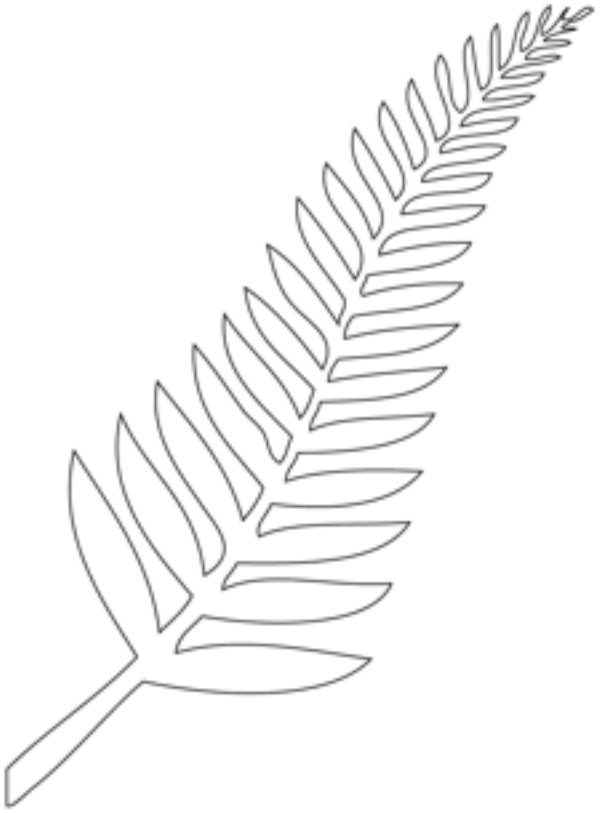
BASE DE DONNÉES

Tutoriel

Jean-Claude SOHM, "Les bases de données relationnelles", 2002, Cerig/Grenoble INP-Pagora

SOMMAIRE



Enseignant: Antoine Melo

1

GÉRER L'INFORMATION

Les activités humaines génèrent des données. Il en a toujours été ainsi et, plus notre civilisation se développe, plus le volume de ces données croît. Hors, les données sont souvent gérées par des moyens informatiques. Le mot "informatique" lui-même résulte de la contraction de "information" et "automatique" - l'informatique est donc la technique qui permet le traitement automatique de l'information.

99 % du temps, on range ses informations dans une base de données. Pour le reste, on peut les enregistrer dans un fichier comme on a appris à le faire… mais quand on a goûté aux bases de données, on peut difficilement s'en passer ensuite !

Dans les entreprises, on manipule souvent des données ayant la même structure. Prenons l'exemple de la liste des membres du personnel : pour chaque personne, on enregistre le nom, le prénom, le sexe, la date de naissance, l'adresse, la fonction dans l'entreprise, etc. Toutes ces données ont la même structure ; si elles sont gérées par des moyens informatiques, on dit qu'elles constituent une base de données. On utilise aussi le sigle BD et le terme anglais correspondant est DataBase DB.

Définition : une base de données est un ensemble structuré, aussi exhaustif que possible, de données organisées (pas de doublons).

Reprenons l'exemple de la base de données du personnel. Elle est utilisée pour la paye mensuelle, pour l'avancement, les mises à la retraite, etc. L'exhaustivité est indispensable pour le personnel, car la personne qui est absente de la base... n'est pas

payée. L'unicité est importante pour l'employeur, car la personne qui est enregistré deux fois... risque de toucher double paye !

Les bases de données sont très utilisées dans les entreprises. Outre la liste des membres du personnel, on y trouve tout ce qui concerne les fournisseurs, les clients, les prospects, les contacts, les commandes, les factures, les produits et services, le stock, le commerce électronique, etc.

Auparavant, pour stocker l'information, on utilisait des fiches, regroupées dans des boites appelées fichiers. Elles étaient triées manuellement. Le développement des bases de données gérées par des moyens informatiques a rendu souvent obsolètes ces anciennes techniques.

Le stockage des données (les tables)

Des données ayant même structure peuvent être rangées dans un même tableau. Dans le cas de la liste des membres du personnel, la première colonne contiendra les noms, la seconde les prénoms, la troisième le sexe, la quatrième la date de naissance, etc. La caractéristique d'un tel tableau est que toutes les données d'une même colonne sont du même type. Dans une base de données, un tel tableau s'appelle une table. Ci-dessous se trouve un exemple simple de table :

Nom	Prénom	Genre	Adresse	Ville	CP
Durand	Pierre	M	31 rue des champs	Uriage	38410
Chose	Stéphanie	F	2 place Stanislas	Nancy	54000
Trombe	Jean	M	18 cours de la libération	Grenoble	38001

Dans une table, les termes ligne, enregistrement ou occurence sont synonymes. Il en est de même pour les termes colonnes et champs.

La table d'une base de données ne doit pas être confondue avec la feuille de calcul d'un tableur. Cette dernière est également constituée d'un tableau, mais toutes les données d'une même colonne ne sont pas forcément du même type. Dans le cas où elles le sont, la feuille de données peut facilement être transformée en table. En re-

vanche, l'exportation d'une table de SGBD vers un tableur est théoriquement toujours possible. En pratique il faut, dans les deux cas, disposer du filtre qui permet à l'un des logiciels de lire le format de l'autre. A défaut, on peut exporter en mode texte (avec délimiteur) dans un logiciel, puis réimporter dans l'autre.

Le logiciel (SGBD)
Le logiciel qui gère une base de données s'appelle un système de gestion de base de données. On le désigne généralement pas son sigle SGBD (DBMS en anglais, pour Data Base Management System). En fait, il devrait s'appeler "logiciel de gestion de base de données" car, en informatique, le mot "système" désigne généralement l'ensemble matériel + logiciel. Mais l'expression SGBD est consacrée par l'usage et n'avons pas vraiment le choix…

Tous les SGBD présentent à peu près les mêmes fonctionnalités. Ils se distinguent par leur coût, par le volume de données qu'ils sont capables de gérer, par le nombre d'utilisateurs qui peuvent interroger la base simultanément, par la facilité avec laquelle ils s'interfacent avec les autres logiciels d'application utilisés par l'entreprise.

Il existe des bases de données de toutes tailles, depuis les plus modestes (une liste des numéros de téléphone utilisée par une seule personne), jusqu'aux plus grandes (la base des données commerciales d'un magasin à succursales multiples, contenant des téraoctets de données ou plus).

Le nombre d'utilisateurs utilisant une base de données est également extrêmement variable. Une BD peut servir à une seule personne, laquelle l'utilise sur son poste de travail, ou être à la disposition de dizaines de milliers d'agents (comme dans les systèmes de réservation des billets d'avion par exemple).

Les éditeurs de SGBD se partagent un marché mondial en progression d'environ 10% depuis des années : de 8 milliards dans les années 2000, .à plus de 20 milliard de dollars de nos jours Les principaux éditeurs sont : Oracle, MySQL et Microsoft SQL Server.

Le classement par nombre d'exemplaires (ou licences) vendus est très différent. Il met en avant les SGBD conçus pour gérer les bases de taille modeste ou modérée. Dans ce domaine l'éditeur Microsoft, qui vend plusieurs millions d'exemplaires de son logiciel Access par mois, pulvérise tous les records. L'usage des SGBD se démocratise à toute vitesse, bien qu'un SGBD soit plus difficile à maîtriser qu'un traitement de texte ou un tableur (pour ne citer que les logiciels les plus courants). L'image du SGBD servant uniquement les très grosses bases, propriété d'une grande multinationale, fonctionnant sous Unix sur une machine monstrueuse, géré par un administrateur dictatorial, et coûtant un prix fou a vécu!

Un SGBD est principalement constitué d'un moteur et d'une interface graphique. Le moteur est le coeur du logiciel, c'est à dire qu'il assure les fonctions essentielles : saisir les données, les stocker, les manipuler, etc. L'interface graphique permet à l'utilisateur de communiquer avec le logiciel. Pour dialoguer avec les SGBD qui ne sont pas équipés d'une interface graphique, on utilise le langage SQL (Structured Query Language) pour lancer des instructions à l'aide d'un éditeur de lignes.

Le matériel (serveur de BD)

Le choix du matériel informatique sur lequel on installe un SGBD est fonction, comme ce dernier, du volume des données stockées dans la base et du nombre maximum d'utilisateurs simultanés.

Lorsque le nombre d'enregistrements par table n'excède pas le million, et que le nombre d'utilisateurs varie de une à quelques personnes, un micro-ordinateur actuel de bonnes performances, un logiciel système pour poste de travail, et un SGBD "bureautique" suffisent.

Si ces chiffres sont dépassés, ou si le temps de traitement des données devient inacceptable, il faut viser vers des logiciels plus performants. Le micro-ordinateur doit être remplacé par un serveur de BD, dont les accès aux disques durs sont nettement plus rapides. Le logiciel système client doit être remplacé par un logiciel système serveur (donc multi-utilisateurs). Ceci dit, la structure d'une grosse base n'est pas

différente de celle d'une petite, et il n'est pas nécessaire de disposer d'un "mainframe" (une grosse machine) gérant des milliers de milliards d'octets pour apprendre à se servir des BD. Ce n'est pas parce qu'il gère un plus grand volume de données qu'un SGBD possède plus de fonctionnalités.

Quelle que soit sa taille, le système constitué de la machine et du SGBD doit être correctement équilibré. Un serveur de BD doit posséder à la fois les qualités de serveur de fichier (bon accès aux disques) et celles d'un serveur d'applications (unité centrale bien dimensionnée, mémoire vive suffisante).

Jusqu'au début des années 2000, les constructeurs de serveurs (et les éditeurs de SGBD) conseillaient à leurs clients de consolider leurs données, en les rassemblant dans un nombre minimum de BD, installées sur un nombre minimum de serveurs surpuissants. Comme le coût des serveurs croît exponentiellement avec le nombre de processeurs, et que le coût des licences (des SGBD) est proportionnel au nombre de processeurs, constructeurs et éditeurs ont gagné de l'or pendant longtemps. Avec le resserrement des cordons de la bourse, les services informatiques des entreprises commencent à recourir - de gré ou de force - à des systèmes plus décentralisés; vous avez peut être déjà entendu parlé du cloud computing.

L'administration de la base de données

L'ensemble "serveur de BD + SGBD" constitue un système informatique dont l'importance ne cesse de croître dans l'entreprise. La personne responsable de la maintenance et de l'évolution de ce système s'appelle l'administrateur de bases de données. Dès que l'entreprise atteint la taille d'une grosse PME, l'administration de la BD peut nécessiter la présence d'une personne à temps plein, voire plus.

Être administrateur de BD requiert des compétences particulières, très différentes de celles requises pour être administrateur de réseau ou de système informatique. Il en résulte le développement de deux pôles de compétences informatiques dans l'entreprise.

Pour être complet, il faut signaler que le développement des sites web contribue à créer un troisième pôle de compétences dans l'entreprise. Le responsable correspondant est appelé webmestre, et non "administrateur de site Internet", parce que le poste requiert des compétences multidisciplinaires (et pas seulement informatiques).

Les différents modèles de bases de données

Les bases de données du modèle "relationnel" sont les plus répandues (depuis le milieu des années 80 environ), car elles conviennent bien à la majorité des besoins des entreprises. Le SGBD qui gère une BD relationnelle est appelé "SGBD relationnel", ce qui est souvent abrégé en SGBDR.

D'autres modèles de bases de données ont été proposés : hiérarchique, en réseau, orienté objet, relationnel objet. Aucun d'entre eux n'a pu détrôner le modèle relationnel, ni se faire une place notable sur le marché (sauf le relationnel objet, prôné par Oracle, qui connaît un certain développement).

Malgré sa généralité, le modèle relationnel ne convient pas à toutes les BD rencontrées en pratique. Il existe donc des SGBD spécialisés. Les deux exemples les plus connus concernent la gestion des BD bibliographiques (ou documentaires), et celle des BD géographiques gérées à l'aide d'un SIG (Système d'Information Géographique).

2

STOCKER L'INFORMATION

Les tables

Les index

Les listes de choix

Les tables

Nous avons vu au chapitre précédent que, dans les BD, les données sont stockées dans des tables. Nous aborderons donc l'étude des BD par celle de la création et de la manipulation des tables. Dans les bases de données, la table est le premier objet par ordre d'importance décroissante. Pas de table, pas de données !

La création d'une table avec MS Access

La première opération consiste à créer d'abord une base de données vide. Les logiciels réclament un nom de fichier et un seul, car toutes les informations relatives à la BD seront stockées dans le même fichier. Avec MS Access, ce dernier comporte l'extension ".mdb", et sa taille initiale est voisine de 96-100 ko).

La fenêtre relative à la base de données apparaît. Dans la colonne de gauche figurent les "objets" de la base de données. Un mot sur ces objets, qui sont utilisés :

- les tables, pour stocker les données ;
- les requêtes, pour retrouver les données ;
- les formulaires, pour saisir les données ou les visualiser à l'écran ;
- les états, pour imprimer les données ;
- les pages, pour communiquer avec la BD via un navigateur (Internet Explorer uniquement) ;
- les macros, pour automatiser des opérations répétitives effectuées sur la base ;
- les modules, pour rajouter des fonctionnalités grâce à de la programmation en VBA (Visual Basic for Applications).

Trois méthodes sont proposées pour créer une nouvelle table :

- créer une table en mode création. C'est la méthode générale ;
- créer une table à l'aide de l'assistant. Ce dernier offre un certain nombre de tables toutes prêtes dont vous pouvez vous inspirer. Cependant, rien ne remplace une bonne analyse du problème de stockage des données, suivie d'une réalisation personnalisée et adaptée ;

- créer une table en entrant des données. Une table toute prête vous est proposée, dans laquelle vous pouvez immédiatement saisir des données, le logiciel se chargeant de déterminer leur type et leur format. Cette façon de procéder est déplorable, et nous la déconseillons absolument, sauf pour des essais sans suite.

En mode création, une fenêtre s'ouvre qui permet de définir la table champ par champ, en précisant le nom du champ et le type de données qu'il contient.

Les types de données

Tous les SGBD offrent la possibilité de stocker du texte, de l'information numérique, et des dates (avec ou sans les heures). Le type "monétaire" est un cas particulier d'information numérique, et le lien hypertexte un cas particulier de texte. Lorsque l'on utilise MS Access, une liste déroulante propose les types de données suivants :

- texte (type par défaut)
- mémo (texte contenant plus de 255 caractères)
- numérique
- date/heure
- monétaire (cas particulier du numérique)
- numéroauto : numérotation automatique, séquentielle ou aléatoire
- oui/non, c'est à dire booléen (deux valeurs possibles seulement)
- objet OLE : pour le stockage des données numériques autres que le texte, les nombres les dates
- lien hypertexte : cas particulier du type texte

Le tableau ci-dessous précise les propriétés de ces différents types. Il est nécessaire, à ce stade, d'effectuer les bons choix. Si l'on modifie ultérieurement le type de données d'un champ, alors que la table contient déjà des informations, ces dernières risquent d'être modifiées ou perdues.

Type	Propriétés	Taille
Texte	Le champ peut contenir n'importe quel caractère alphanumérique (chiffre, lettre, signe de ponctuation). Ce type de données est utilisé pour le texte, mais aussi pour les nombres sur lesquels on n'effectue pas de calculs (code postal, numéro de téléphone)	< 256 caractères
Mémo	Le champ peut contenir n'importe quel caractère alphanumérique. Le type mémo est réservé aux champs de type texte susceptibles de contenir plus de 255 caractères	< 65.536 caractères
Numérique	Données numériques (non monétaires) susceptibles d'être utilisées dans des opérations mathématiques	1 à 16 octets
Date/heure	Données de date et/ou d'heure (pour les années comprises entre 100 et 9999)	8 octets
Monétaire	Données monétaires, présentées avec deux chiffres après la virgule, et le symbole monétaire du pays	8 octets
NuméroAuto	Numérotation automatique, séquentielle (commençant à 1) ou aléatoire. Souvent utilisée pour générer le code des enregistrements	4 octets (entier long)
Oui/non	Variable booléenne (deux valeurs possibles uniquement)	1 bit
Objet OLE	Pour lier un objet extérieur, ou incorporer un objet dans la base. Souvent utilisé pour les données multimédia. Peut servir pour tout fichier binaire (document Word, feuille de calcul Excel, etc.)	< 1 Go
Lien hypertexte	Permet d'enregistrer des URL de sites web et des adresses de courrier électronique	< 2049 caractères

Pour sauvegarder son travail, cliquer sur l'icône "Enregistrer" dans la barre d'outils. Lorsque tous les champs sont définis, fermer la fenêtre, en répondant "non" à la question relative à la clé primaire, et en donnant un nom à la table. Ce nom apparaît désormais dans la fenêtre relative à la base de données. Nous verrons ultérieurement à quoi set la clé primaire.

Pour modifier une table, il faut la sélectionner (dans la fenêtre base de données), puis cliquer sur l'icône "Modifier". La fenêtre s'ouvre en mode création comme précédemment.

Pour supprimer une table, il faut la sélectionner et utiliser la fonction "supprimer" (clic droit) ou la touche du même nom.

Les propriétés des champs

La partie inférieure de la fenêtre du mode création est intitulée "Propriétés du champ". Ces propriétés se trouvent rassemblées dans l'onglet "Général".

La liste des propriétés d'un champ dépend du type de données choisi, mais une propriété donnée peut apparaître pour des types de données différents. Exemple : la propriété "Taille du champ" apparaît pour les types de données "Texte", "Numérique"et "NuméroAuto".

Les principales propriétés sont :

- Taille du champ ;
- Format : définit la manière dont les informations s'affichent. Exemple : le format monétaire affiche deux chiffres après la virgule, puis un espace et le symbole de l'euro ;
- Masque de saisie : guide la saisie des données et exerce un contrôle. Exemple : un code postal français est composé de cinq chiffres ;
- Légende : définit le nom de l'étiquette dans le formulaire associé à la table. Il est préférable d'implémenter cette propriété au niveau du formulaire lui-même ;
- Valeur par défaut : valeur qui s'affiche dans le champ avant saisie par l'utilisateur ;
- Valide si : condition de validité du champ. Exemple : une notation sur 20 doit être comprise entre 0 et 20 ;
- Message si erreur : ce message s'affiche si la condition de validité précédente n'est pas satisfaite ;
- Null interdit : le champ correspondant ne peut rester vide lors de la saisie d'un enregistrement ;
- Chaîne vide autorisée : le champ peut contenir une chaîne ne comportant aucun caractère ;
- Indexé : un fichier index est associé au champ de telle sorte que les recherches d'information s'effectuent plus rapidement. Le chapitre 3 explique ce qu'est un index, et comment on le crée ;

• Compression unicode : un octet suffit pour saisir un caractère (pour les alphabets utilisés dans l'Europe de l'ouest et dans le monde anglophone).

Pour faire fonctionner correctement certaines requêtes, il est important de bien comprendre la différence entre la valeur Null, une chaîne vide et une chaîne blanche. Un champ d'un enregistrement :

• possède la valeur Null si aucune information n'a été introduite, ou si l'information présente a été supprimée ;
• contient une chaîne vide si on a défini la valeur par défaut du champ à l'aide de deux guillemets contigus (""), et si aucune information n'a été introduite (ou si l'information introduite a été supprimée) ;
• contient une chaîne "blanche", si un ou plusieurs espaces ont été introduits et n'ont pas été supprimés.

La définition de certaines propriétés des champs soulève des problèmes de syntaxe. La touche F1 fournit une aide contextuelle, c'est à dire liée à la position du curseur. On pourra également consulter les annexes suivantes :

Saisir les données
Pour introduire des données dans une table, il faut l'ouvrir en mode "feuille de données". Dans la fenêtre base de données (l'objet table étant sélectionné), on peut :

• faire un double clic sur le nom de la table ;
• sélectionner la table, et cliquer sur l'icône "Ouvrir" ;
• faire un clic droit sur la table et sélectionner "Ouvrir" dans la liste déroulante.

Sur le plan pratique, pour passer facilement du mode "création" au mode "feuille de données" ou vice versa, il suffit de cliquer sur l'icône "affichage" représentée ci-contre. Cette icône est présente dès que l'on se trouve déjà dans l'un des deux modes précités.

On peut ainsi vérifier le bon fonctionnement des listes, formats, masques de saisie, etc. On notera que le contrôle des informations se fait lors du passage à l'enregis-

trement suivant. Par exemple, si une liste est obligatoire, une information qui ne fait pas partie de la liste ne sera rejetée qu'au passage à la ligne suivante. Avec l'affichage d'un message qui, selon les bonnes traditions de l'informatique, risque fort d'être sibyllin...

Il est également essentiel de bien réaliser que, dans les BD, les table se présentent sous un double aspect. C'est ainsi qu'il faut distinguer :

- l'aspect structure : noms des champs, types de données, propriétés, listes -- en bref, tout ce qui est défini dans le mode "création" de la table ;
- l'aspect contenu : les valeurs introduites dans les champs des divers enregistrements, en mode "feuille de données".

Nous rencontrerons aussi ce double aspect à propos des requêtes.

Les index

Les bases de données prennent souvent des proportions importantes, voire considérables. Si une recherche d'information dans une table s'effectue de manière simplement séquentielle (c'est à dire en examinant toute la table, ou du moins tous les champs concernés, du début jusqu'à la fin), le temps d'attente peut devenir prohibitif pour l'opérateur. L'index est l'outil qui permet de résoudre ce problème.

La notion d'index est très ancienne. Elle semble remonter à la grande bibliothèque d'Alexandrie. Cette bibliothèque s'était dotée d'un index par auteurs et d'un index par matières pour faciliter les recherches de ses lecteurs parmi les nombreux ouvrages (en papyrus) qu'elle possédait.

Imaginons une bibliothèque dans laquelle les livres sont rangés n'importe comment -- au fur et à mesure de leur acquisition, par exemple. Pour rechercher un livre dont on connaît le titre, il faut parcourir rayons dans l'ordre (recherche séquentielle). Ou l'on finit par trouver (ouf !), ou l'on arrive bredouille au dernier rayonnage et on en conclut que la bibliothèque ne possède pas l'ouvrage.

