f(i,q)$$

On peut considérer l'expression de la similarité comme le produit scalaire de deux vecteurs (représentant les documents) de longueur 1. Elle mesure donc le cosinus de l'angle entre les deux vecteurs.



Sim (D_p, D_q) = cosθ

Le tableau suivant donne les similarités des documents de notre exemple :

	D1	D2	D3	D4	D5	D6
D1	1,000					
D2	0,694	1,000				
D3	0,997	0,704	1,000			
D4	0,590	0,825	0,578	1,000		
D5	0,686	0,998	0,693	0,847	1,000	
D6	0,254	0,436	0,230	0,500	0,418	1,000

On peut ainsi constater que les documents D1 et D3 sont "proches", de même que les documents D2 et D5.

Supposons que l'on effectue une recherche Q de l'occurrence de plusieurs mots dans l'ensemble des documents. On obtiendra une colonne supplémentaire **q** au tableau précédent et on recherchera la proximité de chaque document avec cette nouvelle colonne. Un exemple montre comment le vecteur **q** peut être obtenu. Supposons que l'on souhaite trouver les documents pour lesquels figurent les mots t1 et t4. On posera alors a priori

q =	1		normalisation		q =	0,707
	0					0
	0					0
	1					0,707
	0					0

q représente un document fictif (pseudo document) . La recherche définie par Q consiste alors à chercher la similarité entre **q** et chaque document :

$q^T F$ =	0,707	0,000	0,000	0,707	0,000	x	0,479	0,326	0,544	0,015	0,279	0,015	=	D1	D2	D3	D4	D5	D6
							0,168	0,023	0,142	0,332	0,012	0,806		0,907	0,474	0,924	0,232	0,448	0,025
							0,304	0,740	0,314	0,883	0,768	0,186							
							0,803	0,344	0,762	0,313	0,355	0,019							
							0,070	0,478	0,061	0,109	0,454	0,562							

On constate que les documents D1 et D3 sont ceux qui correspondent le mieux à la requête Q.

Méthode LSI

Notre exemple ne correspond pas à la réalité dans laquelle le nombre de mots et le nombre de documents sont très grands. La matrice **F** d'éléments f(i,j) est une très grande matrice et sa manipulation nécessite de nombreuses opérations, de la place mémoire et un important temps de traitement. Pour réduire la dimension de la matrice **F** et donc le temps de traitement et le volume de stockage, des méthodes sont employées. La plus connue est la méthode LSI (Latent Semantic Indexing).

La méthode LSI s'appuie sur une technique mathématique appelée SVD (Singular Value Decomposition) qui consiste à remplacer en premier lieu la matrice **F** d'éléments f(i,j) par le produit de trois matrices

$$\mathbf{U} \times \mathbf{S} \times \mathbf{V}^T$$

où **S** est une matrice carrée diagonale dont les éléments de la diagonale principale sont décroissants du haut vers le bas et dans un second temps à réduire la matrice **S** (et donc les matrices **U**, **V** et par voie de conséquence **F**) en éliminant les éléments les moins significatifs.

Décomposition

Un résultat mathématique montre que la matrice **F** de dimension mxn (m lignes, n colonnes) peut s'écrire

$$\mathbf{F} = \mathbf{U} \times \mathbf{S} \times \mathbf{V}^T$$

où

- **U** est une matrice de dimension mxm telle que $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ où $\mathbf{U}^T$ est la transposée de **U** et **I** est la matrice identité (qui contient des 1 sur la diagonale principale et 0 ailleurs)
- **V** est une matrice de dimension nxn telle que $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ où $\mathbf{V}^T$ est la transposée de **V** et **I** est la matrice identité
- **S** est une matrice rectangulaire diagonale mxn dont les éléments de la diagonale principale sont décroissants du haut vers le bas