Bien entendu, personne ne sera jamais assez sot pour organiser une bibliothèque de pareille façon. Les livres seront rangés par ordre alphabétique du titre, par exemple. Si le titre que nous recherchons commence par un L, nous irons vers le rayon du milieu et nous examinerons un ouvrage. Si son titre commence par un P, nous conclurons que nous sommes allés trop loin. Nous reculerons quelque peu, et nous réitérerons notre démarche . Nous arriverons beaucoup plus vite que précédemment à trouver le livre que nous recherchons, ou à conclure qu'il n'est pas dans la bibliothèque, parce que nous pratiquons une méthode dichotomique (optimisée), qui nous permet d'arriver au résultat en n'examinant qu'une petite fraction des livres contenus dans la bibliothèque. Nous pouvons pratiquer une technique efficace de recherche parce que les livres constituent un ensemble ordonné. Dans un ensemble désordonné, on ne peut pratiquer qu'une recherche séquentielle, beaucoup plus lente.

Mais... comment ferons-nous si nous recherchons un livre dont nous connaissons l'auteur, mais pas le titre ? Les livres de la bibliothèque ne peuvent pas être triés à la fois par ordre alphabétique de leur titre, et celui de leur auteur. C'est ici qu'intervient la notion d'index. Pour chaque livre, nous créons une fiche sur laquelle nous inscrivons le nom de l'auteur et le titre du livre. Puis nous rangeons ces fiches par ordre alphabétique des noms d'auteur. Pour rechercher le livre d'un auteur donné, nous compulsons les fiches. Comme elles constituent un ensemble ordonné, l'opération est rapide ; ou nous obtenons le titre du livre, ou nous concluons que le livre ne se trouve pas dans la bibliothèque. Dans le premier cas, nous nous rendons dans les rayons munis du titre, et comme les livres sont classés par ordre alphabétique de leur titre, notre trouvons rapidement l'ouvrage en question.

Imaginons maintenant que la bibliothèque soit gérée par ordinateur. Si la table qui contient les livres est triée par ordre alphabétique des titres, il faut que nous construisions un index informatique sur le champ auteur pour que la recherche d'un livre dont on connaît l'auteur s'effectue rapidement. Car l'ordinateur est programmé à l'image de ce que font les humains : dans un ensemble non trié il recherche séquentiellement, alors que dans un ensemble trié il recherche par dichotomie. Dans le second cas, il va beaucoup plus vite.

Le fonctionnement de l'index

Nous disposons maintenant d'un ordinateur pour gérer la bibliothèque. Au fur et à mesure que nous achetons des livres, nous les numérotons dans l'ordre. Puis nous saisissons dans la table d'une BD les données qui les caractérisent (numéro, titre, auteur, éditeur, année d'édition, ISBN, etc.), et nous les rangeons sur les rayons dans l'ordre de leur numéro. La table se présente ainsi :

N°	Titre	Auteur	Éditeur	Année	ISBN
1	Mon jardin	J. Machin	Eyrolles	1998	5-1234-4321-8
2	Access	A. Chose	Dunod	2002	3-6789-9876-2
3	Les écoles	S. Truc	Lattès	2006	4-1985-5891-3

En informatique, un index est représenté par une table à une seule colonne, comme on le voit sur la figure ci-dessous. Dans le premier index (index sur le titre), le premier titre par ordre alphabétique correspond au livre n° 2 (Access), suivi du livre n ° 3 (Les écoles) et du livre n° 1 (Mon jardin). Les autres index s'interprètent de la même façon.

2	2	2	1	2
3	1	1	3	3
1	3	3	2	1
Titre	Auteur	Editeur	Année	ISBN

L'index présente des avantages :

- il accélère les recherches d'information. En effet, l'index est une représentation de la table, triée sur un champ donné. On peut donc lui appliquer les méthodes connues de recherche rapide sur un ensemble ordonné (c'est le SGBD qui se charge de l'opération, laquelle est transparente pour l'opérateur) ;
- il est de taille très inférieure à celle de la table : on peut le remettre à jour en temps réel à chaque modification de cette dernière ;
- il peut servir à empêcher l'opérateur de créer des enregistrements dupliquées en saisissant deux fois, par erreur, les mêmes données. Nous reviendrons sur ce point au paragraphe suivant.

L'index ne possède pas que des avantages. Voici pour ses inconvénients :

- chaque fois que nous demandons au système de créer (et de maintenir) un index, nous augmentons sa charge de travail, et par conséquent nous le freinons. Ainsi, les opérations de saisie et de maintenance sont ralenties par la présence d'index, car ces derniers doivent être mis à jour immédiatement ;
- un index occupe de la place en mémoire sur le disque. En fait, ce dernier argument a beaucoup perdu de sa valeur avec le temps, parce que la mémoire de masse des ordinateurs ne cesse de croître rapidement, et qu'elle est devenue si bon marché (son coût à l'octet est divisé par deux tous les deux ans environ) qu'on la gaspille allégrement.

L'informatique permet de créer des index sur plusieurs champs. Imaginons que nous ayons séparé le nom et le prénom de l'auteur, par exemple. Un index sur les deux champs nom et prénom correspond en fait à l'index créé sur un champ unique dans lequel nous aurions concaténé (mis ensemble) le nom et le prénom.

Les doublons

On appelle "doublon" une information qui apparaît au moins deux fois dans une table. La notion de doublons s'applique à une colonne donnée, ou à plusieurs colonnes, ou à la totalité des colonnes d'une même table (figure ci-dessous). Dans cedernier cas, nous avons affaire à deux enregistrements (ou plus) identiques, une situation qu'il faut toujours considérer comme anormale.

A	B	C
1	aa	$
2	bb	%
3	cc	+
1	dd	&

Doublon sur une colonne

A	B	C
1	**aa**	$
2	bb	%
3	cc	+
1	**dd**	&

Doublon sur deux colonnes

A	B	C
1	**aa**	**$**
2	bb	%
3	cc	+
1	**dd**	**&**

Enregistrement dupliqué

Dans une BD, les enregistrements dupliqués peuvent provenir de deux sources :

- les erreurs de saisie. Le taux des erreurs humaines est de l'ordre de un à quelques pourcents. Il est inévitable que, de temps en temps, un opérateur tente d'introduire dans une BD des informations qui s'y trouvent déjà. Il est normal de confier au SGBD le soin de l'en empêcher ;
- la manipulation des informations contenues dans la base. Considérons par exemple la table qui illustre ci-dessus le cas du doublon sur deux colonnes. Si, pour une raison quelconque, nous supprimons la troisième colonne, nous transformons ce doublon sur deux colonnes en un enregistrement dupliqué, dont la présence peut être souhaitée (comptage), inutile ou nuisible suivant les cas.

Lorsque nous introduisons de l'information dans une table pourvue d'un index, le SGBD met ce dernier à jour en temps réel. Au cours de cette opération, il peut détecter si cette nouvelle information constitue un doublon sur les champs concernés. Il est ainsi aisé de doter le SGBD d'une fonction permettant, si on le désire, d'empêcher la validation de la saisie d'un enregistrement constituant un doublon.

Nous reviendrons, dans les chapitres relatifs aux requêtes, sur le problème de la création de doublons indésirables lors de la manipulation des informations d'une BD.

L'indexation d'un champ

Dans le chapitre consacré aux tables, nous mentionnions la "Propriété du champ" intitulée "Indexé", pour tous les types de données sauf "Objet OLE". Quand nous cliquons dans la zone de texte correspondante, une liste déroulante nous est proposée, qui contient les trois choix suivants :

- Non
- Oui - Avec doublons
- Oui - Sans doublons

Pour créer un index sur le champ correspondant, il suffit de répondre "Oui", avec ou sans doublon selon le cas. Si nous conservons la valeur "Non" par défaut, aucun index ne sera créé.

Il est inutile de ralentir le fonctionnement du système lors de la saisie des données, si cela ne nous fait pas gagner du temps ultérieurement. Il est donc préférable de répondre "Non" dans les cas suivants :

- la table considérée contient peu d'enregistrements ;
- nous effectuons rarement (voire jamais) de recherche dans ce champ ;
- nous ne trions jamais la table sur ce champ ;
- il est normal que le champ contienne des doublons. Ainsi, il est totalement inutile d'indexer un champ booléen, bien que ce soit techniquement possible.

Dans les cas contraires, la création d'un index présente de l'intérêt. Se pose alors le problème de savoir si nous admettons ou non les doublons :

- si les doublons ne posent pas de problème (ex : homonymie), nous choisirons l'option "Avec doublons". Ceci dit, indexer dans le seul but de rendre les recherches plus rapides, sans chercher à empêcher les fautes de saisie, peut constituer dans certains cas une erreur ;
- dans le cas général, nous choisirons l'option "Sans doublons", ce qui aura pour effet de nous empêcher de créer des doublons par mégarde. Le système refusera de valider l'enregistrement fautif (lors du passage à la ligne suivante, ou lors de la fermeture de la table).

Rappelons pour mémoire qu'un champ doté d'une clé est toujours indexé sans doublons. Or une table ne peut contenir qu'une seule clé, alors qu'elle peut être dotée de plusieurs index. Il faut donc réserver la clé pour la réalisation des relations, et ne pas l'utiliser comme index.

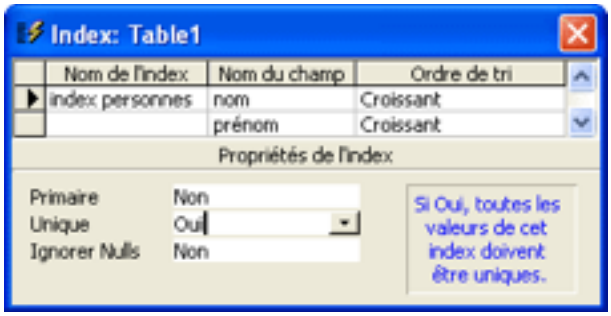
La création d'un index multi-champ

Dans une table contenant des données relatives à plusieurs milliers de personnes, le risque d'homonymie devient important. A fortiori dans une plus grande table, celle représentant l'annuaire téléphonique d'une grande ville par exemple. Pour accepter l'homonymie tout en rejetant les doublons dus à des erreurs de saisie, on utilise un index basé sur plusieurs champs. Si la probabilité de trouver deux fois "Dupont" est importante, celle de trouver deux fois "Dupont Jean" est déjà nettement plus faible, et celle de trouver deux fois "Dupont Jean né le 12/06/1978" est pratiquement nulle. En créant un index sur deux champs (Nom + Prénom) ou sur trois champs (Nom + Prénom + Date de naissance), on peut rejeter les doublons dus à des erreurs de saisie tout en tolérant parfaitement l'homonymie. On notera que le SGBD Access permet de grouper dix champs au maximum dans un index multi-champ, mais qu'on ne dépasse pratiquement jamais la valeur trois.

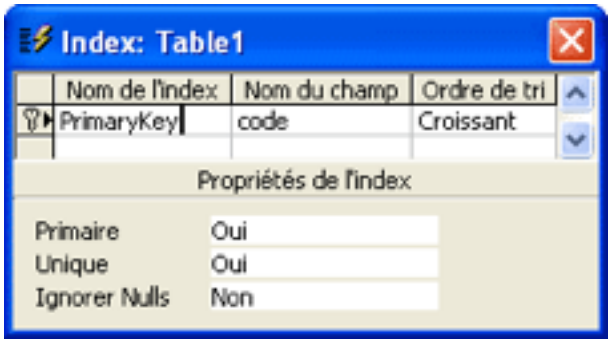
Pour créer un index multi champ, il faut se trouver en mode création de la table, et cliquer sur l'icône "Index". Une fenêtre s'ouvre, et l'on procède aux opérations suivantes :

- dans la colonne de gauche, on donne un nom à l'index multi-champ ;
- dans la colonne médiane, on écrit les uns sous les autres les noms des champs constitutifs de l'index ;
- dans la colonne de droite, on précise l'ordre de tri. Par défaut, on conserve "Croissant" ;
- on clique sur le nom de l'index puis, dans la moitié inférieure de la boite, intitulée "Propriétés de l'index", on fixe à "Oui" la propriété "Unique" si l'on désire interdire les doublons.

La figure ci-dessous représente l'état de la boite de dialogue en fin de saisie, dans le cas simple où seulement deux champs ("Nom" et "Prénom") sont concernés.



Plusieurs index multi-champ peuvent coexister dans une même table. Ils peuvent également cohabiter avec une clé. Tout ce petit monde se retrouve listé dans la fenêtre de définition des index. L'index relatif à une clé se repère à l'icône correspondante dans la colonne (grisée) la plus à gauche, à son nom "PrimaryKey", et par le fait que la propriété "Primaire" affiche "Oui", comme le montre la figure ci-dessous.



On notera pour terminer que, si un champ fait partie d'un index multi-champ, sa propriété "Indexé" vaut "Non". C'est normal, il ne faut rien y changer.

Les listes de choix

Considérons l'exemple d'une table (que nous appellerons "Personnes") constituée comme le montre l'exemple ci-dessous.

Nom	Prénom	Slts	Adresse	Ville	CP
Durand	Pierre	M.	31 rue des champs	Uriage	38410
Chose	Stéphanie	Mlle	2 place Stanislas	Nancy	54000
Trombe	Jean	M.	18 cours de la libération	Grenoble	38001
Infra	Andrée	Mme	10 cours Berriat	Grenoble	38001

Il est tout à fait fastidieux de saisir de nombreuses fois la même information, telle que celle du titre (Mme, Mlle, M.). En outre, si la liste est assez longue, le même nom de commune sera saisi à plusieurs reprises -- avec le risque d'une faute de frappe, suivie d'une erreur si l'on effectue dans la table des recherches basées sur le nom de la commune. Enfin, on n'est pas à l'abri d'une erreur de saisie conduisant à associer à une commune un code postal erroné.

Pour éviter de saisir plusieurs fois le titre ou le même nom de commune, nous pouvons l'enregistrer dans une table séparée, et travailler ensuite par copier/coller. C'est encore mieux s'il nous suffit d'indiquer au système où se trouve l'information correspondante pour l'enregistrement que nous sommes en train de renseigner.

Pour ce faire, certains SGBD sont dotés d'un outil appelé liste de choix (ou plus simplement liste), que nous allons maintenant examiner. Comme dans le chapitre précédent, nous utiliserons le SGBD MS Access comme support pratique.

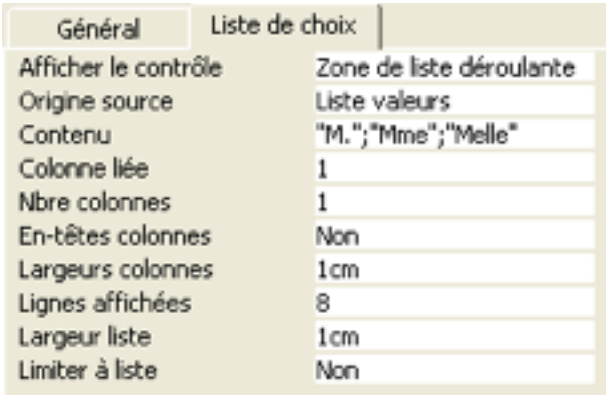
La liste simple (liste interne)

Dans un premier temps, nous créons une table "Personnes" contenant seulement les trois champs "Nom", "Prénom" et "Titre", possédant tous le type de données texte. Nous allons faire en sorte de faire écrire le titre par le système lors du remplissage de la table.

Lors de la création de la table, lorsque nous arrivons au type de données du champ "Titre", nous sélectionnons "Texte", puis "Assistant liste de choix..." dans la liste déroulante qui est à notre disposition. Nous procédons alors aux opérations suivantes :

- nous choisissons l'option "Je taperai les valeurs souhaitées". Il ne serait pas raisonnable, en effet, de créer une table pour y introduire seulement trois abréviations ;
- nous conservons le nombre de colonnes égal à 1. Nous saisissons les trois valeurs (M., Mme, Melle) les unes sous les autres dans la colonne intitulée "Col1" (utiliser la tabulation ou les flèches pour passer d'une valeur à l'autre). Enfin, nous réglons la largeur de la colonne, en la saisissant par le haut de son bord droit ;
- nous laissons le choix au système du nom de la liste (l'étiquette), et l'opération est terminée.

Dans la fenêtre de définition de la table, aux "Propriétés du champ", onglet "Liste de choix", nous trouvons les informations représentées sur la figure suivante :

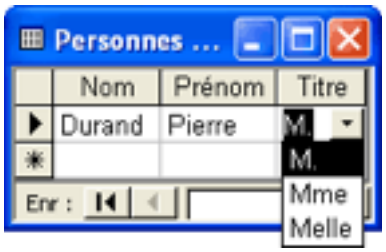


Commentons ces propriétés :

- Afficher le contrôle : Zone de liste déroulante. Une liste non déroulante conviendrait tout aussi bien, puisque la liste est fort courte, et le système ne nous proposera pas de barre de défilement. Mais attention : choisir "zone de texte" conduit à supprimer la liste, et il faudra la recréer ;

- Origine source : Liste valeurs. Pour nous rappeler que nous avons saisi la liste directement dans l'assistant ;
- Contenu : "M.";"Mme";"Melle". Les trois termes saisis sont rassemblés ici, séparés par des points-virgules, et mis entre guillemets pour rappeler qu'il s'agit de chaînes de caractères ;
- Colonne liée : 1. La colonne liée (ici la première colonne) est celle qui contient l'information que le système copiera / collera pour nous ;
- Nbre colonnes : 1. Nous n'avons demandé qu'une seule colonne ;
- En-têtes colonnes : Non. Sans objet pour nous ;
- Largeurs colonnes : 1 cm (par exemple). C'est la valeur que nous avons fixée dans l'assistant ;
- Lignes affichées : 8. C'est la valeur par défaut, mais le système limitera aux seules trois lignes utiles ;
- Largeur liste : 1 cm. C'est la largeur de l'unique colonne. La valeur "auto" convient également ;
- Limiter à liste : Non. C'est la valeur proposée par défaut. Nous reviendrons sur ce choix au paragraphe suivant.

Enregistrons et passons en mode "feuille de données" pour introduire du contenu dans la table. Quand nous cliquons dans le champ "Titre", l'icône "v" de la liste apparaît. Si nous cliquons dessus, la liste que nous avons saisie nous est proposée telle quelle par le système pour remplir le champ "Titre". Il suffit que nous cliquions sur la valeur désirée pour que le système l'inscrive à notre place, comme le montre la figure ci-dessous.



La liste obligatoire

En saisissant des données dans la table "Personnes", nous constatons que nous pouvons introduire la chaîne de notre choix dans la colonne titre. Or il serait plus judicieux que nous soyons limités aux seules trois valeurs qui ont un sens. Pour ce faire, il nous faut revenir en "mode création". Dans la ligne du champ "titre", nous cliquons dans la colonne "type de données". Dans l'onglet "liste de choix", nous réglons à "oui" la propriété "Limiter à liste". Puis nous revenons au"mode feuille de données".

Nous constatons alors que le système nous permet toujours de saisir dans le champ "Titre" une information qui n'est pas dans la liste, mais il refuse de l'enregistrer lorsque nous passons à la ligne suivante. Notre liste de titres est effectivement devenue obligatoire, le contrôle s'effectuant lors du passage à l'enregistrement suivant, ou lors de la fermeture de la table.

Rendre une liste obligatoire est une décision qui doit être prise au coup par coup, en fonction des besoins. Il est souvent utile de rendre une liste obligatoire, mais ce n'est pas une règle absolue.

La liste issue d'une table (liste externe)

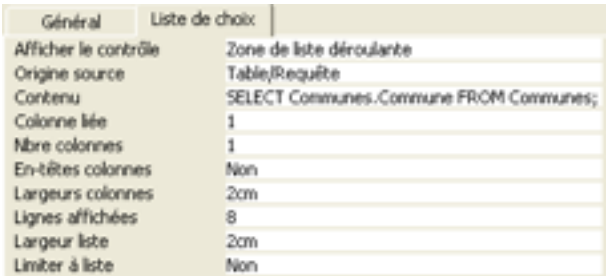
Lorsque le nombre d'éléments de la liste est important, et / ou s'il est susceptible d'être complété de temps en temps, il est plus judicieux de placer les éléments de la liste dans une table plutôt que de les saisir dans l'assistant. C'est le cas, par exemple, de la liste des communes.

La liste peut de nouveau être mise en place avec l'aide de l'assistant, mais il faut au préalable créer la table correspondante. Nous l'appelons "Communes", nous lui attribuons un seul champ (intitulé "commune"), doté du type de données "texte". Passons en mode "feuille de données" et introduisons quelques noms de communes dans la nouvelle table.

Nous retournons à la table "Personnes" en mode création, et nous rajoutons le champs "Commune" en mode texte. Nous lançons l'assistant liste de choix et nous procédons aux opérations suivantes :

- nous choisissons l'option "Je veux que la liste de choix recherche les valeurs dans une table ou requête" ;
- nous choisissons la table "Communes" ;
- nous sélectionnons son unique champ, intitulé "Commune" ;
- nous réglons la largeur de la future liste ;
- nous laissons le système régler tout seul son problème d'étiquette ;
- nous répondons "oui" à la demande d'enregistrement, et l'opération est terminée.

Les propriétés du champ "Commune" de la table "Personnes", apparaissent ainsi (onglet "liste de choix") :



Par comparaison avec la liste simple, nous remarquons les deux différences suivantes :

- la propriété "Origine source" contient maintenant "Table/Requête", ce qui est normal ;
- la propriété "Contenu" contient le code SQL "SELECT Communes.commune FROM Communes; ", ce qui signifie en clair : "choisir le champ commune de la table Communes".

De retour dans la table "Personnes" en mode "feuille de données", nous constatons que la liste contenue dans la table "Communes" nous est proposée, et que de plus elle est triée par ordre alphabétique (parce que la propriété "Indexé" est passée de "Non" à "Oui - Avec doublons").

Pour assurer la cohérence entre la table principale "Personnes" et la table "Communes", nous avons tout intérêt à rendre la liste obligatoire comme précédemment. Ceci nous astreint à renseigner le nom de commune dans la table "Communes" avant de saisir l'enregistrement correspondant dans la table "Personnes", ce qui n'est pas très commode. Nous verrons plus loin comment l'usage d'une sous-table permet de régler élégamment le problème.

La clé (clé primaire)

Nous pouvons améliorer la fiabilité du système précédent en faisant en sorte que nous ne puissions pas saisir deux fois le même nom dans la table Communes. Nous ouvrons cette dernière en mode "création", nous sélectionnons le champ "Commune", et nous cliquons sur l'icône qui représente une petite clé.

La clé (encore appelée "clé primaire") identifie de manière unique chaque enregistrement de la table. Le champ auquel on applique une clé acquière les propriétés suivantes :

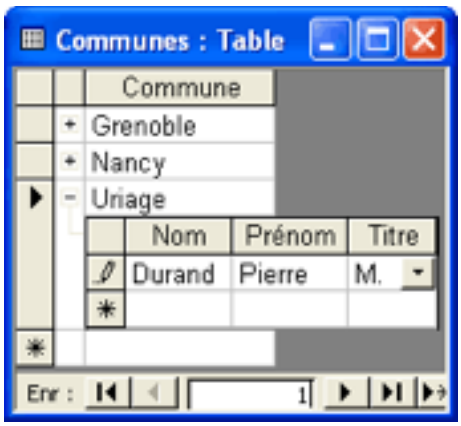
- les doublons (deux informations identiques ou plus) sont désormais interdits par le système. La propriété "Indexé" passe automatiquement à "Oui - Sans doublons" ;
- la présence de la clé interdit la présence d'un champ vide dans un enregistrement. Bien que cela n'apparaisse pas dans dans les propriétés du champ (encore un petit bug !), la valeur "Null" est désormais bannie ;
- le champ auquel on applique une clé est automatiquement trié par ordre croissant.

Pour supprimer une clé, il faut sélectionner le champ et cliquer sur l'icône de la clé ; cette icône fonctionne comme un commutateur. Notons enfin qu'il ne peut y avoir qu'une seule clé par table.

La sous-table

La présence de la clé a aussi pour effet de faire apparaître la table "Personnes" comme sous-table de la table "Communes". En effet, si nous ouvrons cette

dernière, nous voyons que chaque ligne commence maintenant par un signe + (appelé "indicateur de développement" dans Access). Si nous cliquons sur ce signe (en face de la commune "Uriage", par exemple), la liste des personnes de la table "Personnes" habitant Uriage apparaît (figure ci-dessous), et nous pouvons la compléter. Si nous cliquons sur le signe - qui se trouve maintenant en face de la commune "Uriage", la sous-table disparaît. On peut faire apparaître plusieurs sous-tables de la table "Personnes" dans la table "Communes" si on le désire.



L'existence de la sous-table nous permet de remplir simultanément la table "Personnes" et la table "Communes" qui lui sert de liste, d'autant que la liste simple du champ "Titre" fonctionne effectivement dans la sous-table. Comme on peut le constater, les sous-tables sont fort commodes, et elles rendent superfétatoire l'usage des formulaires.

Utiliser un code: exemple des codes postaux

Associons maintenant le code postal au nom de la commune. Un problème naît immédiatement du fait que, parfois, plusieurs codes postaux peuvent être attribués à une même commune. A Paris, par exemple, chaque arrondissement possède son code postal, lequel varie donc de 75001 à 75020. Si nous rajoutons une colonne "Code postal" à la table "Communes", nous serons amenés à écrire deux fois Grenoble dans la colonne "Commune", ce que la présence de la clé nous interdit.

La solution consiste à utiliser un code. Le code est une donnée qui identifie un enregistrement de manière univoque. En d'autres termes, on ne peut pas trouver dans une table deux lignes possédant le même code.

Nous rajoutons donc une colonne "Code_com" à la table "Communes", et nous attribuons la clé à cette nouvelle colonne (ce qui assure l'unicité du code). Ainsi, les couples 38000-Grenoble et 38001-Grenoble seront dotés d'un code différent. De même, il y aura 20 codes pour Paris, un par arrondissement.

La gestion du code peut être confiée au SGBD. Pour ce faire, il suffit d'utiliser le type de données NumératoAuto (entier long) pour le champ "Code". Le système attribuera les codes dans l'ordre croissant, ou de manière aléatoire, comme nous le désirons.

La première opération consiste à détruire la liste externe (voir plus haut). En mode création, nous modifions la propriété "Afficher le contrôle" (du champ "Commune" de la table "Personnes") de "Zone de liste déroulante" à "Zone de texte". Si nous repassons en mode "Feuille de données" (après avoir enregistré), nous constatons qu'il n'y a plus de liste pour alimenter le champ "Commune". Mais tout n'a pas disparu pour autant, car une liste externe implique une relation. Comme nous n'avons pas encore étudié les relations, nous ne savons pas comment les détruire. Nous allons donc procéder de la manière suivante :

- nous sélectionnons la table "Communes", nous donnons l'ordre de la supprimer, et nous confirmons ;
- le système nous alerte : "Impossible d'effacer la table 'Communes' avant la suppression de ses relations avec d'autres tables. Effacer les relations maintenant ?" ;
- nous confirmons et la table "Communes" disparaît avec les dernières traces de la liste de choix.

Nous recréons maintenant la table "Communes", qui va désormais comporter trois colonnes :

- la première colonne ("Code_com") est du type NumératoAuto. Elle est dotée de la clé ;
- la seconde colonne ("Commune") est recréée à l'identique ;

- la troisième colonne ("Code postal") est en mode texte (5 caractères), et non en numérique. En effet, certains codes postaux commencent par un zéro (exemple : 01000 pour l'Ain), et le mode numérique supprimerait le premier zéro.

Nous revenons dans le champ "Commune" de la table "Personnes", et nous relançons l'assistant "Liste de choix". Nous procédons aux opérations suivantes :

- nous choisissons "Je veux que la liste de choix recherche les valeurs dans une table ou requête" de manière à créer une liste externe ;
- nous choisissons la table "Communes" ;
- nous sélectionnons les deux champs "Code_com" et "Commune" ;
- nous conservons la coche "Colonne clé cachée" et nous réglons la largeur de la liste ;
- nous laissons le système donner un nom (étiquette) à la liste, nous cliquons sur "Terminer", et nous enregistrons.

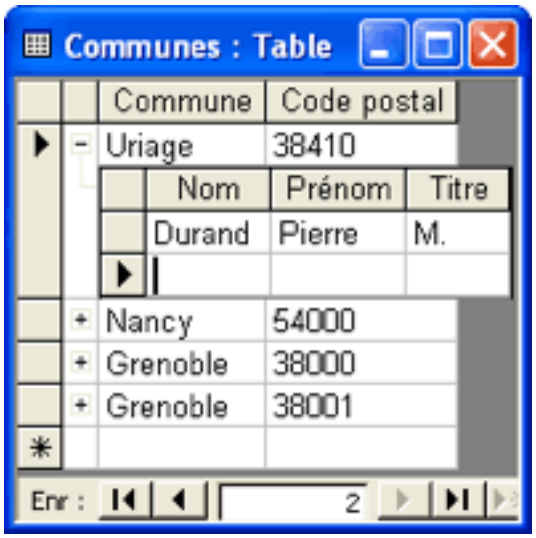
Si nous regardons ce que sont devenues les propriétés du champ "Commune" de la table "Personnes", nous constatons de sérieux changements :

- le champ "Commune" est devenu numérique, avec comme taille de champ "Entier long" ;
- la propriété "Valeur par défaut" vaut zéro. Nous supprimons cela immédiatement, pour des raisons d'esthétique uniquement ;
- le code SQL de la propriété "Contenu" a changé, ce qui est normal (la liste implique désormais deux colonnes au lieu d'une) ;
- dans la propriété "Largeurs colonnes", nous constatons que la première colonne de la liste possède une largeur nulle.

Mais nous ne sommes pas au bout de nos surprises : si nous utilisons la liste de choix ainsi créée dans la table "Personnes", nous constatons que le système propose et écrit le nom de la commune et non son code (figure ci-dessous). Des noms de commune (c'est à dire du texte) dans un champ numérique, c'est une catastrophe ! Pas de panique : le champ est, et reste, numérique. Simplement, le SGBD a affiché la traduction du code en texte, pour nous faciliter la lecture de la table.

Que le code soit caché résulte directement du fait que la largeur de sa colonne soit déclarée nulle dans les propriétés de la liste de choix. Il suffit que nous donnions une valeur non nulle à cette largeur pour que le code remplace le nom de la commune (faire l'expérience). L'opération est réversible, et nous retrouvons l'affichage de la figure ci-dessus si nous donnons de nouveau une valeur nulle à la première colonne de la liste.

Si le code "Code_com" est caché dans la table "Personnes", pourquoi ne pas en faire autant dans la table "Communes" ? Il suffit de tirer avec la souris sur le bord droit de la colonne "Code_com" jusqu'à ce que cette dernière disparaisse. La table "Communes" se présente alors (avec la sous-table "Personnes") comme le montre la figure ci-dessous. Désormais, le SGBD gère les codes, mais nous ne les voyons pas. Aucun regret !



En cas de besoin (ou si nous aimons les codes...), nous pouvons faire réapparaître le champ "Code_com". Affichons la table "Communes" en mode feuille de données

et, dans le menu, utilisons "Format > Afficher les colonnes...". La boite de dialogue "Afficher les colonnes" s'ouvre ; cochons "Code_com", et refermons.

Compléments

Comme nous pouvons le constater sur la figure ci-dessus, la champ "Commune" n'est pas trié par ordre alphabétique, ce qui n'est pas pratique du tout. Nous pouvons le trier en le sélectionnant, puis en cliquant sur l'icône "Tri croissant".

Par contre, nous pouvons faire en sorte que, dans la table "Personnes", la liste des communes apparaisse automatiquement triée par ordre alphabétique. Nous ouvrons la table "Personnes" en mode modification, nous cliquons sur le champ "Commune" puis sur l'onglet "Liste de choix", et nous modifions comme suit le code SQL de la propriété "Contenu" :

```
SELECT Communes.Code_com, Communes.Commune FROM
Communes ORDER BY Communes.Commune;
```

Nous aurions pu obtenir le même résultat en cliquant sur le code, puis sur le bouton … Une fenêtre intitulée "Instruction SQL : Générateur de requête" s'ouvre. Nous pouvons alors demander un tri croissant dans la colonne "Commune". Nous apprendrons l'usage de ce type de fenêtre lorsque nous étudierons les requêtes.

Notons que l'on peut remplacer le code SQL de la propriété "Contenu" par le nom de la table contenant la liste. C'est souvent ce que pratiquent ceux qui n'utilisent pas l'assistant pour créer leurs listes. Mais on ne peut plus demander que la liste apparaisse automatiquement triée par ordre alphabétique. Évidemment, on peut toujours la trier via l'icône "Tri croissant". En pratique, il est fortement conseillé de toujours utiliser l'assistant pour créer une liste de choix.

Arrivés à ce stade, vous souhaiteriez sans doute que le code postal s'inscrive automatiquement dans la table "Personnes", dès lors que le choix de la commune a été effectué. Dit comme tel, c'est impossible. Mais il existe deux solutions pour rassembler les informations réparties dans les deux tables "Personnes" et "Communes" :

• créer une vue via une requête portant sur les deux tables ;
• créer un formulaire basé sur les deux tables.

Ces deux opérations seront étudiées dans les chapitres suivants.

Rappel

Il existe deux façons de réaliser une liste de choix : en saisissant immédiatement les valeurs (liste interne), ou en les introduisant dans une table auxiliaire (liste externe). La première façon est recommandée lorsque la liste est courte, et peu susceptible de changer. La seconde façon est recommandée lorsque la liste est longue, et / ou susceptible d'être souvent modifiée ou complétée. Quelle que soit la manière utilisée pour réaliser une liste de choix, l'usage de l'assistant "liste de choix" est fortement recommandé.

Rappelons que le codage constitue une solution pour régler un problème d'homonymie. Si un tel problème n'est pas susceptible de se poser, le codage constitue une complication inutile. Si l'usage de codes s'avère indispensable, on peut toujours faire en sorte de ne pas les voir, alors que le SGBD continue à les gérer.

3

LES RELATIONS

Le mécanisme relationnel

Le schéma relationnel de la base

Les relations

Listes VS relations

Le mécanisme relationnel

Introduction

Les entreprises sont nées et se sont développées bien avant que l'informatique n'apparaisse. De ce fait, la plupart des logiciels ont été créés pour informatiser des opérations que l'on effectuait auparavant manuellement. Au fur et à mesure que les ordinateurs acquéraient de la puissance, et que leur coût baissait, le nombre des applications informatisables ne cessait d'augmenter. Les SGBD n'ont pas fait exception à la règle, même si on l'a passablement oublié aujourd'hui.

Tout ce que les anciens gestionnaires d'entreprise avaient inventé -- l'usage des codes, des index et des opérateurs logiques, le fractionnement des informations et leur répartition dans des "fichiers" multiples mais reliés, etc. -- a été repris (le plus astucieusement possible) pour créer les SGBD relationnels. Il faut bien reconnaître que, lors de l'informatisation des données de l'entreprise, on a beaucoup adapté, beaucoup formalisé, mais peu inventé. Cela devrait inciter messieurs les informaticiens à un peu plus de modestie.

Il est de bon ton, dans l'enseignement des BD, d'oublier tout le passé. Pire, on présente souvent les choses sous un aspect aussi abstrait que possible -- la théorie des ensembles, vous connaissez ? -- et en usant d'un vocabulaire ésotérique à souhait. Nous nous donnons donc pour but, dans ce chapitre, de montrer que le fonctionnement des bases de données relationnelles est fondé sur des techniques éprouvées, qu'il est possible d'exposer simplement, et qu'il relativement facile de comprendre et d'assimiler.

Les données redondantes

Réduire le plus possible la saisie d'informations redondantes est l'un des gros problèmes auquel se sont heurtés les gestionnaires des données de l'entreprise, avant que l'informatique ne vienne à la rescousse. Expliquons-nous à l'aide d'un exemple.

Vous travaillez dans une entreprise qui vend des matériaux de construction, et l'informatique n'existe pas encore. Vos principaux clients sont des entrepreneurs du

bâtiment et des entreprises de génie civil. Les clients les plus assidus viennent chercher des matériaux au moins une fois par jour. Quand vous écrivez dans vos livres que le camion de la "Société des Grands Travaux du Dauphiné et de la Matheysine" est venu charger 30 sacs de ciment, vous n'allez pas pour la centième fois depuis le début de l'année écrire laborieusement le nom et les coordonnées de ce fidèle client. Vous serez même lassé d'écrire seulement son nom, qui est beaucoup trop long ! Vous remplacerez ce nom par un code, qui en constitue une représentation très abrégée. Vous pouvez utiliser un code à consonance mnémotechnique (SGTDM par exemple), ou au contraire parfaitement arbitraire (530-Z, pourquoi pas).

Lorsque votre comptable exploitera le bon de livraison pour alimenter le compte du client ou pour générer une facture, il n'aura aucun mal à déterminer ce que désigne le code SGTDM. S'il ne s'en souvient plus, un fichier "Clients" est là pour l'aider. Tous les clients y ont leur fiche, et ces fiches sont classées par ordre alphabétique des codes. Chaque fiche contient tout ce qu'il faut pour identifier le client : son nom, son adresse, son numéro de téléphone, les remises consenties, etc. Toutes ces informations ont été saisies une fois et une seule, mais elles vont servir à de multiples reprises : à chaque facturation, à chaque rappel (le client paye en retard), à chaque coup de fil, à chaque envoi de publicité ciblée, etc. Bien entendu, le fait que les fiches soient classées par ordre alphabétique permet de retrouver très vite la fiche d'un code donné. Si les fiches se trouvaient dans le désordre, il faudrait en lire la moitié (en moyenne) pour retrouver la bonne.

Votre entreprise applique la même technique pour gérer son stock de produits, la liste de ses fournisseurs, son personnel, etc.

Vous n'êtes pas surpris, aujourd'hui, de voir des codes partout autour de vous (avec des codes barres, pour en rendre la lecture automatique) : dans les grandes surfaces pour les produits de consommation courante, dans les pharmacies pour les médicaments, dans les catalogues de vente par correspondance, chez le garagiste pour les pièces détachées, etc. De même que M. Jourdain faisait de la prose sans le savoir, vous baignez dans le relationnel sans vous en rendre compte.

L'informatique arrive

Le temps passe, le coût de la main d'oeuvre ne cesse d'augmenter, et les décideurs de l'entreprise font leurs comptes : informatiser la gestion des données courantes de l'entreprise va permettre de faire des économies. D'ailleurs, les concurrents suivent le même chemin, pas question de rester à la traîne.

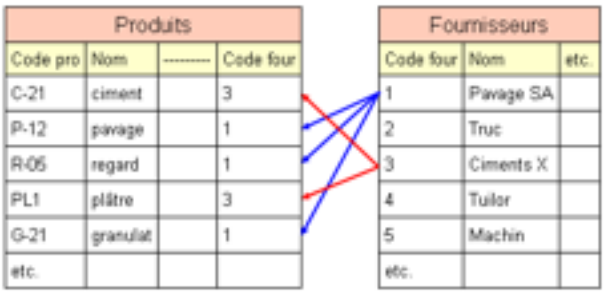
Les données que manie couramment l'entreprise sont structurées. Pour chaque produit du stock, par exemple, on enregistre les mêmes séquences de données : code, nom, prix unitaire, code du fournisseur, état du stock, état minimal admissible, quantité minimale par commande, etc. Ces données peuvent donc être rangées dans des tables, dotées du nombre de colonnes requis. La première conséquence de l'informatisation a été la dématérialisation des données : chaque fichier est devenu une table, chaque fiche un enregistrement. Au lieur de remplir les fiches à la main, ou à la machine à écrire, on saisit désormais les informations au clavier. Les informations ne résident plus sur les fiches en bristol d'une boite appelée "fichier", mais sur le disque dur d'un ordinateur.

Mais ce n'est pas tout. Le gestionnaire du stock sait que le code du fournisseur relie la fiche produit au fichier "fournisseurs", mais il est doté d'intelligence, alors que l'ordinateur en est totalement dépourvu. Il faut donc imaginer un moyen pour faire en sorte que l'ordinateur sache que la table des produits et celle des fournisseurs sont liées par une relation basée sur des codes, et qu'il la gère. Il faut donc construire une base de données relationnelle, c'est à dire capable de gérer les relations comme le faisait jusque-là instinctivement le personnel de l'entreprise.

Traduire les relations

Pour comprendre comment fonctionne une relation, il suffit de jeter un coup d'oeil à l'exemple représenté ci-dessous. Désormais, les "fichiers" sont dématérialisés et transformés en tables. La table de gauche contient la liste des produits commercialisés par l'entreprise. La table de droite contient la liste des fournisseurs. Considérons le premier produit, le ciment C-21 ; son code fournisseur vaut 3. Dans la table de droite, l'ordinateur lira que le fournisseur est la société "Ciments X", ainsi

que toutes les informations qui la concernent (l'adresse, le téléphone, le fax, les conditions consenties, etc.). Sa recherche sera d'autant plus rapide que la table "Fournisseurs" sera trié dans l'ordre croissant des codes. Bref, le SGBD maniera les relations comme le ferait un être humain.



Produits			
Code pro	Nom	--------	Code four
C-21	ciment		3
P-12	pavage		1
R-05	regard		1
PL1	plâtre		3
G-21	granulat		1
etc.			

Fournisseurs		
Code four	Nom	etc.
1	Pavage SA	
2	Truc	
3	Ciments X	
4	Tulor	
5	Machin	
etc.		

Pour rendre le système plus sûr, nous allons demander à l'ordinateur de vérifier que, dans la table "Fournisseurs", nous n'avons pas attribué deux fois le même code (cela s'appellera "l'unicité de la clé"). Puis nous allons imposer que l'on ne puisse pas introduire un produit dans la table de gauche tant que le code fournisseur correspondant n'est pas défini dans la table de droite (cela fait partie de l'intégrité référentielle). Bref, l'ordinateur fera ce que faisaient les employés qui manipulaient les données, mais il ira beaucoup plus vite, il fera plus de contrôles, et il se trompera beaucoup moins. Cela ne veut pas dire qu'il n'y a plus personne dans l'entreprise, car il faut bien que quelqu'un pilote l'ordinateur -- et reçoive le client !

Mais... tous les étudiants des écoles de commerce vous le diront : il faut, chaque fois que c'est possible, avoir deux fournisseurs par produit. Cela évite les difficultés d'approvisionnement et le dérapage excessif des prix. Comment, dans le schéma ci-dessus, traduire le fait que le ciment peut provenir soit des Ciments X, soit de la société Truc ?

Créer une nouvelle colonne "Code four" dans la table de gauche ne servirait à rien, car rien n'indiquerait dans quel cas il faut utiliser le premier code, et quand il faut utiliser le second. La bonne solution consiste à créer un nouveau code, "C-22" si vous voulez. Nul ne sera surpris que le ciment ait deux codes car, même s'il s'agit du même produit (un Portland courant, par exemple), l'emballage des sacs est différent. Les clients les plus pointilleux vous diront même que le ciment C-22 et

meilleur que le C-21. Après tout, ce n'est pas impossible, même si le ciment Portland a fait l'objet d'une norme.

Résumons-nous : la relation entre les deux tables nous permet de ne saisir qu'une seule fois les informations relatives à un fournisseur, même si ce dernier nous fournit plusieurs produits. Le mécanisme de la relation est basé sur l'usage de codes. Ce mécanisme fonctionne parce que, si un fournisseur nous livre plusieurs produits, chaque produit ne peut provenir que d'un seul fournisseur (pour qu'il en soit bien ainsi, nous avons dû créer quelques codes supplémentaires). En informatique, on parle de "relation 1-n" ou de "relation un à plusieurs". Les SGBD gèrent les relations 1-n sans problème, comme le faisaient les humains auparavant.

Un cas difficile

Continuons l'informatisation de l'entreprise, et attaquons-nous aux bons de livraison. Pour chaque commande exécutée, il faut noter le numéro de la commande, la date, le nom -- ou plutôt le code -- de l'entreprise, puis la liste des produits -- ou plutôt de leurs codes -- avec les quantités livrées. Sur papier, pas de difficulté : on peut toujours faire tenir quelques lignes ou quelques dizaines de ligne sur une feuille de papier A4. Mais, pour reporter le tout dans une table, nous rencontrons un sérieux problème, comme le montre la figure ci-dessous. Combien devons-nous prévoir de colonnes pour le code du produit et la quantité correspondante ?