Les logiciels de mathématiques proposent le calcul de cette décomposition. Par exemple Scilab (http://www.scilab.org/fr) fournit les résultats suivants pour le tableau précédent :

```
Console Scilab
 F  =

    0.4794238    0.326122     0.5440893    0.0148823    0.2792917    0.0154957
    0.1676034    0.0232944    0.1417697    0.3323722    0.0116372    0.8057780
    0.3040248    0.7395981    0.3141924    0.8830186    0.7680522    0.1859488
    0.8029374    0.3435928    0.7624913    0.3125291    0.3549332    0.0193697
    0.0701596    0.4775358    0.0613058    0.1091371    0.4538490    0.5617202

-->[U,S,V]=svd(F)
 V  =

  - 0.4156382    0.4312229  - 0.3762994  - 0.0907231    0.6434029    0.2783745
  - 0.4573471  - 0.0870789    0.3364317    0.4395307  - 0.2558578    0.6414158
  - 0.4148925    0.4504709  - 0.3449015  - 0.0203365  - 0.6458275  - 0.2974499
  - 0.4207297  - 0.2539466    0.2985810  - 0.8147572  - 0.0604972    0.0431032
  - 0.4565553  - 0.0880399    0.3810621    0.3451172    0.3158977  - 0.6478431
  - 0.2459400  - 0.7289034  - 0.6260163    0.1234238  - 0.0050008  - 0.0325348
 S  =

    2.0640288    0.           0.           0.           0.           0.
    0.           0.9920309    0.           0.           0.           0.
    0.           0.           0.7430795    0.           0.           0.
    0.           0.           0.           0.4479153    0.           0.
    0.           0.           0.           0.           0.0535224    0.
 U  =

  - 0.3448309    0.3868564  - 0.2115195    0.3906015  - 0.7308377
  - 0.2337468  - 0.5429816  - 0.6794486  - 0.3911103  - 0.1895146
  - 0.6602990  - 0.2209244    0.6270873  - 0.3132798  - 0.1543206
  - 0.5356152    0.5393712  - 0.3136846  - 0.1497668    0.5489690
  - 0.3218317  - 0.4645258  - 0.0444129    0.7575545    0.3236955
```

On a bien

$\mathbf{U.S.V^T}$ =

-0,345	0,387	-0,212	0,391	-0,731
-0,234	-0,543	-0,679	-0,391	-0,19
-0,66	-0,221	0,627	-0,313	-0,154
-0,536	0,539	-0,314	-0,15	0,549
-0,322	-0,465	-0,044	0,758	0,324

x

2,064	0,000	0,000	0,000	0,000	0,000
0,000	0,992	0,000	0,000	0,000	0,000
0,000	0,000	0,743	0,000	0,000	0,000
0,000	0,000	0,000	0,448	0,000	0,000
0,000	0,000	0,000	0,000	0,054	0,000

x

-0,416	-0,457	-0,415	-0,421	-0,457	-0,246
0,431	-0,087	0,450	-0,254	-0,088	-0,729
-0,376	0,336	-0,345	0,299	0,381	-0,626
-0,091	0,440	-0,020	-0,815	0,345	0,123
0,643	-0,256	-0,646	-0,060	0,316	-0,005
0,278	0,641	-0,297	0,043	-0,648	-0,033

=

0,479	0,326	0,544	0,015	0,279	0,015
0,168	0,023	0,142	0,332	0,012	0,806
0,304	0,740	0,314	0,883	0,768	0,186
0,803	0,344	0,762	0,313	0,355	0,019
0,070	0,478	0,061	0,109	0,454	0,562

Réduction

On remplace maintenant la matrice **S** par une matrice **S*** de dimension moindre en ne gardant que les éléments diagonaux non négligeables. Par exemple, sur l'expression ci-dessus, on négligera dans **S** le dernier élément très négligeable par rapport aux autres, ce qui implique aussi une réduction des matrices **U** et **V** (pour que la multiplication puisse se faire) qui deviennent U* et V* :

$U^*S^*V^{*T}$ =

-0,345	0,387	-0,212	0,391
-0,234	-0,543	-0,679	-0,391
-0,66	-0,221	0,627	-0,313
-0,536	0,539	-0,314	-0,15
-0,322	-0,465	-0,044	0,758

x

2,064	0,000	0,000	0,000
0,000	0,992	0,000	0,000
0,000	0,000	0,743	0,000
0,000	0,000	0,000	0,448

x

-0,416	-0,457	-0,415	-0,421	-0,457	-0,246
0,431	-0,087	0,450	-0,254	-0,088	-0,729
-0,376	0,336	-0,345	0,299	0,381	-0,626
-0,091	0,440	-0,020	-0,815	0,345	0,123

=

0,505	0,316	0,519	0,013	0,292	0,015
0,174	0,021	0,135	0,332	0,015	0,806
0,309	0,737	0,309	0,883	0,771	0,186
0,784	0,351	0,781	0,314	0,346	0,020
0,059	0,482	0,072	0,110	0,448	0,562

Rappelons que si la matrice F initiale (que l'on doit retrouver ici) est de dimensions qxm (q lignes, m colonnes), si la matrice s* est de dimensions p x n , la matrice U* doit être de dimensions q xp et la matrice V* de dimensions m x n (donc V^{*T} de dimensions n xm).