No	Date	Code ent.	Code_prod	Quantité
1225	15/2/2015	SGTDM		
1226	15/2/2015	XYZT		
1227	15/2/2015	CQFD		

Si nous en prévoyons beaucoup, nous gaspillerons de la mémoire à tour de bras. Si nous en prévoyons peu, nous serons obligés de faire plusieurs bons pour une même livraison, et nous dirons en hochant la tête : "C'est la faute de l'ordinateur". Un mien collègue, en pareil cas, donnait à ses étudiants le conseil suivant : "Vous ne savez pas comment faire ? Créez une nouvelle table !". Comparé à de l'algèbre rela-

tionnelle, cela fait un peu simpliste, mais... cela marche dans bien des cas, y compris celui qui nous occupe actuellement.

La nouvelle table (dite table de jonction) comportera trois colonnes :

- la première colonne contiendra un code assurant le lien avec la table "Bons de livraison". Ce code, qui doit désigner de manière unique chaque bon de livraison, sera tout simplement son numéro ;
- la seconde colonne contiendra les codes des produits ;
- la troisième colonne contiendra les quantités livrées.

Nous écrirons autant de lignes dans cette nouvelle table qu'il y avait de produits mentionnés sur chaque bon de livraison, comme le montre la figure ci-dessous. Nous voyons dans cette table que le bon de livraison n° 1225 comporte 30 sacs de ciment (C-22), 2 palettes de pavés (P-5) et 5 regards de 40 cm (R-10), toutes informations tirées de la table "Produits", la relation étant assurée via le code produit. La table "Bons de livraison" nous montre (grâce à la relation assurée par le numéro de bon) que l'entreprise livrée est notre bon client SGTDM. Les informations le concernant se trouvent dans la table "Clients", la relation étant assurée par le code client.

No du bon	Code_prod	Quantité_prod
1225	C-22	30
1225	P-5	2
1225	R-10	5

Que s'est-il passé dans le cas des bons de livraison ? Nous nous sommes trouvés devant une relation plus complexe que précédemment. Un même bon de livraison peut mentionner plusieurs produit, et un même produit apparaît dans de nombreux bons de livraison (sinon, c'est un produit qui ne se vend pas, et il faut l'abandonner). Nous avons affaire à une relation n-n (ou plusieurs à plusieurs), difficile à gérer par des moyens informatiques, alors qu'il n'y a pas de problème avec les moyens manuels.

Heureusement pour nous, la création d'une table de jonction, puisant ses codes dans deux autres tables ("Bons de livraison" d'un côté, "Produits" de l'autre), nous a tirés d'affaire. Les informaticiens vous confirmerons qu'une relation n-n peut toujours être scindée en deux relations 1-n par création d'une table supplémentaire, laquelle est parfois appelée "table de jonction". Les mathématiciens pourront même vous en faire la démonstration rigoureuse, si vous appréciez l'abstraction.

Le schéma relationnel de la base

Nous avons vu au chapitre précédent que l'informatisation des données courantes de l'entreprise nécessite la création de plusieurs tables, reliées entre elles via des codes. Pour que le système fonctionne, il faut que les relations entre tables soient du type 1-n. Si on rencontre une relation de type n-n, on peut toujours la scinder en deux relations de type 1-n par création d'une table supplémentaire, dite table de jonction.

On attribue généralement la paternité des premiers travaux consacrés aux BD relationnelles à un chercheur de la compagnie IBM nommé Ted Codd. En 1970, il publia un article sur les bases de données relationnelles, au contenu très mathématique. Les méchantes langues vous diront que tous ceux qui publient sur le sujet des BD citent cet article, mais que fort peu l'ont lu (l'auteur de ces lignes est dans ce cas).

En termes savants, Codd voulait assurer l'indépendance entre l'organisation logique des données et la technique informatique utilisée pour les stocker. En termes simples, il cherchait une méthode permettant de stocker des données (structurées) de toute nature, sans recourir chaque fois à de la programmation spécifique. Ted Codd est considéré comme le créateur de l'algèbre relationnelle (l'aspect théorique des bases de données), qui utilise la théorie des ensembles.

En 1976, P. Chen proposa le modèle entité-relation. Depuis, ce modèle est presque universellement utilisé pour établir le schéma relationnel des BD.

Au cours des années 70, des laboratoires universitaires et des entreprises travaillèrent à mettre au point les bases de données relationnelles. A la fin des années 70, plusieurs produits arrivèrent sur le marché. A cette époque, les micro-ordinateurs étaient encore dans l'enfance, et les premiers SGBD relationnels furent implantés sur des mini-ordinateurs ou des mainframes. Progressivement, les SGBD relationnels reléguèrent aux oubliettes les SGBD hiérarchiques qui les avaient précédés.

C'est également à cette époque qu'apparut le SQL, le langage de manipulation des BD relationnelles.

Une dizaine d'années plus tard, les micros avaient acquis assez de puissance pour accueillir les SGBD relationnels. C'est alors que Microsoft introduisit la première version d'Access sur le marché. En une dizaine d'années, ce SGBD de milieu de gamme est devenu très populaire, bien qu'il reste moins connu que le traitement de texte et le tableur qui l'accompagnent dans la version professionnelle de la suite bureautique "Office".

Les entités

Le terme "entité" est utilisé de manière générique pour désigner les données. A la grande réprobation des puristes, nous utiliserons ces deux termes comme s'ils étaient synonymes.

Lorsqu'on veut gérer des données (structurées) par des moyens informatiques, la première opération consiste à les recenser, puis à les classer (dans la mesure du possible) par ordre d'importance décroissante. Un exemple relativement simple concerne les données que l'on trouve sur les cartes de visite, données que l'on peut utiliser pour se créer une liste de contacts, à l'échelle d'une personne, d'un service ou d'une entreprise. Ces données sont peu nombreuses, et elles se trouvent pratiquement rangées par ordre d'importance décroissante.

Le logo de l'entreprise mis à part, on trouve typiquement sur une carte de visite professionnelle :

- le nom de l'entreprise
- le nom et le prénom de la personne
- la fonction
- le contact : adresse, téléphone, fax, mail, etc.

Les relations 1-1

Commençons par un cas simple : le nom d'une personne et son prénom sont liés de manière univoque. Nous dirons que le nom et le prénom sont liés par une relation "un à un" ou "1-1". Nous les placerons dans la même table (que nous appellerons "Personnes"), sur la même ligne, et dans des colonnes adjacentes. D'une manière générale, nous placerons dans la même table les données qui sont en relation 1-1 entre elles. Ce sera notre première règle.

Certes, un même prénom peut être associé à des noms de famille différents, mais il ne viendrait à l'esprit de personne de se compliquer la vie pour si peu. Ce n'est pas parce que le même prénom revient toutes les 50 lignes dans une base de données qu'il faut crier à la redondance. Par ailleurs, une faute d'orthographe sur le prénom n'est pas un drame : il est peu probable que nous effectuions des recherches dans notre base de contacts en utilisant un critère basé sur le prénom.

On pourrait même songer à rassembler le nom et le prénom dans une même colonne. Cette façon de procéder est généralement considérée comme maladroite, car on n'est pas sûr de la manière dont seront saisies les informations : le nom d'abord et le prénom ensuite, ou l'inverse ? Même si une consigne est édictée, il n'est pas sûr qu'elle soit toujours respectée. Il est donc préférable de séparer les deux informations, et de les placer dans des colonnes distinctes. Cette façon de procéder est appelée l'atomisation des données. Il faut en user avec bon sens.

Les relations 1-n

Examinons maintenant la relation qui existe entre la personne et l'entreprise. Excluons pour l'instant le cas où une personne exerce plusieurs fonctions. Nous pouvons alors construire les deux phrases suivantes :

- une personne est employée par une seule entreprise ;
- une entreprise emploie (généralement) plusieurs personnes.

Nous avons affaire à une relation "un à plusieurs" ou "1-n" entre la personne et l'entreprise. Si nous plaçons le nom de l'entreprise dans la même table que le nom

de la personne, nous créons de la redondance chaque fois que nous établissons un contact avec une nouvelle personne de la même entreprise. Nous placerons donc les personnes et les entreprises dans des tables distinctes (nous appellerons la seconde "Organismes", et non "Entreprises", parce qu'une même entreprise peut comporter plusieurs établissements ou organismes : un siège social, des usines, des agences, des filiales, etc.).

D'une manière générale, chaque fois que nous rencontrerons une nouvelle relation 1-n, nous créerons une nouvelle table. Ce sera notre deuxième règle.

De plus, nous devons indiquer au système quelles sont les personnes qui font partie d'une entreprise donnée. Nous créerons donc une relation entre les tables "Personnes" et "Organismes". En pratique, nous attribuerons un code à chaque organisme, et nous utiliserons ce code dans la table "Personnes", comme le montre l'exemple ci-dessous. Nous constatons immédiatement que nous avons eu un contact avec deux personnes (Durand Pierre et Machin Jean) travaillant pour l'organisme CQFD.

Nom	Prénom	Code_Org
Durand	Pierre	3
Chose	Monique	1
Machin	Jean	3
Truc	Stéphanie	4

Table "Personnes"

Code_Org	Organisme
1	ABC
2	XYZ
3	CQFD
4	EFPG

Table "Organismes"

D'une manière générale, nous recenserons toutes les relations 1-n existant entre les données, de manière à les introduire dans le SGBD. Ce sera notre troisième règle.

Les relations n-n

L'expérience montre que l'on rencontre des personnes qui exercent dans des entreprises différentes (affiliation multiple). Le cas est même fréquent chez les cadres supérieurs, où l'on est volontiers directeur d'une usine et PDG d'une filiale. On rencontre également le cas de personnes qui partagent leur temps entre une entreprise

et un établissement d'enseignement, ou une entreprise et un syndicat patronal, etc. Si nous voulons tenir compte de ces cas en évitant la redondance, nous sommes amenés à modifier les phrases précitées :

- une personne peut être employée par plusieurs organismes (entreprise, établissement d'enseignement, syndicat patronal, association, etc.) ;
- un organisme emploie généralement plusieurs personnes.

Nous nous trouvons alors face à une relation qui semble être 1-n dans les deux sens, ce qui signifie qu'il s'agit d'une relation "plusieurs à plusieurs" ou "n-n".

Pour gérer une telle relation, il faut introduire un code dans la table "Personnes", puis créer une table supplémentaire (appelée "Affiliation"), dans laquelle on introduit les informations relatives aux couples personne-organisme, en utilisant les codes correspondants. Cette procédure est illustrée dans l'exemple ci-dessous. Nous voyons que Durand travaille pour deux organismes, CQFD et EFPG.

Code_Pers	Nom	Prénom
1	Durand	Pierre
2	Chose	Monique
3	Machin	Jean
4	Truc	Stéphanie

Table "Personnes"

Code_Pers	Code_Org
2	1
1	3
1	4
3	3

Table "Affiliation"

Code_Org	Organisme
1	ABC
2	XYZ
3	CQFD
4	EFPG

Table "Organismes"

Entre une personne et une affiliation, il existe une relation 1-n, de même qu'entre un organisme et une affiliation. Cet exemple nous montre, comme dans le chapitre précédent, que toute relation n-n peut être scindée en deux relations 1-n en introduisant une table supplémentaire appelée table de jonction. Ce sera notre quatrième règle.

Représentation du schéma relationnel

En poursuivant l'analyse des relations existant entre les données comme nous l'avons fait ci-dessus, nous dressons la liste des tables et des relations. Il est d'usage de représenter l'ensemble tables+relations dans un schéma relationnel qui se présente comme le montre l'exemple ci-dessous. Pour des raisons de simplicité, nous avons évité d'atomiser l'adresse.

Comme vous pouvez le constater, les tables sont ici représentées de manière différente. La liste des champs s'étend verticalement sous le nom de la table, de manière à pouvoir représenter correctement les relations. Ce changement de représentation est dû au fait que nous traitons ici des problèmes de structure et non de contenu.

Les relations

Nous avons vu, au chapitre précédent, comment établir le schéma relationnel d'une base de données. Pour implémenter ce schéma dans un SGBD, il faut créer des tables et des relations. Les tables ayant fait l'objet de chapitres précédents, il nous faut maintenant apprendre à créer les relations.

Comme précédemment, nous utilisons le logiciel Access de l'éditeur Microsoft comme support.

Un exemple simple

Nous commençons par un cas simple, celui où la base ne contient que deux tables liées par une relation 1-n. Pour ce faire, nous réutilisons l'exemple traité précédemment. Une table intitulée "Personnes" contient les champs "Nom", "Prénom", et "Commune". Une personne habite dans une commune et une seule, mais une commune peut héberger plusieurs personnes. Pour éviter la saisie redondante du nom de la commune dans la table "Personnes", nous avons le choix entre deux méthodes. Toutes deux impliquent la création d'une seconde table, que nous intitulerons "Communes", et qui contiendra le champ "Commune". Nous pouvons :

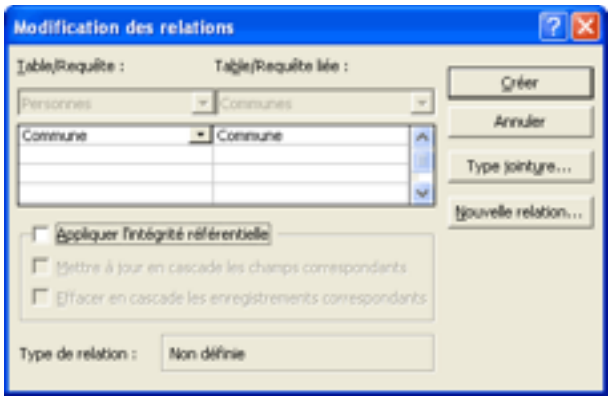
- créer une liste de choix externe dans la table "Personnes", les noms de communes provenant du champ "Commune" de la table "Communes" ;
- relier les deux tables "Communes" et "Personnes" par une relation 1-n portant sur leur champ "Commune".

Cet exemple simple nous montre qu'une liste de choix externe n'est pas différente, dans son principe, d'une relation 1-n entre deux tables. Les différences se manifestent sur le plan pratique, car les techniques utilisées pour créer une liste externe et une relation ne sont pas exactement les mêmes. Nous examinerons ces différences en détail dans le chapitre suivant.

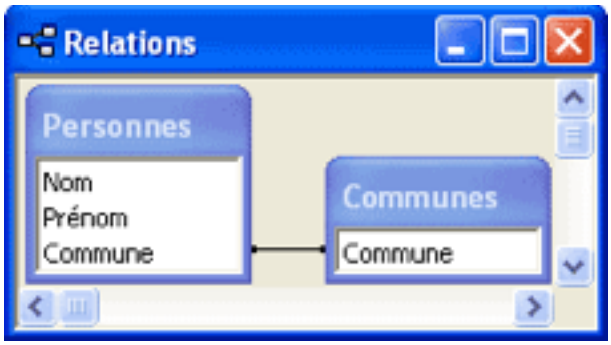
La création d'une relation

Les deux tables "Personnes" et "Communes" étant créées, et la fenêtre "Base de données" étant active, nous ouvrons la fenêtre "Relations" en cliquant sur le bouton  du même nom. Si les deux tables n'apparaissent pas, nous cliquons sur le bouton + "Afficher la table". Une fenêtre du même nom s'ouvre, qui nous permet d'ajouter les deux tables.

Pour créer la relation désirée entre les deux tables, nous cliquons sur le champ "Commune" de l'une d'elles, et nous tirons le curseur (le bouton gauche de la souris maintenu enfoncé) vers le champ "Commune" de l'autre table. Une fenêtre intitulée "Modification des relations" s'ouvre, comme le montre la figure ci-dessous.



Il suffit de cliquer sur le bouton "Créer" pour que la relation apparaisse, comme le montre la figure suivante.



Pour supprimer une relation : nous ouvrons la fenêtre "Relations", nous sélectionnons la relation d'un clic droit, nous choisissons "Supprimer" dans la liste qui s'affiche, et nous confirmons la suppression.

Utilisation de la clé

Telle quelle, la relation que nous venons de créer ne sert pas à grand'chose, parce qu'elle est dépourvue de propriétés. En particulier, le SGBD ne sait pas que la relation est du type 1-n.

La bonne démarche consiste à poser une clé sur le champ "Commune" de la table "Communes". Il en résulte que les doublons sont interdits, et que le champ est trié par ordre alphabétique croissant. C'est indispensable pour que le champ puisse servir de côté 1 dans la relation 1-n. Nous avons expliqué, au chapitre 4, comment on pose une clé sur un champ. Rappelons-le brièvement : il faut ouvrir la table "Communes" en mode modification, sélectionner le champ "Commune", et cliquer sur l'icône "Clé primaire".

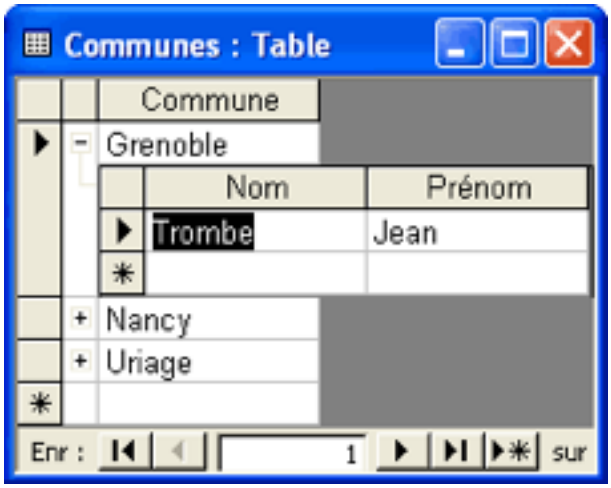
Pour vérifier que la relation est bien du type 1-n, il faut ouvrir la fenêtre "Relations", effectuer un clic droit sur la relation, choisir "Modifier une relation...", de telle sorte que la fenêtre "Modification des relations s'ouvre. Nous constatons alors que, dans le bas de cette fenêtre, la propriété "Type de relation :" est passée de "Non définie" à "Un-à-plusieurs". Le SGBD sait désormais que la relation est du type 1-n, et que le côté 1 est du côté de la clé.

Dans la fenêtre "Relations", la présence de la clé est révélée par le fait que le nom du champ correspondant est écrit en caractères gras, comme le montre la figure ci-dessous.

La présence de la clé fait a un autre effet, qu'illustre le paragraphe suivant.

La sous-table

Si nous ouvrons la table "Communes", nous constatons que nous pouvons faire apparaître "Personnes" en sous-table. Nous pouvons ainsi saisir des données dans les deux tables, sans avoir à passer de l'une à l'autre (seule la table "Communes" est ouverte). La figure ci-dessous explicite cette situation.



Introduisons quelques données dans la table, puis faisons l'expérience suivante : refermons la sous-table, sélectionnons la première ligne et supprimons-la. Le SGBD ne proteste pas. Dans la table "Personnes", l'enregistrement de M. Trombe Jean, qui habite Grenoble, est toujours présent, alors que Grenoble ne figure plus dans la liste des communes.

Dans la table "Personnes", nous pouvons introduire un enregistrement avec un nom de commune qui ne figure pas dans la table "Communes", et le SGBD ne proteste toujours pas.

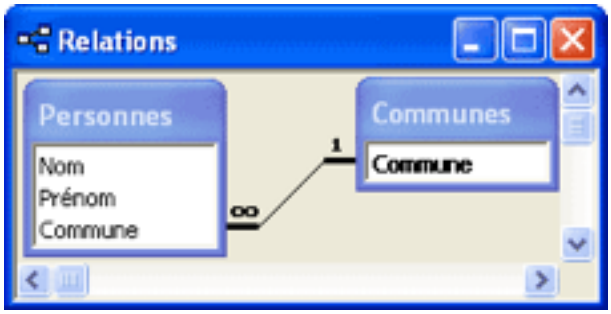
En pareil cas, on dit que la base de données manque de cohérence. La relation entre les deux tables n'est pas assez contraignante, et l'opérateur peut faire un peu n'importe quoi. Pour remédier à cette situation, il faut renforcer la relation, comme expliqué au paragraphe suivant.

L'intégrité référentielle

Dans la fenêtre "Modification des relations", un choix s'offre à nous, celui de l'intégrité référentielle. Ce terme implique que le SGBD effectue un certain nombre de contrôles, pour assurer la cohérence interne de la BD. Si nous appliquons l'intégrité référentielle :

- un nom de commune ne provenant pas de la table "Communes" sera refusé dans la table "Personnes" ;
- il ne sera pas possible de supprimer un nom de commune dans la table "Communes" s'il a été utilisé dans la table "Personnes".

Nous cochons donc la case "Appliquer l'intégrité référentielle", puis nous appuyons sur le bouton "OK". Dans la fenêtre "Relations", la présence des signes 1 et infini traduit l'application de l'intégrité référentielle, comme le montre la figure ci-dessous. On remarquera que le nom du champ qui porte la clé (et qui se trouve du côté 1 de la relation) est toujours écrit en caractères gras.



Attention ! le SGBD refusera d'appliquer l'intégrité référentielle si les deux champs liés par la relation ne possèdent pas le même type de données. Seule exception : si le champ côté 1 est du type NuméroAuto, il doit être du type numérique (entier long) du côté n. De même, le SGBD refusera d'appliquer l'intégrité référentielle si les tables contiennent déjà des données, dont certaines ont des valeurs empêchant l'intégrité référentielle de s'appliquer. Exemple : un nom de commune dans la table "Personnes" ne figure pas dans la table "Communes".

Si nous demandons l'intégrité référentielle (et il est très fortement conseillé de le faire !), le système nous propose deux autres choix. Le premier, "Mettre à jour en cascade les champs correspondants", signifie que si nous modifions l'écriture du nom d'une commune du côté 1 de la relation, cette modification sera reportée partout du côté n. D'une manière générale, il est recommandé d'activer cette mise à jour en cascade. Si nous ne le faisons pas, et si nous tentons de modifier un nom de commune (pour corriger une faute d'orthographe, par exemple), le système nous arrêtera, avec le message suivant : "Impossible de supprimer ou de modifier l'enregistrement car la table 'Personnes' comprend des enregistrements connexes". C'est clair, n'est-ce pas ?

Le second choix, "Effacer en cascade les enregistrements correspondants", signifie que si nous supprimons une donnée du côté 1 de la relation, tous les enregistrements utilisant cette donnée du côté n seront supprimés. Cela implique que, si nous supprimons par erreur un nom de commune dans la table "Communes", nous supprimons en même temps de la table "Personnes" toutes les personnes habitant cette commune. Il ne faut donc pas activer cette option, sauf momentanément et en cas de besoin spécifique.

Supposons par exemple que des noms de fournisseurs se trouvent du côté 1 de la relation, et des noms de produits du côté n. Si un fournisseur disparaît, nous pouvons activer l'effacement en cascade. Quand nous supprimons le nom du fournisseur côté 1, tous ses produits disparaissent du côté n. Nous effectuons ainsi la mise à jour de la base. Ensuite, nous décochons l'effacement en cascade pour éviter tout risque d'effacement involontaire.

Remarque : le bouton "Type jointure..." ouvre la boite de dialogue intitulée "Propriétés de la jointure". Nous étudierons la notion de jointure plus loin, dans le cadre des requêtes.

Listes VS relations

Arrivés à ce stade de notre étude des SGBD, nous sommes amenés à nous poser la question suivante : quelle est la différence entre une liste externe (basée sur une table) et une relation ? Pourquoi ne pas toujours utiliser l'une et ignorer l'autre ?

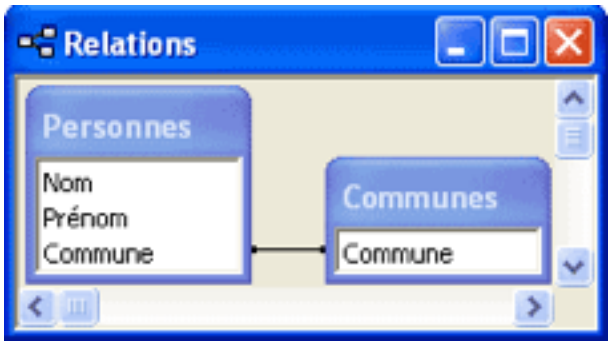
Comme nous allons le montrer, une liste externe implique une relation, et nous pourrons la doter de tous les attributs d'une relation. Une relation, par contre, ne crée pas de liste déroulante, et ne peut donc pas jouer le rôle de liste. Nous sommes tentés d'en déduire que, dans Access tout au moins, il est préférable de créer une relation sous forme de liste, puis d'attribuer les propriétés voulues à la relation sous-jacente. En fait, la conclusion finale est plus nuancée.

Comme pour les autres chapitres, nous utiliserons le SGBD Access comme support pratique de ce tutoriel (ou tutorial, ou cours en ligne).

La relation sous-jacente à une liste

Nous réutilisons l'exemple du chapitre 4, dans lequel une table appelée "Communes" sert de liste externe à une table appelée "Personnes". La table "Communes" possède un champ intitulé "Commune". La table "Personnes" possède trois champs intitulés "Nom", "Prénom" et "Commune".

Dès que la liste est créée à l'aide l'assistant "Assistant liste de choix", les deux tables se trouvent liées par une relation. IL suffit, pour s'en rendre compte, d'ouvrir la fenêtre "Relations" en cliquant sur l'icône  correspondante. Si nécessaire, on clique sur l'icône  "Afficher la table" pour ouvrir la boite de dialogue du même nom, et introduire les deux tables précitées. La figure ci-dessous représente le résultat obtenu : lorsqu'il a créé la liste, l'assistant a simultanément créé une relation, qui est en quelque sorte sous-jacente à la liste.



L'existence de cette relation n'est pas vraiment une surprise. Comme nous l'avons déjà signalé dans le chapitre précédent, à une liste externe correspond effectivement une relation 1-n.

Si nous supprimons la relation sous-jacente (clic droit, option "Supprimer"), nous constatons que la liste fonctionne comme si de rien n'était, et que ses propriétés (onglet "Liste de choix") ne sont pas modifiées. Nous en concluons que la relation sous-jacente n'est pas indispensable au fonctionnement de la liste.

Au passage, nous en déduisons la méthode qui permet de supprimer une liste. Pour l'appliquer à l'exemple que nous avons choisi :

- dans la fenêtre "Relations", nous supprimons la relation correspondant à la liste ;
- la table "Personnes" étant ouverte en mode "Modifier", nous sélectionnons le champ "Commune", puis l'onglet "Liste de choix", et nous modifions la propriété "Afficher le contrôle" de "Zone de liste déroulante" en "Zone de texte".

Une liste est aussi une relation

En règle générale, nous n'avons pas intérêt à supprimer la relation sous-jacente à une liste, car nous pouvons profiter de sa présence pour bénéficier de ses propriétés, en plus de celles de la liste.

Ainsi, si nous plaçons une clé sur le champ "Commune" de la table "Communes", nous voyons apparaître la sous-table, comportement normal d'une relation. Mais nous disposons aussi, dans la table "Personnes", de la présence de la liste permet-

tant de remplir le champ "Commune" (si la commune requise est déjà présente dans la table "Communes").

La relation sous-jacente peut également être dotée de l'intégrité référentielle, ce qui rend plus sûr le fonctionnement de la liste. Pourquoi s'en priver ?

Bref, une liste fonctionne à la fois comme une liste et comme une relation. Il est des cas où nous n'en avons pas vraiment besoin, mais il est aussi des cas où cela peut nous rendre service -- celui des tables de jonction, par exemple.

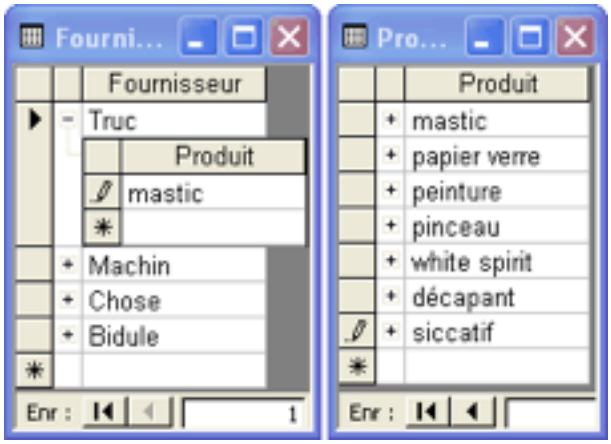
Attention ! il est impossible, dans Access, de créer une liste quand la relation correspondante existe déjà. Il faut détruire la relation, créer la liste, puis doter la relation sous-jacente des propriétés désirées.

Le cas des tables de jonction

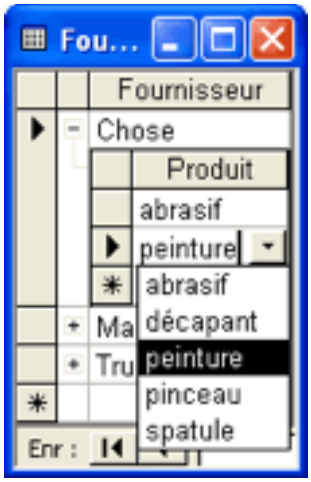
Rappelons qu'une table de jonction est une table que l'on crée pour scinder une relation n-n en deux relations 1-n. Pour illustrer ce cas, nous utiliserons l'exemple d'une liste de fournisseurs et de leurs produits.

Un même fournisseur peut fournir plusieurs produits, et un même produit peut provenir de plusieurs fournisseurs. Nous nous trouvons dans le cas classique d'une relation n-n, que nous scindons en deux relations 1-n en créant une table de jonction. Notre exemple implique donc trois tables ("Fournisseurs", "Produits", "Jonction") liées par deux relations. Nous faisons l'hypothèse qu'il n'y a pas de problème d'homonymie, ni pour les fournisseurs, ni pour les produits. Nous plaçons une clé sur le champ "Fournisseur" de la table "Fournisseurs", et une sur le champ "Produit" de la table "Produits".

Les tables se présentent comme le montre la figure ci-dessous. Pour remplir la table de jonction, il faut recopier soit le nom du produit (quand la sous-table produit est affichée), soit le nom du fournisseur (quand la sous-table fournisseurs est affichée), ce qui n'est pas admissible.

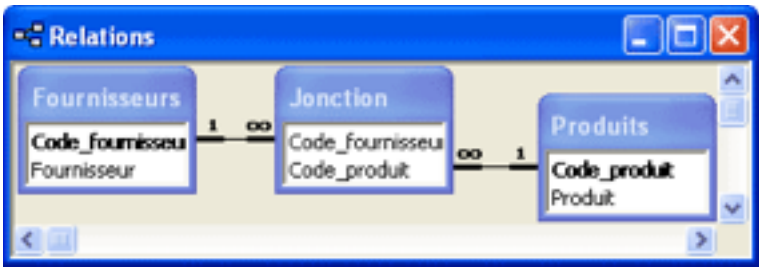


Supprimons les relations, reconstruisons-les sous forme de liste, puis dotons-les de l'intégrité référentielle. La nouvelle situation est représentée par la figure ci-dessous. Si la table "Produit" est renseignée, on peut remplir la table "Fournisseurs" et la table "Jonction" (sous-table) dans une seule fenêtre, comme le montre la figure.



L'usage des codes

Tout ce que nous avons dit au chapitre 4 sur l'usage des codes dans les listes s'applique tel quel aux relations. Supposons que, dans l'exemple que nous utilisons, nous ayons des problèmes d'homonymie pour les fournisseurs et pour les produits. Nous allons donc introduire des codes pour ces deux entités. Le schéma relationnel se trouve modifié comme suit :



Bien entendu, comme précédemment, on masque les codes, et tout se passe comme s'ils n'existaient pas quand on remplit les tables.

4

LES REQUÊTES (RETROUVER DES INFORMATIONS)

La notion de jointure

La requête de regroupement

Recherche manuelle

Dès que des données ont été introduites dans une table, des moyens simples sont à notre disposition pour y rechercher de l'information. Ces moyens, qui font partie de ce que nous appelons la "recherche manuelle", sont très spécifiques du SGBD considéré. Dans le cas d'Access, nous pouvons utiliser :

- la fonction " Rechercher". Cette fonction est présente (avec plus ou moins de perfectionnements) dans tous les logiciels qui manipulent du texte, y compris Access et les autres SGBD ;
- le tri. Nous pouvons facilement rechercher de l'information dans une colonne si elle est triée par ordre croissant. De plus, le tri croissant fait apparaître en tête de colonne la valeur la plus faible, alors que le tri décroissant fait apparaître en tête la valeur la plus forte ;
- les filtres. Appliqués à une table, ils ne laissent apparaître que les enregistrements répondant à un -- ou à quelques -- critères simples. Il existe plusieurs sortes de filtre : le filtre par sélection, le filtre hors sélection (l'inverse du précédent), le filtre par formulaire et le filtre/tri.

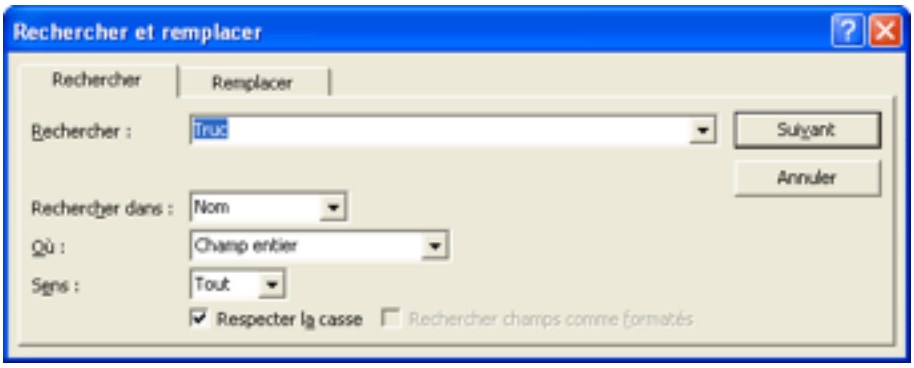
Ces techniques de recherche "manuelle" ne permettent pas d'effectuer des opérations très sophistiquées, mais elles ont le mérite de la rapidité et de la simplicité. Quand elles s'avèrent insuffisantes, il faut utiliser l'outil de recherche dont sont dotés tous les SGBD, et qui s'appelle la requête. Nous étudierons les requêtes dans les chapitres suivants.

Il existe d'ailleurs une transition presque continue entre filtres et requêtes, puisque la formulation d'un filtre élaboré fait appel aux mêmes techniques que celle d'une requête, qu'un filtre peut être enregistré comme une requête, et qu'une requête peut servir de filtre.

On notera que tout ce qui concerne le tri et les filtres s'applique non seulement aux tables, mais aussi aux formulaires.

La fonction "Rechercher"

Une table étant ouverte, et une colonne étant sélectionnée, cliquons dans le menu sur "Édition", puis sur "Rechercher..." : la fenêtre "Rechercher et remplacer" s'ouvre, l'onglet "Rechercher" étant sélectionné. Nous serions arrivés au même résultat en cliquant sur l'icône "Rechercher". Par défaut, la zone "Rechercher dans :" contient le nom de la première colonne, et la zone "Rechercher :" le premier élément de cette colonne, comme le montre l'image ci-dessous.



Il est une valeur par défaut dont il faut se méfier comme de la peste, c'est "Champ entier" dans la zone "Où :". Cette zone, en effet, propose trois options :

- N'importe où dans le champ
- Champ entier
- Début de champ

Dans le premier cas, la chaîne recherchée occupe tout ou partie du champ. Dans le second cas, beaucoup plus restrictif, la chaîne correspond exactement au contenu du champ. Dans le troisième cas, également restrictif, la chaîne occupe le début ou la totalité du champ. Le résultat d'une recherche dépend évidemment beaucoup du choix effectué, et "Champ entier" ne correspond pas forcément à ce que vous avez l'intention de faire. Attention, donc, à ce "Où :" !

Le tri

Trier une table sur une colonne donnée est une opération fort simple. La table étant ouverte, nous sélectionnons la colonne désirée en plaçant le curseur en tête de

colonne, puis en cliquant lorsqu'une petite flèche noire apparaît. Le contenu de la colonne apparaît en blanc sur fond noir. Nous cliquons alors sur l'icône pour obtenir le tri en ordre croissant, ou sur l'icône pour obtenir le tri en ordre décroissant. Si ces icônes n'apparaissent pas, nous sélectionnons "Affichage" dans le menu, puis "Barres d'outils", puis nous cochons "Feuilles de données de tables".

Si la table ne contient que quelques centaines d'enregistrements, l'opération de tri est presque instantanée. Si la table contient plusieurs centaines de milliers d'enregistrements, l'opération peut demander une minute environ, pour un ordinateur de qualité moyenne. Comme on peut le constater de visu, le tri s'effectue sur le disque dur, si bien qu'une machine possédant un disque rapide (un serveur de fichiers, par exemple) se montrera beaucoup plus rapide qu'un ordinateur ordinaire en pareil cas.

Lorsque nous refermons la table, le SGBD nous demande si nous voulons conserver la modification que nous lui avons fait subir. En cas de réponse affirmative, la table apparaît de nouveau triée quand nous la rouvrons. En fait, le SGBD Access conserve les données dans l'ordre où elles ont été initialement saisies, mais ré-applique le tri lors de l'ouverture de la table.

Le filtre par sélection et le filtre hors sélection

Le filtre par sélection. Une table étant ouverte, sélectionnons une chaîne de caractères dans l'une de ses colonnes, puis cliquons sur l'icône  "Filtrer par sélection". Tous les enregistrements disparaissent, à l'exception de ceux qui contiennent la chaîne sélectionnée dans le champ considéré. En bas de la table s'inscrit le nouveau nombre de lignes, suivi de la mention "(Filtré)".

Le retour de la table à son état initial s'obtient en cliquant sur l'icône  "Supprimer le filtre", ou en refermant la table (avec ou sans confirmation). On notera que l'icône  fonctionne comme un commutateur : le filtre étant supprimé, l'icône prend le nom "Appliquer le filtre", et cliquer de nouveau dessus a pour effet de rétablir le filtre.

Il est fréquent que le filtre par sélection facilite considérablement l'examen du contenu d'une table. Supposons par exemple que la table considérée contiennent le résultat des ventes d'une entreprise, et que dans une colonne figure le nom du commercial responsable de chaque vente. Pour obtenir l'ensemble des ventes d'un commercial donné, il suffit que nous sélectionnions son nom, puis que nous cliquions sur l'icône  . Toutes les ventes ne le concernant pas disparaissent instantanément. Le filtre par sélection est un outil simple -- mais extrêmement rapide -- d'analyse des données contenues dans une table.

Attention ! le filtre par sélection fonctionne comme la fonction "Rechercher", en ce sens que la position de la chaîne sélectionnée importe beaucoup. Si cette chaîne :

- n'est pas en contact avec les extrémités du champ, le SGBD retient les enregistrements qui contiennent cette chaîne (n'importe où dans le champ) ;
- se trouve au début du champ, le SGBD retient les enregistrements dont le champ commence par cette chaîne ;
- se trouve en fin de champ, le SGBD retient les enregistrements dont le champ finit par cette chaîne.

Le filtre hors sélection. Ce filtre fonctionne à l'opposé du filtre par sélection : tous les enregistrements disparaissent, à l'exception de ceux qui ne contiennent pas la chaîne sélectionnée. Mais il n'existe pas d'icône qui corresponde au filtre hors sélection, si bien qu'il nous faut passer par le menu. Nous cliquons sur "Enregistrements", puis sur "Filtrer", et enfin sur "Filtrer hors sélection".

Nous pouvons appliquer un filtre par sélection au résultat d'un précédent filtre par sélection, de manière à affiner une recherche. En d'autres termes, les filtres par sélection sont emboîtables.

Le filtre/tri

Le tri que nous avons considéré plus haut ne concerne qu'une seule colonne, c'est un tri simple. Dans certains cas, nous avons besoin d'effectuer un tri sur plusieurs

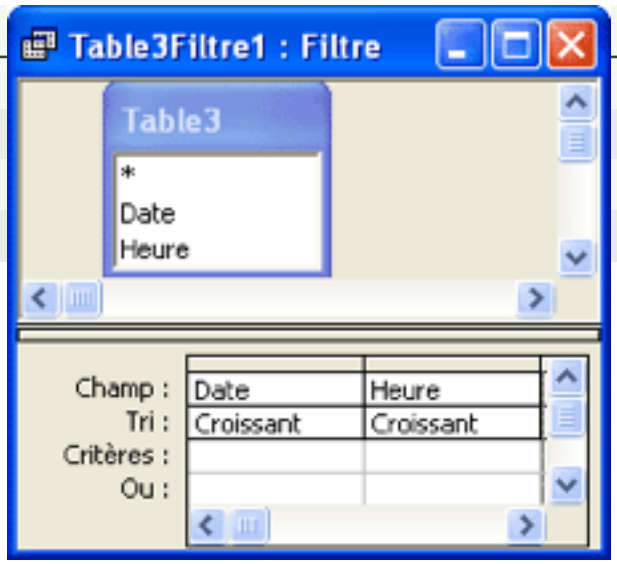
colonnes, encore appelé tri multiple. Le filtre/tri est l'outil qui nous permet d'arriver à nos fins sans avoir besoin de créer une requête.

Considérons l'exemple du fichier journal d'un site web, dans lequel chaque ligne correspond à une requête (une demande de fichier). Importé dans un SGBD, ce fichier devient une table, dans laquelle la première colonne contient la date, la seconde colonne l'heure, etc. Mais l'importation, et les manipulations qui la suivent, peuvent perturber l'ordre dans lequel les requêtes ont été traitées par le serveur web. Pour rétablir cet ordre, il faut que nous puissions trier la table sur la date d'abord, et sur l'heure ensuite.

Pour ce faire, nous cliquons dans le menu sur "Enregistrements", puis sur "Filtrer", et enfin sur "Filtre/tri avancé...". S'ouvre une fenêtre "Filtre" qui ressemble, à s'y méprendre, à celle qui permet de définir une requête. A noter que le filtre/tri possède une icône , mais que cette dernière ne se trouve pas en standard dans la barre d'outils "Feuille de données de table". On peut l'y introduire par personnalisation de la barre d'outils.

Nous remplissons la grille comme indiqué dans la figure ci-dessous, puis nous appliquons le filtre en cliquant sur l'icône "Appliquer le filtre", enfin nous refermons la fenêtre "Filtre". Nous vérifions que la table est effectivement triée comme nous l'avons demandé.

Date	Heure
10/9/2015	16:21:36
11/9/2015	10:01:25
10/9/2015	12:23:18
9/9/2015	13:40:01



Date	Heure
9/9/2015	13:40:01
10/9/2015	12:23:18
10/9/2015	16:21:36
11/9/2015	10:01:25

Si nous refermons puis rouvrons la table, nous constatons que le filtre/tri agit toujours : la table reste triée. Si nous cliquons comme précédemment sur "Enregistrements", puis sur "Filtrer", et enfin sur "Filtre/tri avancé...", le filtre/tri s'affiche de nouveau comme nous l'avons défini, et nous pouvons le modifier à loisir. Par contre, si nous créons une nouvelle colonne autonumérotée, nous constatons que la table reprend son ordre initial, comme dans le cas d'un tri simple. Nous constatons de plus que le filtre/tri est de nouveau vierge, ce qui montre que toute modification notable de la table fait disparaître le filtre/tri. Dans le cas où la table n'est pas modifiée, par contre, le filtre/tri la suit comme son ombre.

La ligne "Critères :" permet, comme dans une requête, de sélectionner les enregistrements à afficher sur des critères plus ou moins complexes. Les techniques correspondantes sont exposées dans les chapitres relatifs aux requêtes.

Le filtre/tri est un outil intéressant, car il permet de doter une table d'un tri permanent plus ou moins élaboré. Chaque fois que nous ouvrons la table, elle se présente sous la forme triée que nous désirons, sans que nous ayons à intervenir.

L'enregistrement d'un tri ou d'un filtre

Ouvrons la table contenant des noms, et faisons-lui subir un tri par sélection sur "Truc". Ouvrons ensuite la fenêtre "Filtre" : le terme "Truc" apparaît sur la ligne "Critères :", les guillemets indiquant qu'il s'agit d'une chaîne de caractères. La fenêtre "Filtre" révèle donc à la fois le stockage des filtres et celui des tris.