Le résultat obtenu est une approximation de **F**. Nous avons cependant moins de calculs à faire dans la multiplication matricielle. Pour étudier la similarité de documents ou les documents qui se rapprochent le ,plus d'une requête, la réduction permet un gain de temps calcul appréciable.

L'interprétation de cette décomposition de la matrice F (matrice mot-document) est donnée dans les applications suivantes :

application 1 : comparaison de deux documents

Pour obtenir la ,similarité entre deux documents on effectue les produits Sim(**D**p, **D**q) ce qui conduit au produit matriciel $\mathbf{F}^T\mathbf{F} = (\mathbf{USV}^T)^T(\mathbf{USV}^T) = \mathbf{VS}^T\mathbf{U}^T\mathbf{USV}^T = (\mathbf{VS}^T)I(\mathbf{SV}^T) = (\mathbf{VS}^T)(\mathbf{VS}^T)^T$. En version réduite l'expression correspondante est $\mathbf{F^*}^T\mathbf{F^*} = (\mathbf{V^*S^*}^T)(\mathbf{V^*S^*}^T)^T = (\mathbf{V^*S^*})(\mathbf{V^*S^*})^T$ ce qui signifie que **F*** (approximation de **F**) est représentée par $(\mathbf{V^*S^*})^T$. Les colonnes de cette matrice **D** = $(\mathbf{V^*S^*})^T$ sont donc représentatives des vecteurs documents (mais comportent moins d'éléments). Appliquons à notre exemple :

V*S* =

-0,858	0,428	-0,280	-0,041
-0,944	-0,086	0,250	0,197
-0,856	0,447	-0,256	-0,009
-0,868	-0,252	0,222	-0,365
-0,942	-0,087	0,283	0,155
-0,508	-0,723	-0,465	0,055

$(V^*S^*)^T$ =

-0,858	-0,944	-0,856	-0,868	-0,942	-0,508
0,428	-0,086	0,447	-0,252	-0,087	-0,723
-0,280	0,250	-0,256	0,222	0,283	-0,465
-0,041	0,197	-0,009	-0,365	0,155	0,055

(V*S*)(V*S*)T =

	D1	D2	D3	D4	D5	D6
D1	0,999	0,695	0,998	0,590	0,686	0,254
D2	0,695	1,000	0,704	0,825	0,998	0,436
D3	0,998	0,704	0,999	0,578	0,694	0,230
D4	0,590	0,825	0,578	1,000	0,847	0,500
D5	0,686	0,998	0,694	0,847	1,000	0,418
D6	0,254	0,436	0,230	0,500	0,418	1,000

On constate que les résultats sont très proches de ceux obtenus par le calcul global mais ici les calculs sont moins longs. On note toujours la forte similarité entre D1 et D3 et entre D2 et D5.

application 2 : recherche de documents répondant à une requête Q

Supposons que l'on recherche les documents qui se distinguent par la présence des mots t1 et t4. On peut alors construire un pseudo-document représenté par le vecteur **q** que l'on normalise à 1, Comme dans le cas général, on va chercher la similarité avec les vrais documents D1 à D6.

On est alors ramené au cas précédent en cherchant la similarité de **q** avec les documents D1, D2, etc. On va donc être amené à ajouter à la matrice **V*S*** la ligne **QS*** telle que $\mathbf{q}^T = \mathbf{U^*S^*Q^{*T}}$ (par analogie à $\mathbf{F^*=U^*S^*V^{*T}}$). On en déduit que $\mathbf{q} = \mathbf{QS^{*T}U^{*T}} = \mathbf{QS^*U^{*T}}$ ou $\mathbf{QS^*} = \mathbf{qU^*}$. Cette dernière expression permet de calculer **QS*** :

$QS^* = q^TU^* =$

0,707	0,000	0,000	0,707	0,000

x

-0,345	0,387	-0,212	0,391
-0,234	-0,543	-0,679	-0,391
-0,660	-0,221	0,627	-0,313
-0,536	0,539	-0,314	-0,150
-0,322	-0,465	-0,044	0,758

=

-0,623	0,655	-0,371	0,170

La nouvelle matrice V*S* augmentée de la ligne QS*, sa transposée et le produit des deux, sont alors

$V^*S^* =$

-0,858	0,428	-0,280	-0,041
-0,944	-0,086	0,250	0,197
-0,856	0,447	-0,256	-0,009
-0,868	-0,252	0,222	-0,365
-0,942	-0,087	0,283	0,155
-0,508	-0,723	-0,465	0,055
-0,623	0,655	-0,371	0,170