Si nous appliquons successivement à une table un filtre par sélection, puis un filtre/tri, nous retrouverons trace des deux opérations dans la fenêtre "Filtre", comme le montre la figure ci-dessous, relative à la table contenant des dates et des heures.

Champ :	Date	Heure
Tri :	Croissant	Croissant
Critères :	#02/03/2003#	
Ou :		

Un filtre/tri peut être enregistré : il devient alors une requête. Pour ce faire, la fenêtre filtre étant ouverte, nous cliquons sur l'icône  "Enregistrer en tant que re-

quête". Pour distinguer le filtre d'une requête, nous lui donnons un nom commençant par "Filtre...". Par précaution, nous rajoutons le nom de la table à laquelle il s'applique. Exemple : Filtre3_table5.

Pour réutiliser le filtre ainsi enregistré, nous ouvrons la table concernée, puis la fenêtre "Filtre", et nous cliquons sur l'icône "Charger à partir d'une requête". La liste des requêtes s'affiche, dans laquelle nous choisissons le filtre désiré. Ce dernier s'affiche dans la grille, et nous pouvons l'appliquer en cliquant sur l'icône .

Un filtre enregistré sous forme d'une requête bénéficie de toutes les propriétés de ces dernières.

Introduction aux requêtes

Nous savons désormais stocker des informations structurées dans les tables d'une base de données relationnelle. Cette étape franchie, il nous faut maintenant apprendre à gérer ces informations, et à retrouver celles dont nous avons besoin quand cela s'avère nécessaire.

Une base de données a besoin de maintenance. Il faut pouvoir supprimer les informations obsolètes après les avoir archivées. Il est, par exemple, inutile de laisser traîner dans une BD des données relatives à des factures qui ont été réglées, et qui sont relatives à un exercice clos.

Une base de données est souvent une mine d'informations, en particulier dans le domaine économique et financier. Il est très important pour le bon fonctionnement d'une entreprise que ces informations puissent être retrouvées rapidement et simplement par les personnes qui en ont besoin et qui sauront en faire bon usage. Pour ce faire, la requête constitue l'outil adéquat. La requête est, par ordre d'importance décroissante , le deuxième "objet" des BD après la table.

Les trois fonctions des requêtes

L'outil requête a trois fonctions principales :

- la réalisation de vues présentant tout ou partie de l'information contenue dans la BD. Dans une base de données relationnelle, les données sont éparpillées dans de multiple tables, liées par des relations, et contenant souvent des codes non explicites. Pour appréhender, en partie ou en totalité, le contenu de la base, il faut rassembler les données utiles dans une seule table, que l'utilisateur peut consulter directement ou via un formulaire. Pour ce faire, on sélectionne des colonnes dans différentes tables, et on met les lignes en correspondance grâce aux relations ;
- la maintenance de la BD. Cette opération consiste à archiver et / ou supprimer des enregistrements obsolètes, mettre à jour des données révisables, rechercher et supprimer les doublons indésirables, etc. Elle concerne des lignes particulières, mais le nombre de colonnes n'est pas modifié ;

- la recherche d'information dans la BD. Cette opération consiste à créer une sous-table contenant les enregistrements répondant à certains critères et appartenant à certains champs. Elle porte à la fois sur les lignes et les colonnes d'une table, ou de plusieurs tables liées par des relations.

Les différents types de requêtes

Pour assurer les trois fonctions précitées, différents types de requêtes ont été créés, que l'on retrouve dans presque tous les SGBD. On peut les classer ainsi :

- La sélection simple ou projection permet de réaliser les vues précitées ;
- La sélection est l'outil de recherche d'information par excellence, même si ce n'est pas le seul qui soit utilisé. Cette requête est dotée de deux perfectionnements importants (la jointure et le regroupement) ;
- Les opérations ensemblistes (dont la plus importante est l'union), auxquelles on peut associer l'ajout. Elles permettent de regrouper dans une même table des enregistrements provenant de deux tables différentes ;
- Les requêtes de maintenance sont principalement la mise à jour et la suppression. La première permet de modifier le contenu de certains champs, la seconde de supprimer certains enregistrements ;
- L'analyse croisée est une spécificité d'Access. Comme son nom l'indique, c'est un outil d'analyse qui permet, sous certaines conditions, de réorganiser complètement une table.

Le SGBD Access permet de créer des requêtes en utilisant soit une interface graphique, soit le langage SQL. Nous étudions tour à tour ces deux possibilités :

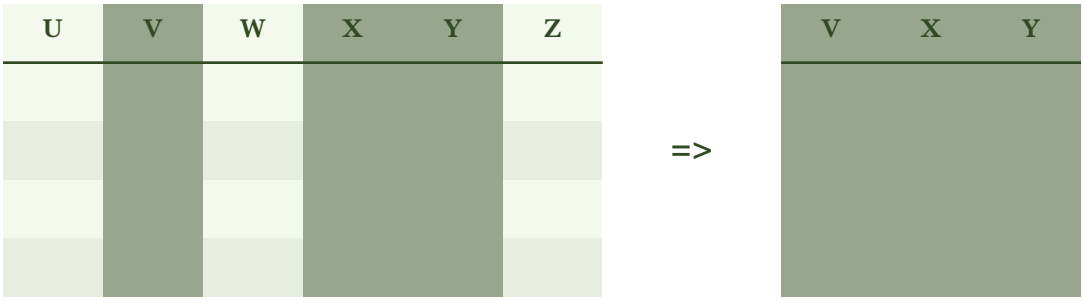
- Interface graphique. La sélection simple (ou projection) fait l'objet du chapitre 11. La sélection est étudiée dans le chapitre 12, et ses perfectionnements dans les trois chapitres suivants. L'ajout et l'analyse croisée sont regroupées dans le chapitre 16. La mise à jour et la suppression sont étudiées dans le chapitre 17.
- Langage SQL que nous n’aborderons pas ici.

Dans MS Access, la création d'une requête union n'est possible qu'en SQL.

La sélection simple

Dans ce chapitre (le second d'une série de quatre consacrés aux requêtes), nous apprendrons à réaliser des opérations de sélection simple (encore appelée projection). La sélection simple opère sur les colonnes. Il n'y a pas de critère de sélection relatif au contenu des enregistrements, et de ce fait le nombre de lignes reste inchangé.

La figure ci-dessous représente schématiquement une table contenant 7 colonnes. Grâce à une sélection simple (ou projection), nous pouvons reconstituer une table ne contenant que les colonnes V, Y et Z (colorées en jaune).



En fait, le nombre de lignes peut diminuer quelque peu. C'est le cas lorsqu'on élimine les doublons, ou lorsqu'on effectue une requête basée sur plusieurs tables et qu'il manque des informations dans certaines d'entre elles.

Nous déborderons quelque peu du cadre de la sélection simple, pour apprendre à mettre en forme l'information obtenue, par élimination des doublons, concaténation de chaînes, etc.

La création d'une requête

A titre de premier exemple de sélection simple, nous allons créer une requête qui extrait d'une table une liste de personnes désignées par leur nom et leur prénom. Notre point de départ sera la table "Personnes" représentée ci-dessous.

Nom_Personne	Prenom	Nom_Organisation	Fonction
Turlutu	Jean	Chose et Cie	technicien
Surpont	Yvette	EFPG	secrétaire
Lechant	Paul	Société Machin	directeur
Durand	Nathalie	Entreprise Truc	ingénieure
Lechant	Paul	Association Z	président
Verseau	Pierre	Bidule SA	commercial

Notons d'abord qu'une requête opère sur une ou sur plusieurs tables. On ne peut donc pas créer de requête dans une base de données vide. Certes, le SGBD Access ne refusera pas d'ouvrir la fenêtre de création d'une requête dans une base vide, mais si aucune table n'est présente, nous ne pourrons rien faire d'autre que créer une requête vide.

Ouvrons donc la BD contenant la table "Personnes" représentée ci-dessus. Dans la fenêtre "Base de données", sélectionnons l'objet "Requêtes". Double cliquons sur "Créer une requête en mode création". Une fenêtre intitulée "Requête1 : Requête Sélection" s'ouvre, ainsi qu'une boite de dialogue intitulée "Afficher la table". Cette boite affiche la liste des tables que contient la BD. Nous sélectionnons la table "Personnes" sur laquelle doit porter la requête, puis nous cliquons successivement sur les boutons "Ajouter" et "Fermer". La table "Personnes" est maintenant présente dans la moitié haute de la fenêtre de création de la requête.

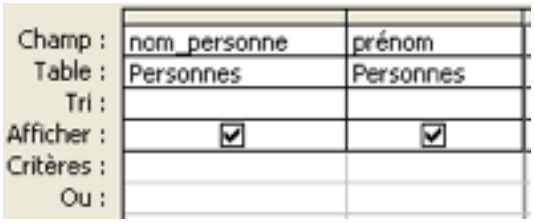
La moitié basse contient la grille de définition de la requête. Pour y introduire un champ (on notera au passage que l'astérisque représente la totalité des champs), nous disposons de trois méthodes :

• cliquer sur la ligne "Champ :" et choisir dans la liste déroulante qui s'affiche ;

• double cliquer sur le nom du champ ;

• tirer le nom du champ avec la souris de la table vers la grille.

Pour extraire de la table "Personnes" les deux premières colonnes, nous introduisons dans la grille les champs correspondants. Sur la ligne "Afficher :", les cases doivent être cochées (elles le sont par défaut). La figure suivante est extraite de la grille de définition de la requête :



La requête étant définie, nous l'exécutons en cliquant sur le bouton de la barre d'outils. Nous obtenons le résultat suivant :

Nom_Personne	Prenom
Turlutu	Jean
Surpont	Yvette
Lechant	Paul
Durand	Nathalie
Lechant	Paul
Verseau	Pierre

Nous voyons que, comme une table, une requête présente un double aspect :

- l'aspect structure, lequel est défini en mode création ;
- l'aspect résultat, qui est représentée par une table à l'existence volatile, laquelle s'appelle "feuille de données" dans la terminologie de Microsoft.

Comme pour une table également, on peut passer rapidement d'un aspect à l'autre en cliquant dans la barre d'outils sur le bouton (en mode feuille de données), ou le bouton (en mode création).

Pour conserver la structure de la requête, il suffit de cliquer sur l'icône "Enregistrer", de donner un nom (par exemple, "Sélection des personnes") à la requête

dans la boite de dialogue qui s'ouvre, et de confirmer. Ce nom figurera désormais dans la fenêtre "Base de données" (l'objet "Requêtes" étant sélectionné), précédé de l'icône  , pour rappeler qu'il s'agit d'une requête de sélection. Si nous fermons la fenêtre de définition de la requête sans avoir préalablement enregistré la structure, le SGBD nous demande si nous voulons conserver la requête. Dans l'affirmative, la boite de dialogue s'ouvre, et nous procédons comme précédemment.

Mais le résultat de la requête a disparu ! Pour retrouver cette "feuille de données" volatile, il faut relancer la requête, soit en double-cliquant sur son nom, soit en la sélectionnant et en cliquant sur le bouton  "Ouvrir" (lequel devrait plutôt s'appeler "Exécuter").

La requête avec création de table

Le résultat d'une requête est une table, et il peut être enregistré comme tel. Pour ce faire, nous sélectionnons la requête précédente, et nous cliquons sur le bouton "Modifier". La requête s'ouvre en mode création. Nous cliquons sur le bouton  de la barre d'outils et, dans la liste déroulante qui s'affiche, nous sélectionnons "Requête Création de table...". Une boite de dialogue s'ouvre, dans laquelle nous renseignons le nom de la table ("Liste de personnes", par exemple). Dans la liste des requêtes, "Sélection des personnes" apparaît maintenant avec l'icône  , qui rappelle que le résultat de la requête est enregistré dans la base sous forme d'une table.

Exécutons la requête : deux boites d'alerte s'ouvrent successivement. Pas de panique, répondons "oui" dans les deux cas. Si la table existe déjà, une troisième boite d'alerte prévient de son écrasement. Que de précautions ! (Si ces alertes vous agacent, vous pouvez les supprimer en utilisant la rubrique "Outils" du menu. Cliquez sur "Options...", puis sur l'onglet "Modifier/Rechercher, et décochez les cases de la zone "Confirmer").

Nous pouvons maintenant vérifier, dans la fenêtre "Base de donnée" (l'objet "Tables" étant sélectionné), que la table "Liste de personnes" a bien été créée. Si nous l'ouvrons, nous constatons que son contenu correspond bien à la structure de la requête "Sélection des personnes".

Comment faire pour qu'une requête ne crée plus de table ? Il semble que l'éditeur Microsoft n'ait pas prévu la chose en mode graphique, si bien qu'il faut passer en mode SQL. La fenêtre de création (ou modification) de la requête étant ouverte, nous cliquons sur la petite flèche adjacente à l'icône  "Affichage". Dans la liste déroulante, nous choisissons "Mode SQL", et la traduction de notre requête en langage SQL s'affiche. Dans la première ligne du code, nous repérons le terme "INTO" suivi du nom de la table (éventuellement écrit entre crochets). Nous les supprimons tous les deux, nous refermons la fenêtre, et nous confirmons la modification de la requête.

Le tri simple et le tri multiple

On ne peut retrouver rapidement des informations dans une liste que si elle est triée (par ordre alphabétique). Or la liste des personnes que crée notre requête présente le défaut d'être présentée dans l'ordre où les informations ont été saisies. Une table, en effet, se remplit toujours par la ligne la plus basse. Pour trier la table, nous pouvons utiliser le bouton  , mais il est plus pratique de rendre l'opération automatique. Sélectionnons la requête, et cliquons sur  . Dans la grille, cliquons à l'intersection de la colonne "nom_personne" et de la ligne "Tri :". Une liste s'affiche, qui nous propose les trois options possibles : croissant, décroissant et non trié. Choisissons "croissant", refermons la fenêtre, confirmons la modification, et relançons la requête : la table "Liste de personnes" s'affiche désormais par ordre alphabétique des noms, comme le montre la feuille de données ci-dessous.

Nom_Personne	Prenom
Durand	Nathalie
Lechant	Paul
Lechant	Paul
Surpont	Yvette
Turlutu	Jean
Verseau	Pierre

Nous pouvons également demander le tri croissant dans le champ "prénom". Si deux personnes portent le même nom, elles apparaîtront dans l'ordre croissant de leurs prénoms respectifs. Attention ! ce tri multiple s'exécute de gauche à droite : par les noms d'abord, par les prénoms ensuite. Si nous voulons obtenir le résultat inverse, il faut que nous placions la colonne nom à droite de la colonne prénom dans la grille de création de la requête. Pour ce faire, il faut sélectionner (par le haut) la colonne à déplacer, puis la tirer (toujours par le haut) jusqu'à sa nouvelle position.

L'élimination des doublons

Dans la table "Liste de personnes", Paul Lechant apparaît à deux reprises : nous avons affaire à un doublon, une information répétée deux fois ou plus. Dans la table "Personnes" de départ, cette double apparition de Paul Lechant était justifiée par deux affiliations distinctes. La sélection a fait disparaître les informations correspondant à l'affiliation et créé le doublon. Nous pouvons faire en sorte que les doublons soient éliminés du résultat :

- grâce à une modification des propriétés de la requête ;
- grâce à une opération de regroupement.

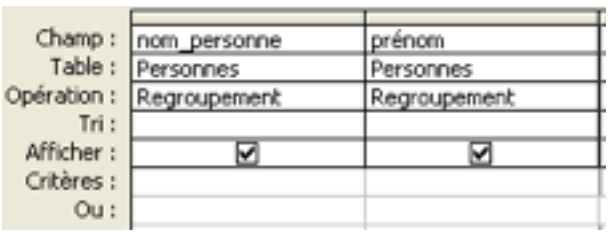
Première méthode. Ouvrons la requête "Requête1" en mode création. Effectuons un clic droit dans la fenêtre de définition de la requête et sélectionnons "Propriétés" dans la liste déroulante, ou cliquons sur l'icône  "Propriétés". La boite de dialogue "Propriétés de la requête" s'ouvre. Modifions la propriété "Valeurs distinctes" de "Non" à "Oui". Fermons la boite de dialogue, et basculons en mode feuille de données : les doublons ont disparu, comme le montre la feuille de données ci-dessous.

Nom_Personne	Prenom
Durand	Nathalie
Lechant	Paul
Surpont	Yvette
Turlutu	Jean
Verseau	Pierre

L'opération est réversible : si nous basculons en mode création, ramenons la propriété "Valeurs distinctes" de "Oui" à "Non", et rebasculons en mode feuilles de données, les doublons sont de retour.

Deuxième méthode. Comme son nom l'indique, l'opération de regroupement consiste à rassembler les lignes d'une table qui ont quelque chose en commun -- la même valeur dans un champ donné, par exemple. Au cours de l'opération de regroupement, les doublons sont automatiquement éliminés. On se sert habituellement du regroupement pour effectuer des calculs sur des groupes de lignes, au lieu de les effectuer sur la table entière. Mais on peut aussi utiliser le regroupement pour éliminer les doublons.

Créons une requête simple basée sur la table "Personnes", et sélectionnons les deux champs "nom_personne" et "prénom". Cliquons sur l'icône Σ "Totaux" : une nouvelle ligne (intitulée "Opération :") apparaît dans la grille de définition de la requête, avec la mention "Regroupement" déjà inscrite par défaut pour chacun des deux champs (si cette mention n'apparaît pas, il faut cliquer à l'endroit correspondant, et choisir "Regroupement" dans la liste déroulante qui s'affiche). La requête se présente comme le montre la figure ci-dessous.

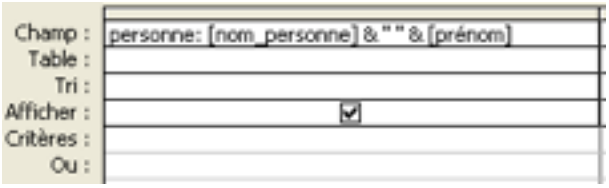


Basculons ensuite en mode "feuille de données" : les doublons ont disparu et la feuille de données est triée par ordre alphabétique croissant.

Notons que ces deux technique éliminent également les doublons éventuellement présents dans la table de départ.

La requête avec création de champ

Dans la liste des personnes, nous voulons maintenant rassembler chaque nom, suivi de son prénom, dans une même colonne. Pour ce faire, nous créons la requête suivante :



La signification du contenu de la ligne "Champ :" de la grille ci-dessus est la suivante :

- la requête crée une feuille de données contenant une colonne intitulée "personne" ;
- chaque ligne contiendra le nom, puis un espace, puis le prénom. Ces données proviendront de la table située au-dessus de la grille.

Le signe & désigne l'opérateur de concaténation de chaînes. Les crochets [.....] signifient que l'on évoque le contenu des champs correspondants. L'espace qui sépare le nom du prénom est mis entre guillemets pour rappeler qu'il s'agit d'une chaîne de caractères. Le résultat de la requête est le suivant :

Nom_Personne
Durand Nathalie
Lechant Paul
Surpont Yvette
Turlutu Jean
Verseau Pierre

De la même manière, on peut concaténer le code postal avec un tiret suivi du nom de la commune, reconstituer une adresse complète, etc. Cette technique de reconstitution de chaînes est intéressante parce que, au nom du principe d'atomisation, les informations situées dans une BD sont divisées le plus possible en petits morceaux.

De manière plus générale, une requête avec création de champ permet d'effectuer des opérations (numériques ou sur chaînes de caractères) sur le contenu des champs d'un même enregistrement, c'est à dire horizontalement. On peut effectuer des opérations verticalement dans une table (en utilisant ou non la notion de regroupement), mais on obtient une meilleure présentation en se servant des états, que nous étudierons dans un chapitre ultérieur.

Les requêtes emboîtées

Une requête peut prendre comme point de départ la feuille de données résultant de l'exécution d'une autre requête. Il suffit de lancer la seconde requête pour que la première s'exécute en premier lieu. On peut généraliser, et créer une chaîne de requêtes qui s'exécutent dans l'ordre par simple lancement de la dernière. Il faut simplement veiller à ce que chaque requête (à l'exclusion de la dernière) ne crée pas de table. Sinon, le logiciel proposera de partir de cette table, et la chaîne sera rompue.

A titre d'exemple, créons les requêtes suivantes :

- la requête n° 1 extrait les colonnes nom et prénom de la table "Personnes", et trie par ordre croissant des noms ;

- la requête n° 2 part du résultat de la requête n° 1 et élimine les doublons (par modification de propriété, ou par regroupement) ;
- la requête n° 3 part du résultat de la requête n° 2 et concatène nom et prénom.

Il suffit de lancer la troisième requête pour que l'ensemble s'exécute et fournisse le résultat obtenu au paragraphe précédent. Nous avons ainsi créé un automatisme élémentaire. Nous verrons dans un chapitre ultérieur que l'on peut obtenir le même résultat avec une macro.

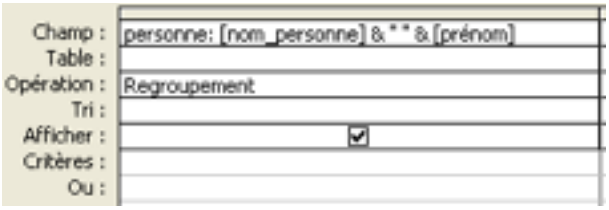
Il ne faut pas abuser de l'emboîtement, et les professionnels conseillent généralement de ne pas emboîter plus de 3 requêtes à la file. Il y a plusieurs raisons à cela :

- si une requête est utilisée plusieurs fois dans une application, toutes les requêtes emboîtées qui la précédent seront re-exécutées. On allonge ainsi le temps machine requis pour l'application ;
- si une requête faisant partie d'un emboîtement contient une erreur, cette erreur sera signalée par le système (du moins par le SGBD Access) comme faisant partie de la dernière requête de l'emboîtement. L'emboîtement rend donc la correction des erreurs plus difficile ;
- l'emboîtement se programme malaisément lorsqu'on utilise le langage SQL, et le risque d'erreur croit avec le nombre de requêtes emboîtées.

La requête multifonctionnelle

Pour des raisons didactiques, nous avons créé une nouvelle requête pour chaque opération que nous voulions réaliser. Dans la pratique, nous éviterons de multiplier les requêtes, en regroupant le plus possible les opérations à effectuer dans une même requête.

Ainsi, la requête représentée par la figure ci-dessous permet d'obtenir le résultat final (la liste des noms concaténés avec les prénoms, dans l'ordre alphabétique, et sans doublons) en une seule étape :



et on peut lui demander en plus de créer une table si on le désire. On notera qu'il n'est pas nécessaire que le nom de la table figure dans la grille (mais la table doit être présente au-dessus de la grille), et qu'il est inutile de spécifier un tri car ce dernier est implicite en cas de regroupement.

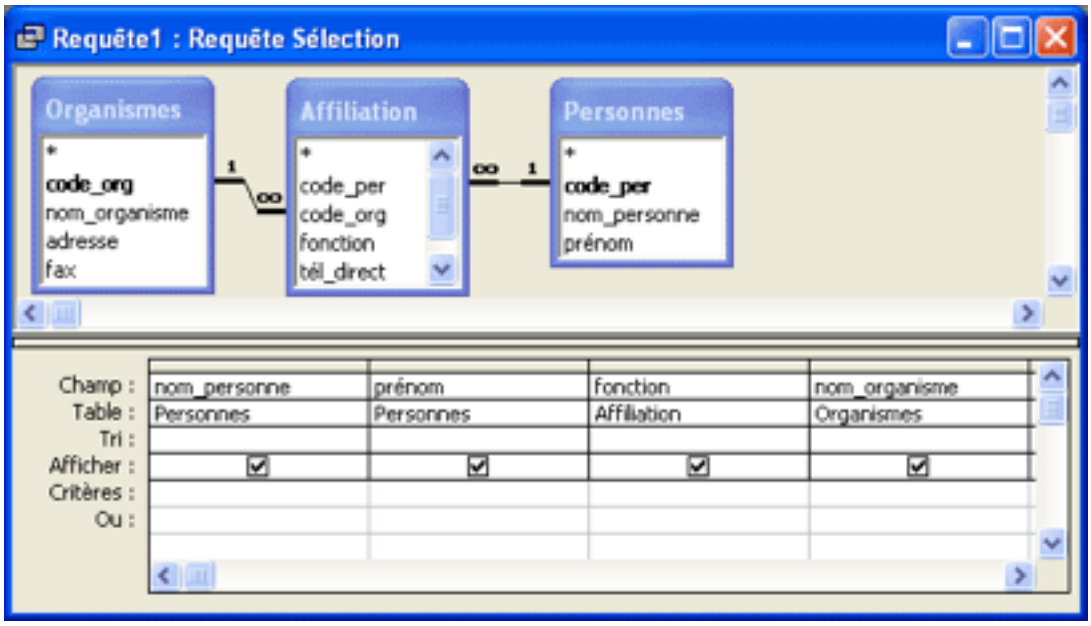
La requête multi-table

Dans une BD relationnelle, les informations sont généralement dispersées dans plusieurs tables (une dizaine couramment, voire plus) liées par un nombre similaire de relations, ce qui fait qu'il est impossible d'avoir une vue globale du contenu de la base. Une requête multi-table permet de rassembler dans une même table les informations désirées, et d'obtenir au premier coup d'oeil une idée de ce contenu.

Revenons à la table "Personnes" que nous avons utilisée au début de ce chapitre. Une personne pouvant travailler pour plusieurs organismes, et un organisme pouvant employer les services de plusieurs personnes, la table "Personnes" doit être séparée en trois tables (dont une table de jonction), liées par des relations. Le schéma relationnel correspondant apparaît sur la figure ci-dessous.

Mais cette séparation en trois tables fragmente les données, et nous empêche de voir simplement qui travaille pour qui. Si nous nous plaçons dans la table "Personnes", nous voyons aussi (grâce à la sous-table) les données de la table "Affiliation", mais pas celles de la table "Organismes". Si nous nous plaçons dans la table "Organismes", nous voyons aussi (grâce à la sous-table) les données de la table "Affiliation", mais pas celles de la table "Personnes". La solution consiste à rassembler pour examen, dans une même table, les données que nous voulons examiner simultanément. Bref, il faut que nous exécutions une requête de sélection simple multi-table.

Dans la fenêtre "Base de données", l'objet "Requêtes" étant sélectionné, nous cliquons sur "Créer une requête en mode Création". Dans la boite de dialogue "Afficher la table", nous sélectionnons les trois tables nécessaires (l'une après l'autre, ou simultanément grâce à la touche CTRL), et nousconstruisons la requête représentée ci-dessous.



Nous obtenons ainsi une vue claire du contenu de la base, vue que nous n'avons absolument pas lorsque nous examinons les trois tables de départ.

Attention ! Si nous effectuons une sélection sur les colonnes de deux tables qui ne sont pas liées par une relation, le logiciel associe chaque ligne de la première table à toutes les lignes de la seconde (cela s'appelle faire leur produit vectoriel). Le résultat est généralement sans intérêt et, si les deux tables sont conséquentes, l'opération risque d'être fort longue.

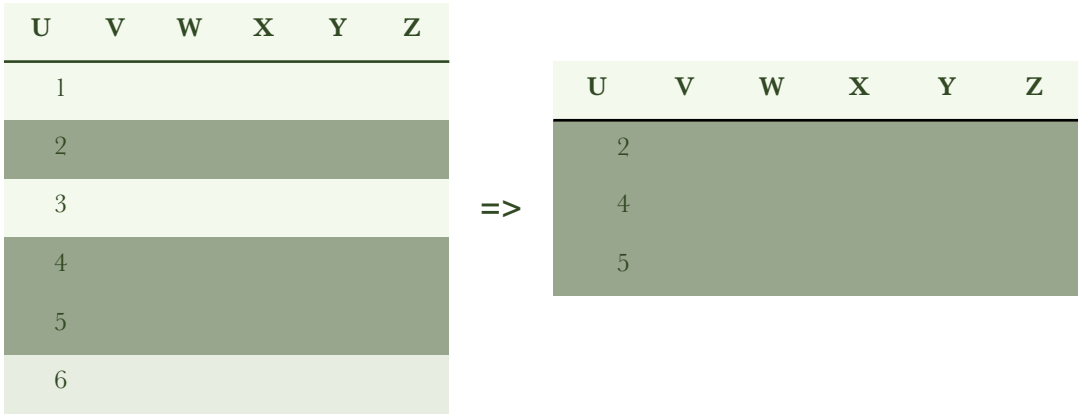
Dans le même ordre d'idée, il ne faut jamais introduire dans la fenêtre de création de requête une table dont la présence n'est pas nécessaire. Le résultat de la requête risque d'être tout à fait aberrant.

Requête de sélection

Stocker sans cesse des informations dans une base de données, et en assurer la maintenance, n'est pas une fin en soi. Il faut pouvoir retrouver, chaque fois que cela est nécessaire, les informations pertinentes dont on a besoin. La requête de sélection a été créée dans ce but. Elle joue, dans les BD, un rôle très important.

La requête de sélection

Dans le cas le plus simple, la requête sélection s'applique à une seule table dont elle conserve toutes les colonnes. Contrairement à la sélection simple (ou projection) qui permet d'extraire d'une table certaines colonnes nommément désignées, la sélection permet d'extraire d'une table les lignes répondant à certains critères, comme le montre la figure ci-dessous. L'ensemble des critères d'une requête de sélection est parfois appelé filtre (par analogie avec le filtre manuel), et l'expression filtrer une table à l'aide d'une requête est assez courante.



La sélection représente l'outil courant de recherche de l'information dans les bases de données. D'une manière générale, la sélection :

- s'applique soit à une seule table, soit à plusieurs tables liées par des relations ;
- permet de sélectionner les lignes par application d'un ou plusieurs critères portant sur un ou plusieurs champs ;
- permet de choisir les colonnes que l'on veut conserver (comme la sélection simple) ;

- peut enregistrer son résultat sous forme d'une table ;
- peut créer une nouvelle colonne ;
- peut être paramétrée.

Tout ce que nous avons exposé au chapitre précédent sur la sélection simple s'applique a fortiori à la sélection en général : choix des colonnes, requête multi table, création de table, création de champ, tri, requêtes emboîtées.

La formulation d'une requête de sélection met en jeu des critères liés par des opérateurs logiques. Sa réalisation pratique pose des problèmes de syntaxe, qui sont propres au SGBD utilisé. A titre d'exemple, recherchons les clients du représentant Dupont, ou de son collègue Durand, qui ont passé une commande de plus de 1000 le mois dernier. Dans une des tables de notre BD se trouve une colonne "Représentant", et il faut que nous exprimions le fait que nous recherchons les enregistrements qui possèdent le nom Durand, ce qui soulève un problème de syntaxe (le nom "Durand" doit être mis entre guillemets). Ensuite, il faut que nous exprimions le fait que c'est "Durand" OU "Dupont", ce qui met en jeu l'opérateur logique OU. Dans une table nommée "Commandes" existe un champ "Coût total", et il faut que nous exprimions le fait que ce coût est supérieur à 1000, ce qui met en jeu l'opérateur de comparaison "supérieur à".

La syntaxe

La syntaxe varie avec le type de données du champ sur lequel porte le critère :

- une donnée de type texte doit être écrite entre guillemets ("..."), et précédée de l'opérateur "Comme". Cet opérateur peut être omis si aucun opérateur de comparaison n'est présent ;
- les nombres sont écrits tels quels ;
- la date et / ou l'heure doivent être placées entre dièses (exemple : #01/01/2003#). Dans certains SGBD, le dièse est remplacé par l'apostrophe ;
- un booléen doit être déclaré vrai ou faux.

La valeur Null (case vide, pas de données) possède une syntaxe particulière. Pour détecter les enregistrements dont un champ particulier est vide, il faut écrire : “Est Null”. Dans le cas contraire, il faut écrire : “Est Pas Null”.

La casse n'a pas d'importance, le SGBD corrigeant de lui-même.

Les caractères génériques

Pour exprimer le fait que nous recherchons les enregistrements qui possèdent la chaîne de caractères "truc" dans un champ donné, nous écrivons, conformément aux indications du paragraphe précédent : comme “truc”.

L'application d'un critère à un champ de type texte recèle un piège particulier. Quand nous exprimons ce critère comme ci-dessus, nous ne sélectionnons que les enregistrements possédant exactement la chaîne "truc" dans le champ considéré. Si le champ contient "trucage", ou "le truc", l'enregistrement correspondant est ignoré.

Il nous faut donc pouvoir préciser comment la chaîne recherchée se présente dans le champ : occupe-t-elle tout le champ, est-elle précédée ou suivie d'autres caractères, et (éventuellement) quel est leur nombre. Pour ce faire, nous utilisons des caractères génériques, c'est à dire des caractères qui peuvent remplacer un ou plusieurs autres caractères quels qu'ils soient. Le caractère générique le plus fréquemment utilisé est l'astérisque, qui remplace un nombre quelconque de caractères. Ainsi : comme “*truc” permet de sélectionner tout enregistrement dont le champ considéré contient une chaîne de caractères se terminant par "truc", telle que "truc" et "le truc" par exemple. Par contre, "trucage" sera ignoré.

De même : comme “truc*” permet de sélectionner tout enregistrement dont le champ considéré contient une chaîne de caractères commençant par "truc", telle que "truc" et "trucage". Par contre, "le truc" sera ignoré.

Enfin : comme “*truc*" permet de sélectionner tout enregistrement dont le champ considéré contient la chaîne "truc", tel que "truc", "trucage" et "le truc".

L'astérisque peut être placée n'importe où, et non pas seulement en début ou en fin de chaîne. Ainsi, une requête exprimée ainsi : comme “/tutoriel/*html" retrouvera toutes les pages HTML dédiées.

Le second caractère générique (par fréquence d'usage) est le point d'interrogation, qui remplace un caractère quelconque et un seul. Ainsi, le critère : comme “c?d" retrouvera par exemple cad, ced, cid, cod, mais pas cd car le point d'interrogation implique la présence d'un caractère.

Si l'on veut rechercher l'astérisque ou le point d'interrogation dans un champ, il faut placer ces caractères entre crochets. Par exemple, la recherche du point d'interrogation (placé n'importe où dans un champ) s'écrit : comme "*[?]*"

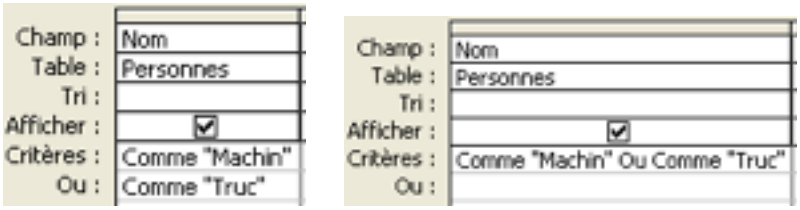
Attention ! Les caractères génériques * et ? ne s'appliquent qu'à l'interrogation des champs de type texte. Dans un champ de type Date/Heure, une expression telle que #**/**/2002# est considérée comme invalide. Notons de plus que, dans les versions récentes de la plupart des SGBD, l'astérisque est remplacée par le pourcent (%) et le point d'interrogation par le caractère de soulignement (_).

Les opérateurs logiques

Une requête un peu élaborée fait appel à plusieurs critères s'appliquant soit à un même champ, soit à des champs distincts. Ces critères sont liés par des opérateurs logiques, dont les plus utilisés sont ET, OU et PAS. L'utilisation de parenthèses permet de définir l'ordre dans lequel s'appliquent les opérateurs. Les personnes familiarisées avec la recherche documentaire connaissent bien cette façon de procéder, qui provient directement de la théorie des ensembles.

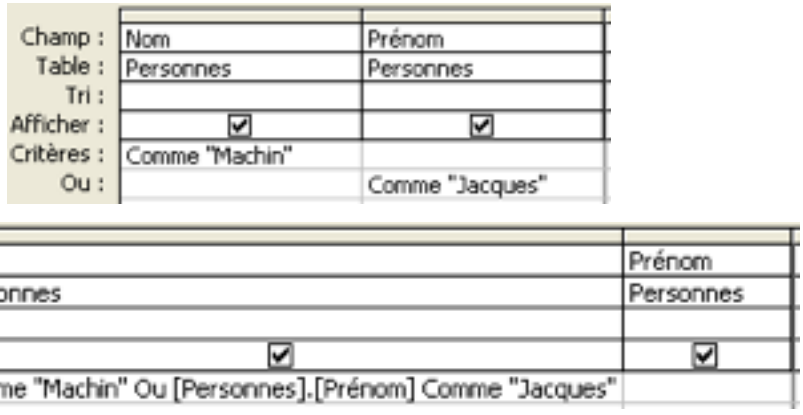
Dans Access, les opérateurs logiques Et et OU peuvent être écrits explicitement, ou être représentés graphiquement dans la grille de l'interface graphique de création d'une requête. L'opérateur PAS doit être écrit explicitement.

Pour rechercher, dans le champ "Nom" d'une table intitulée "Personnes", les individus s'appelant Truc ou Machin, nous avons le choix entre les deux solutions que nous avons représentées ci-dessous (en détourant une partie de la grille de définition de la requête) :



Nous voyons que l'opérateur OU peut être écrit explicitement (à droite), ou traduit graphiquement (à gauche). Le résultat de la requête est, bien entendu, le même dans les deux cas.

L'opérateur logique peut porter sur deux champs distincts. La traduction graphique de l'opérateur OU est alors plus simple que son écriture explicite, comme le montrent les figures ci-dessous. On notera que le OU s'obtient en se décalant d'une ligne... sinon c'est l'opérateur ET qui fonctionne !



L'opérateur ET peut lui aussi être traduit graphiquement ou écrit explicitement, comme le montrent les figures ci-dessous. La requête recherche les noms commençant par la lettre "m" et finissant par la lettre "n", et retrouve par exemple "Machin".

Champ :	Nom
Table :	Personnes
Tri :	
Afficher :	☑
Critères :	Comme "m*" Et Comme "*n"
Ou :	

Champ :	Nom	Nom
Table :	Personnes	Personnes
Tri :		
Afficher :	☑	☐
Critères :	Comme "m*"	Comme "*n"
Ou :		

Notons que la requête aurait pu être formulée plus simplement : Comme "m*n".

L'opérateur logique peut impliquer deux champs distincts, comme le montre l'exemple ci-dessous. La requête recherche les enregistrements relatifs à une personne nommée Pierre Machin.

Champ :	Nom	Prénom
Table :	Personnes	Personnes
Tri :		
Afficher :	☑	☑
Critères :	Comme "Machin"	Comme "Pierre"
Ou :		

Champ :	Nom	Prénom
Table :	Personnes	Personnes
Tri :		
Afficher :	☑	☑
Critères :	Comme "machin" Et [Personnes].[Prénom] Comme "pierre"	
Ou :		

Conclusion : dans la grille de création d'une requête, le déplacement horizontal correspond à l'opérateur ET, et le déplacement vertical correspond à l'opérateur OU. L'opérateur PAS est sans représentation graphique.

Dans une requête complexe, l'application des opérateurs s'effectue ligne par ligne, comme le montre l'exemple ci-dessous.

Champ :	Nom	Prénom	Date naissance
Table :	Personnes	Personnes	Personnes
Tri :			
Afficher :	☑	☑	☑
Critères :	Comme "machin"	Comme "pierre"	
Ou :		Comme "jacques"	#30/08/1975#

La requête fonctionne selon l'expression ensembliste suivante :

```
((Personnes.Nom Comme "machin") ET (Person-
nes.Prénom Comme "pierre"))

                        OU

((Personnes.Prénom Comme "jacques") ET (Person-
nes.[Date naissance]=#8/30/1975#))
```

soit que :

• l'expression qui se trouve sur la ligne "Critères :" est évaluée (elle réalise un ET) ;

• l'expression qui se trouve sur la ligne "Ou :" est évaluée (elle réalise aussi un ET) ;

• un OU est ensuite effectué entre les résultats des deux expressions précédentes.

Les opérateurs de comparaison

Les opérateurs de comparaison arithmétiques :

```
= (égal), < (inférieur), <= (inférieur ou égal),
> (supérieur), >= (supérieur ou égal), <> (dif-
férent)
```

s'appliquent aux données numériques et monétaires, mais aussi aux dates et aux chaînes de caractères. Pour ces dernières, on notera que :

• le signe égal est équivalent à l'opérateur "Comme" ;

• le signe différent est équivalent à l'opérateur "Pas Comme".

Pour préciser un intervalle, on peut utiliser l'expression :

```
Entre ... Et ...
```

qui fonctionne avec les types de données texte, date/heure et numérique/monétaire.

Les fonctions

Pour exprimer des critères, on peut utiliser des fonctions, mais ces dernières sont spécifiques à la fois du SGBD et du type de données du champ considéré. Nous consacrerons une annexe aux fonctions, et nous nous contenterons ici de citer quelques exemples (dont nous avons vérifié qu'ils fonctionnaient effectivement).

Pour les champs en mode texte :

- NbCar([Nom])="4" retrouve les noms de 4 caractères ;
- Droite([Nom];2)="se" retrouve les noms se terminant par "se" ;
- Gauche([Nom];2)="du" retrouve les noms commençant par "du" ;
- ExtracChaîne([Nom];2;3)="ach" retrouve le nom "Machin", lequel contient la chaîne de 3 caractères "ach" en commençant au deuxième caractère.

Pour les dates et les heures :

- PartDate("aaaa";[Date_commande])=2000 retrouve les commandes de l'année 2000. Cette fonction opère aussi avec "j" pour le jour, "m" pour le mois, et "t" pour le trimestre ;
- DiffDate("j";[Date_commande];[Date_livraison])>100 retrouve les produits qui ont été livrés plus de 100 jours après avoir été commandés. Cette fonction opère aussi avec "m" pour le mois, et avec "aaaa" pour l'année ;
- Jour([Date_commande])=12 retrouve les commandes effectuées le 12 (des mois présents dans la table) ;
- Mois([Date_commande])=6 retrouve les commandes du mois de juin ;
- Année([Date_commande])=2000 retrouve les commandes de l'année 2000 ;
- AjDate("j";-10;[Date_livraison]) fournit une date antérieure de 10 jours à la date de livraison.

Attention ! La francisation des fonctions (date/heure) n'a pas toujours été effectuée par l'éditeur avec tout le sérieux nécessaire, et l'utilisateur ne doit pas être surpris s'il se heurte à des disfonctionnements. Ainsi la fonction :

```
JourSem(#date#)
```

qui donne le numéro du jour d'une date donnée, marche à l'américaine : le jour numéro 1 est le dimanche, et non le lundi comme c'est le cas en Europe. Par contre, la fonction :

```
WeekdayName(n° du jour)
```

qui donne le nom du jour connaissant son numéro, fonctionne à l'européenne : le jour n° 1 est bien le lundi. Il en résulte que l'expression obtenue en emboîtant les deux fonctions précédentes :

```
WeekdayName(JourSem(#date#))
```

donne un résultat faux (le lundi à la place du dimanche, etc.). De même les fonctions qui, dans leurs arguments, acceptent le jour ("j"), le mois ("m"), le trimestre ("t") et l'année ("aaaa"), n'acceptent pas la semaine contrairement à ce qui se passe dans la version anglophone d'Access.

Pour les champs de type numérique ou monétaire, on trouve :

- les fonctions arithmétiques habituelles (somme, différence, produit, quotient) ;
- des fonctions mathématiques et statistiques ;
- des fonctions telles que Abs() (valeur absolue), Arrond() (partie entière), Ent() (partie entière, arrondie inférieurement pour les nombres négatifs), Aléat() (nombre aléatoire compris entre 0 et 1) et Sgn() (signe d'un nombre).

A la valeur particulière Null correspond la fonction :

```
EstNull([Nom d'un champ])
```

Elle retourne la valeur -1 si le champ est vide, et 0 (zéro) dans le cas contraire.

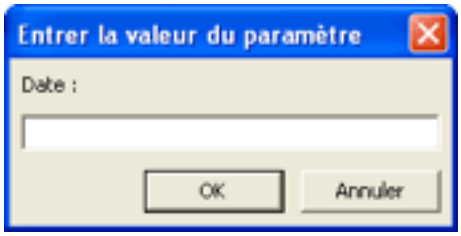
Attention ! Notez bien que, lorsqu'une fonction possède plusieurs arguments, le caractère séparateur est le point-virgule, alors que c'est la virgule dans les versions anglophones. C'est un détail de syntaxe ridicule, mais il est à l'origine de bien des mauvaises surprises.