$(V^*S^*)^T =$

-0,858	-0,944	-0,856	-0,868	-0,942	-0,508	-0,623
0,428	-0,086	0,447	-0,252	-0,087	-0,723	0,655
-0,280	0,250	-0,256	0,222	0,283	-0,465	-0,371
-0,041	0,197	-0,009	-0,365	0,155	0,055	0,170

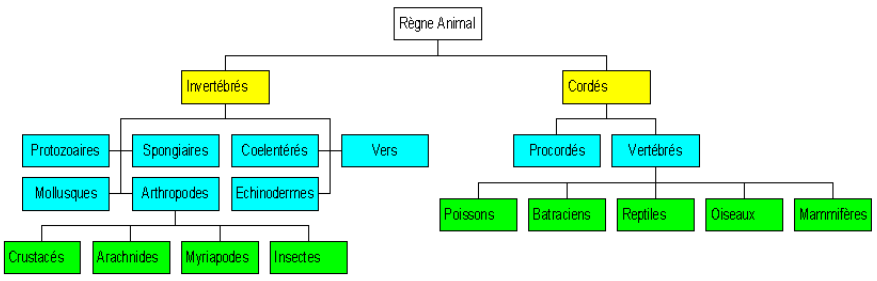
$(V^*S^*)^T(V^*S^*) =$

	D1	D2	D3	D4	D5	D6	Q
D1	0,999	0,695	0,998	0,590	0,686	0,254	0,911
D2	0,695	1,000	0,704	0,825	0,998	0,436	0,472
D3	0,998	0,704	0,999	0,578	0,694	0,230	0,919
D4	0,590	0,825	0,578	1,000	0,847	0,500	0,231
D5	0,686	0,998	0,694	0,847	1,000	0,418	0,451
D6	0,254	0,436	0,230	0,500	0,418	1,000	0,025
Q	0,911	0,472	0,919	0,231	0,451	0,025	0,983

La dernière matrice est celle des similarités. On retrouve un résultat précédent à savoir que les deux documents répondant au mieux à la requête sont D1 et D3.

Indexation par arbres TV

Ce type d'indexation est celui utilisé dans la classification comme par exemple la classification des animaux représentée ci-dessous



source : rvinchon.pagesperso-orange.fr

Pour chercher un animal, on part de la racine puis on fait jouer un critère, puis un autre, etc..

Les arbres TV (telescopic vector) s'appuient sur cette idée en faisant jouer des dimensions que l'on appelle dimensions actives. Par exemple, si on considère un document représenté par un vecteur (0,24 0,14 0,96) et si on recherche des documents voisins, on pourra rechercher d'abord les documents dont la composante 3 (3ème dimension) est proche de 0,96, puis on pourra raffiner la recherche en se focalisant sur les dimensions 1 et 2.

Formellement, un arbre TV se caractérise comme les arbres R par des nœuds représentant des régions, la région d'un nœud père englobant les régions des nœuds fils. Il y a deux paramètres à spécifier pour un arbre TV : le nombre maximum nmax d'enfants que peut avoir un nœud et le nombre maximal α de dimensions actives (avec $0 < \alpha \leq k$) où k est le nombre de dimensions de l'espace de recherche. On note TV(k, nmax, α).

On ajoute les règles suivantes :

- Toutes les données sont dans les feuilles de l'arbre
- Chaque nœud (sauf la racine et les feuilles) doit être au moins à moitié plein.

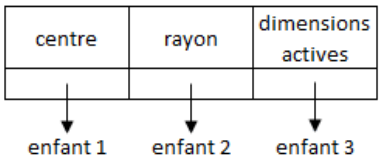
Un document , comme on l'a vu précédemment, peut être représenté par un vecteur (ou un point) dans un espace de dimension k (k étant le nombre de mots pris en considération pour comparer des documents). Pour évaluer la proximité de deux documents, on utilise une distance, comme, par exemple la distance euclidienne. Par exemple la distance entre les documents D1 (0,24 0,14 0,96) et D2 (0 0 1) est

$$d = \sqrt{(0{,}24-0)^2 + (0{,}14-0)^2 + (0{,}96-1)^2} = 0{,}28$$

Mais si l'on veut comparer les deux documents seulement sur leur troisième coordonnée (dimension active : 3) , on aura la dimension active

$$d_3 = \sqrt{(0{,}96-1)^2} = 0{,}04$$

Considérons une représentation par un arbre TV(5,3,2). Les documents sont caractérisés par rapport à 5 mots et correspondent chacun à un point possédant 5 coordonnées. Un nœud ne pourra avoir au maximum 3 enfants et le nombre maximum de dimensions actives est 2 (sur les 5 possibles). Un nœud aura la physionomie suivante :



Un tel nœud détermine une région centrée sur le centre indiqué et de rayon indiqué. Tout point de cette région doit vérifier, si les dimensions actives sont par exemple 2 et 3, la condition $d_{2,3}(C, P) \leq$ rayon.

Sans vouloir trop s'attarder sur la théorie des arbres TV, prenons un exemple illustratif uniquement pour montrer l'esprit de la démarche. On utilisera un arbre TV(5,3,2). Supposons que l'on ait 6 documents D1, D2, D3, D4, D5, D6, définis par leurs coordonnées :

D1	D2	D3	D4	D5	D6
0,23	0,00	0,00	0,61	0,51	0,00
0,14	0,00	0,00	0,67	0,14	0,00
0,93	0,39	0,43	0,13	0,77	0,99
0,05	0,92	0,89	0,00	0,26	0,00
0,23	0,09	0,14	0,40	0,26	0,10

Plaçons d'abord la racine (qui représente tout l'espace) mais que l'on indexe sur les dimensions 2 et 3 (dimensions actives). On peut sans problème loger les documents D1, D2, D3 sous cette racine :

C	R	2,3

D1	D2	D3
0,23	0	0
0,14	0	0
0,93	0,39	0,43
0,05	0,92	0,89
0,23	0,09	0,14

Pour placer D4, on est embêté car il n'y a plus de place car le nombre maximum d'enfants par nœud est 3. Il faut donc effectuer une décomposition en deux nœuds enfants dans lesquels il faudra placer de manière équilibrée les documents D1, D2, D3, D4 (2 par nœud). On choisit les couples en fonction de la distance active pour les dimensions 2 et 3. Le tableau suivant donne cette distance pour tous les documents.

2,3	D1	D2	D3	D4	D5	D6
D1		0,56	0,52	0,96	0,17	0,15
D2			0,04	0,72	0,40	0,60
D3				0,74	0,36	0,56
D4					0,83	1,09
D5						0,03
D6						

On constate que les documents les plus proches (par rapport à la distance active) sont D2 et D3. On aura donc l'arborescence suivante :

C	R	2,3

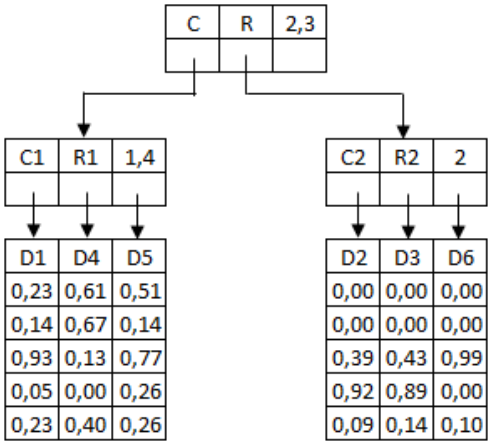
C1	R1	1,4

D1	D4
0,23	0,61
0,14	0,67
0,93	0,13
0,05	0,00
0,23	0,40

C2	R2	2

D2	D3
0,00	0,00
0,00	0,00
0,39	0,43
0,92	0,89
0,09	0,14

On ne précisera pas ici les centres et les rayons des nœuds pour ne pas compliquer les choses. Nous avons toutefois opté pour une décomposition future pour les dimensions actives 1,4 pour le nœud N1 et 2 pour le nœud N2. Reste à placer D5 et D6. Comme il reste deux places, on n'a pas à décompose pour le moment. Mais où mettre D5 (et donc D6) ? On pourra choisir de mettre D6 avec D2 et D3 (ils sont tous alignés sur l'axe 3) et donc D5 avec D1 et D4.



Bibliographie

En bibliothèque

- V.S. Subrahmanian, Principles of Multimedia Database Systems, Ed. Morgan Kaufmann
- Lynne Dunckley, Multimedia Databases, an object-relational approach, Pearson Education
- Massih-Reza Amini, Eric Gaussier, Recherche d'information, applications, modèles et algorithmes, Eyrolles

---

Sur le Web

- http://psoug.org/reference/dbms_lob.html : fonctions de manipulation des LOB
- http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/apex/r31/apex31nf/apex31blob.htm : documentation Oracle pour Application Express
- http://helyos.developpez.com/lob/ : un tutorial de Helyos (un peu ancien)