La requête de sélection paramétrée

Soit une table contenant diverses informations, dont une date, comme le montre l'exemple ci-dessous.

Imaginons que nous ayons régulièrement besoin des informations relatives à un jour donné. Nous pouvons, bien sûr, créer chaque fois une requête nouvelle, mais il est plus commode d'écrire une seule fois la requête et de paramétrer la valeur de la date. Dans la grille de création de la requête, la valeur du paramètre date est remplacée par un message écrit entre crochets :

Si nous lançons la requête, la boite de dialogue suivante s'affiche :



Nous saisissons la date dans le format utilisé par la table (jj/mm/aaaa), et nous validons. Le SGBD affiche les lignes relatives à la date indiquée :

La notion de jointure

Dans une base de données relationnelle, les informations sont réparties sur un grand nombre de tables. Il est donc fréquent qu'une requête porte sur deux tables (ou plus), liées par une (ou plusieurs) relation(s). La notion de jointure précise comment fonctionnent cette (ou ces) relation(s) lors de l'exécution de la requête.

Pour bâtir un exemple, nous faisons appel aux deux tables "Personnes" et "Communes" dont nous nous sommes déjà servis précédemment. Nous remplissons ces deux tables comme le montre la figure ci-dessous. Précisons que le champ "Commune" de la table "Personnes" n'a pas été rendu obligatoire (le Null est autorisé), si bien qu'il peut arriver qu'une commune ne soit pas attribuée à une personne, comme c'est le cas pour Jeanne Dupont.

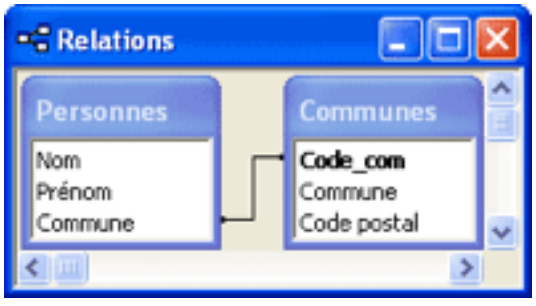
Nom	Prénom	Commune
Truc	Jean	Grenoble
Chose	Pierre	Nancy
Machin	Noémie	Uriage
Dupont	Jeanne	

Table "Personnes"

Commune	Code_Postal
Grenoble	38000
Grenoble	38001
Nancy	54000
Uriage	38410

Table "Communes"

Les deux tables sont liées par une relation, assurée via un code masqué (sinon, comment savoir quel est le code postal correspondant à Jean). Cette relation apparaît dans la fenêtre "Relations", comme le montre la figure ci-dessous.



La relation

Si nous ajoutons les deux tables précitées à la fenêtre de création d'une requête, nous constatons que la relation qui les lie est toujours présente.

Dans la fenêtre de création d'une requête, nous pouvons supprimer cette relation. La procédure est identique à celle pratiquée dans la fenêtre "Relations" : nous effectuons un clic droit sur la relation, et nous choisissons "Supprimer". Nous fermons la fenêtre de création de la requête, et nous enregistrons cette dernière.

Si nous ouvrons la fenêtre "Relations", nous constatons que la relation qui lie les deux tables existe toujours. Cette relation est en quelque sorte une propriété des deux tables.

La suppression que nous avons effectuée est liée à une requête particulière. Elle n'a d'effet que lors de l'exécution de la requête. Ce n'est pas une propriété des deux tables, mais une propriété de la requête.

En conclusion, les opérations que nous effectuons sur les relations (création, suppression, modification des propriétés) ont un effet :

- permanent lorsqu'elles sont effectuées dans la fenêtre "Relations" ;
- éphémère lorsqu'elles sont effectuées dans la fenêtre de création d'une requête particulière.

Remarque : même s'il n'existe pas de relation entre deux tables, le SGBD Access en crée une automatiquement lorsque vous ajoutez ces deux tables à la fenêtre de création d'une requête, à condition que ces tables aient chacune un champ du même nom et du même type de données, et qu'un des deux champs possède une clé primaire.

Le produit vectoriel

Nous rouvrons la requête précédente en mode "Modification". Nous vérifions qu'aucune relation n'apparaît entre les deux tables. Dans la grille, nous introduisons

les champs "Nom" et "Prénom" de la première table, et les champs "Commune" et "Code postal" de la seconde. La feuille de données résultante contient 20 lignes ! Que s'est-il passé ?

Le SGBD a associé chaque ligne de la première table (il y en a 4) à chaque ligne de la seconde (il y en a 5). On dit qu'il a effectué le produit vectoriel des deux tables. L'absence de relation fait que le SGBD ne sait pas comment il doit associer les lignes des deux tables ; de ce fait, il réalise toutes les combinaisons possibles.

Il faut faire attention au fait que le produit vectoriel peut nous conduire à créer des tables gigantesques : le produit de deux tables contenant chacune 1.000 enregistrements est une table possédant 1 million de lignes !

En pratique, on n'utilise pas le produit vectoriel, sauf dans des cas très rares, comme par exemple pour réunir dans une seule table des comptages isolés. Ces derniers se présentent en effet sous forme de tables à une seule ligne, et l'on peut en faire le produit vectoriel sans risque, car le résultat est alors une table à une seule ligne.

La jointure interne

Dans la fenêtre de création de la requête, nous rétablissons la relation entre les deux tables. Cette fois, la feuille de données résultante ne contient plus que 3 lignes, comme le montre la figure ci-dessous.

Nom	Prénom	Commune	Code_Postal
Truc	Jean	Grenoble	38000
Chose	Pierre	Nancy	54000
Machin	Noémie	Uriage	38410

Nous constatons que ne figurent dans la table résultante que les enregistrements qui sont présents dans les deux tables. La personne Dupont Jeanne, dont la commune n'est pas précisée, est absente du résultat. Les villes Grenoble (38001), auxquelles ne

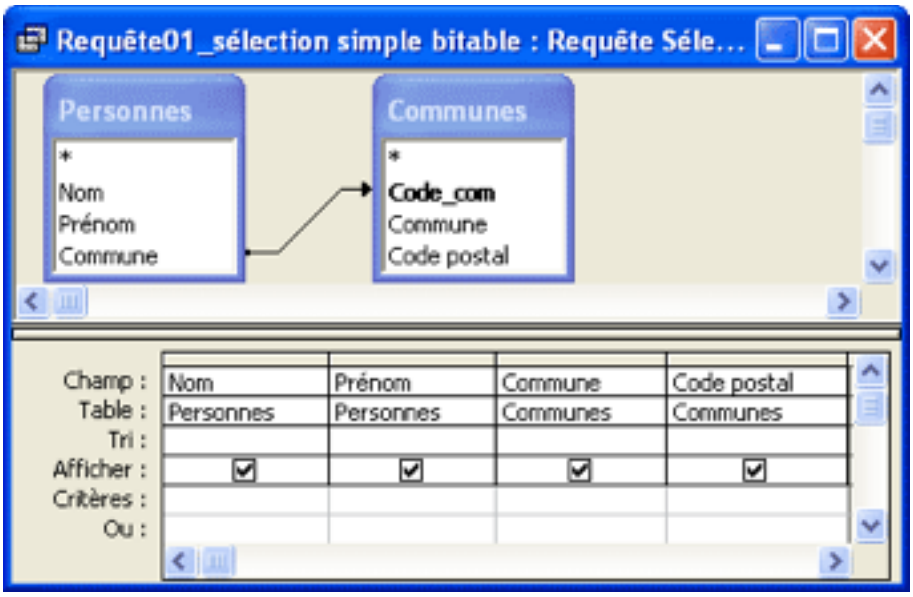
correspond aucune personne, sont également absente. Le SGBD a traité la relation entre les deux tables comme une jointure interne.

Effectuons un clic droit sur la relation, et sélectionnons "Propriétés de la jointure". La fenêtre du même nom s'affiche ; elle se présente comme le montre la figure ci-dessous. Bien que le terme ne soit pas présent, l'option 1 de la fenêtre correspond effectivement à la jointure interne.

Remarque : dans la requête précédente, le champ "Commune" est issu de la table "Communes". S'il provenait de la table "Personnes", le résultat s'afficherait de la même façon. C'est seulement en exportant la table que l'on peut s'apercevoir que dans le second cas, le champ contient un code au lieu d'un nom de commune.

La jointure gauche

La fenêtre "Propriétés de la jointure", représentée ci-dessus, nous fournit deux autres options. Nous activons le bouton 2 et nous validons par "OK". La requête se présente maintenant comme le montre la figure ci-dessous. Nous avons affaire à une jointure gauche. Pour le signaler, la liaison prend la forme d'une flèche... dirigée vers la droite.



Si nous basculons en mode feuille de données, nous obtenons le résultat suivant :

Nom	Prénom	Commune	Code_Postal
Truc	Jean	Grenoble	38000
Chose	Pierre	Nancy	54000
Machin	Noémie	Uriage	38410
Dupont	Jeanne		

Cette fois, le SGBD a conservé tous les enregistrements de la table "Personnes", et il leur a associé les enregistrements disponibles dans la table "Communes". Comme nous n'avons pas précisé de critère de sélection, tous ces enregistrements ont été conservés.

La jointure droite

Dans la fenêtre "Propriétés de la jointure", nous activons le bouton 3 et nous validons par "OK". Nous avons maintenant affaire à une jointure droite. Pour le signaler, la liaison prend la forme d'une flèche... dirigée vers la gauche.

Si nous basculons en mode feuille de données, nous obtenons le résultat suivant :

Nom	Prénom	Commune	Code_Postal
Truc	Jean	Grenoble	38000
		Grenoble	38001
Chose	Pierre	Nancy	54000
Machin	Noémie	Uriage	38410

Cette fois, le SGBD a conservé tous les enregistrements de la table "Communes", et il leur a associé les enregistrements disponibles dans la table "Personnes". Comme nous n'avons pas précisé de critère de sélection, tous ces enregistrements ont été conservés.

Conclusion : le résultat d'une requête multi-table dépend du type de jointure choisi. Par défaut, c'est la jointure interne qui s'applique.

La requête de non correspondance

La requête de non correspondance constitue une application importante de la notion de jointure. Elle met en jeu deux tables ayant en commun un champ possédant le même type de données, et doté des mêmes propriétés (mais pas forcément du même nom). La requête de non-correspondance ne conserve un enregistrement de la première table que si le contenu du champ n'est pas présent dans la seconde table. Les deux tables n'ont pas besoin d'être liées au préalable par une relation, cette dernière sera créée en même temps que la requête.

Pour construire un exemple simple, nous créons deux tables à un seul champ, contenant des prénoms, et intitulées "Prénoms1" et "Prénoms2". Les tables se présentent ainsi :

```
Prénoms1   Prénoms2
Paul       Henri
Jean       Patrick
Marie      Paul
Henri
Claude
```

Nous recherchons les prénoms de la première table qui sont absents de la seconde. Pour ce faire, nous devons passer en revue tous les prénoms de la première table, et regarder s'ils sont ou non dans la seconde. Pour créer la requête correspondante, nous songeons donc à utiliser une jointure gauche.

Pour bien comprendre ce qui se passe, nous pouvons décomposer en deux temps le fonctionnement de la requête. D'abord, le SGBD sélectionne tous les prénoms de la première table, et leur associe les prénoms de la seconde table quand il sont identiques. Le résultat de cette première étape peut être représenté ainsi :

Prénoms1	Prénoms2
Paul	Paul
Jean	
Marie	
Henri	Henri
Claude	

Il faut maintenant que le SGBD applique un critère de sélection, pour ne conserver que les lignes dont la deuxième colonne est vide. Nous plaçons donc le critère "Est Null" dans la colonne relative au champ "Prénom" de la seconde table.

En définitive, nous obtenons la requête représentée sur la figure ci-dessous (remarquez la flèche qui traduit la jointure gauche). Nous avons supprimé l'affichage de la seconde colonne, car il conduirait à créer une colonne vide dans la feuille de données résultante.

Si nous basculons en mode feuille de données, nous obtenons le résultat suivant :

Prénoms

Jean

Marie

Claude

Il est indispensable, dans un requête de non correspondance, de ne pas se tromper sur le type de jointure à utiliser. Ainsi, si nous choisissons la jointure interne (par défaut), le SGBD ne sélectionne que les prénoms qui sont simultanément présents dans les deux tables. Le résultat de cette étape intermédiaire est représenté ci-dessous :

Prénoms1	Prénoms2
Paul	Paul
Henri	Henri

Lorsque nous appliquons le critère "Est Null" à la deuxième colonne, le SGBD ne conserve que les lignes pour lesquelles la deuxième colonne est vide. Comme il n'y en a pas, la feuille de données résultante ne contient aucun enregistrement -- ce que l'expérience confirme.

Si nous utilisons la jointure droite, le SGBD sélectionne tous les prénoms de la seconde table, et seulement ceux de la première table qui se trouvent dans la seconde. Cette étape intermédiaire peut être représentée ainsi :

Prénoms1	Prénoms2
Paul	Paul
Henri	Henri
	Patrcick

Puis le SGBD élimine les lignes pour lesquelles la seconde colonne est vide. Comme il n'y en a pas, la feuille de données résultante ne contient aucun enregistrement -- ce que l'expérience confirme.

Si le raisonnement relatif à la requête de non correspondance vous paraît ardu, ne vous inquiétez pas : vous n'êtes pas le seul. C'est la raison pour laquelle Microsoft a créé un assistant pour ce type de recherche. Vous le trouverez dans la fenêtre "Base de données", l'objet "Requêtes" étant sélectionné. Vous cliquez sur l'icône  "Nouveau", et vous sélectionnez "Assistant Requête de non-correspondance".

La requête de correspondance

La requête de correspondance est en quelque sorte le complément de la précédente. Elle met en jeu deux tables ayant en commun un champ possédant le même type de données, et doté des mêmes propriétés (mais pas forcément du même nom). Elle ne conserve un enregistrement de la première table que si le contenu du champ est présent dans la seconde table. La requête de correspondance constitue elle aussi une application courante de la notion de jointure. Comme précédemment, les deux tables n'ont pas besoin d'être liées au préalable par une relation, cette dernière étant créée en même temps que la requête.

Cette fois, nous recherchons les prénoms de la première table qui sont présents dans la seconde. Pour créer la requête correspondante, il nous faut utiliser une jointure interne. Cela suffit, il n'est pas utile de préciser un critère. Nous obtenons ainsi la requête représentée sur la figure ci-dessous.

Si nous basculons en mode feuille de données, nous obtenons le résultat suivant, complémentaire de celui obtenu avec la requête de non correspondance :

Prénoms
Henri
Paul

Exercez-vous à prévoir ce qui se passe si vous vous trompez de jointure, et vérifiez si l'expérience confirme vos prédictions.

La requête de correspondance étant plus facile à créer que la requête de non-correspondance, l'éditeur n'a pas prévu d'assistant pour aider à la créer. Cependant, vous pouvez utiliser l'assistant précédent, et changer simplement la condition "Est Null" par son contraire "Est Pas Null". C'est inutilement compliqué, mais cela marche.

La requête de regroupement

La requête de regroupement est un important outil d'analyse et de synthèse. Pour cette raison, nous lui consacrons un chapitre entier. Le terme "Requête de regroupement" est le plus courant , mais on rencontre aussi "Requête d'agrégation", qui est synonyme.

Les requêtes de regroupement sont très utilisées dans l'analyse des résultats comptables et financiers. Comme nous le verrons dans le chapitre suivant, elles sont aussi utilisées pour le comptage et l'élimination des doublons.
La notion de regroupement déborde le cadre des seules requêtes. Nous la retrouverons également dans les états et les formulaires.

Comme pour les autres chapitres de ce tutoriel (ou tutorial, ou cours en ligne), nous utilisons le SGBD Access comme support pratique.

La notion de regroupement

Créons une table intitulée "Résultats", contenant le chiffre d'affaires journalier d'une entreprise possédant trois agences. Le champ "Date" est du type "Date/Heure", le champ "Agence" du type texte (10 caractères), et le champ CA (Chiffre d'Affaires) du type monétaire (format euro). Introduisons quelques valeurs dans la table, qui prend alors l'aspect représenté ci-dessous.

Date	Agence	CA
15/10/2015	Nord	927.02
15/10/2015	Sud	1098.46
15/10/2015	Est	561.29
16/10/2015	Nord	1385.55
16/10/2015	Est	681.09
16/10/2015	Sud	1401.56

Pour juger les performances de l'entreprise, ces données brutes sont malcommodes. Un décideur a besoin du chiffre d'affaires non seulement au jour le jour, mais aussi à la semaine, au mois et pour l'exercice annuel. Il le veut toutes agences confondues pour juger des performances de l'entreprise. Il le veut aussi agence par agence, pour juger des performances de ces dernières (le responsable de l'agence Est va prendre un savon). Et il ne veut pas être obligé de sortir sa calculette pour regrouper les chiffres qui l'intéressent ; le regroupement doit être effectué par le SGBD.

Pour l'exemple très simple que nous avons choisi, deux regroupements du chiffre d'affaires sont possibles :

- par date, en sommant les CA des trois agences, de manière à obtenir le CA quotidien de l'entreprise. Dans ce cas, la notion d'agence s'efface ;
- par agence, en sommant les CA de chaque agence sur l'ensemble des dates mentionnées dans la table. Dans ce cas, la notion de date s'efface.

Quand peut-on envisager d'effectuer un regroupement dans une table ?

- Quand il existe un champ possédant des doublons. Dans l'exemple ci-dessus, il serait impossible de regrouper par date si chaque valeur de la date n'apparaissait qu'une seule fois. De même, il serait impossible d'envisager le regroupement par agence, si le nom de chaque agence n'apparaissait pas de manière répétée.

Quelle opération peut-on envisager quand on effectue un regroupement ? La nature de cette opération dépend du type des données à regrouper :

- des données numériques ou monétaires se prêtent à des opérations arithmétiques (somme, moyenne, minimum, maximum), statistiques (variance et écart-type), voire mathématiques. Tout dépend des possibilités offertes par le SGBD ;
- des données de type texte se prêtent au classement et au comptage (la concaténation n'est pas prévue).

Nous voyons tout de suite qu'une requête de regroupement met en jeu le plus souvent deux colonnes :

- une colonne sur laquelle s'effectue le regroupement (elle doit contenir des doublons). On peut effectuer le regroupement sur plusieurs colonnes lorsqu'il existe des doublons s'étendant sur plusieurs colonnes ;
- une colonne sur laquelle s'effectue une opération (somme, ou moyenne, ou etc.).

La mise au point d'une requête de regroupement peut s'avérer délicate, et il faut garder en mémoire les observations suivantes :

- Regroupement. Le SGBD permet d'effectuer le regroupement sur plusieurs colonnes, mais la probabilité pour qu'il existe des doublons (sur plusieurs colonnes) diminue très vite avec le nombre de ces dernières. Dans beaucoup de cas rencontrés en pratique, on effectue le regroupement sur une colonne seulement ;
- Opérations. On peut envisager d'effectuer des opérations sur plusieurs colonnes, si elles s'y prêtent. Dans l'exemple ci-dessus, le CA pourrait être ventilé sur deux colonnes (l'une pour les biens, l'autre sur les services, par exemple), que nous pourrions sommer séparément ;
- Requête multi-table. Une requête de regroupement peut impliquer plusieurs tables liées par des relations, mais il est alors beaucoup plus facile de commettre des erreurs de conception. Il est donc prudent d'utiliser d'abord une requête de sélection pour regrouper dans une même table les données dont on a besoin, avant d'appliquer la requête de regroupement, même si cela risque d'augmenter un peu le temps d'exécution.

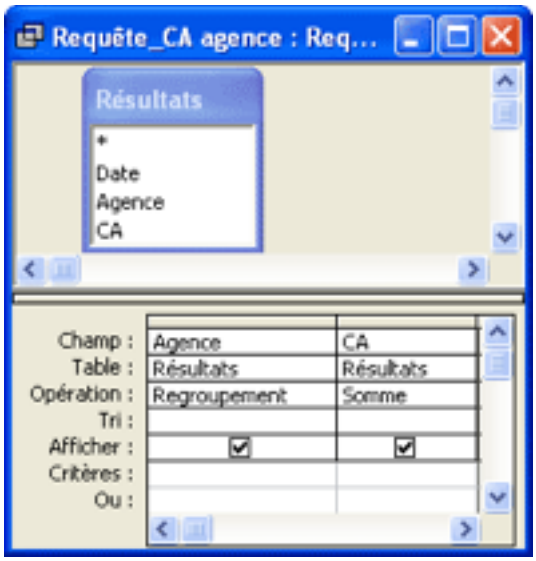
Il résulte de ces considérations qu'une requête de regroupement met généralement en jeu un nombre très restreint de champs. En fait, il est fortement conseillé de commencer la mise au point d'une requête de regroupement sur deux colonnes seulement, et que ces deux colonnes appartiennent à la même table (ou à la même feuille de données).

La création de la requête

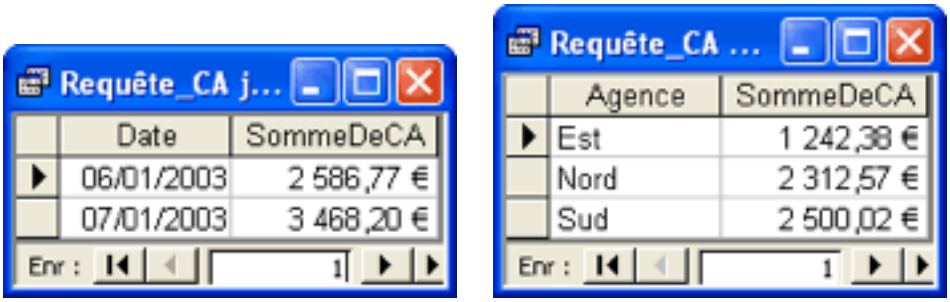
Nous allons créer le premier regroupement envisagé au paragraphe précédent (calcul du CA quotidien de l'entreprise). Pour ne pas nous tromper, nous allons opérer de manière méthodique.

1. Elle consiste à ouvrir la fenêtre de définition d'une requête, et à y introduire la table sur laquelle on veut effectuer l'opération de regroupement (ici “Résultats").
2. Elle consiste à introduire dans la grille le champ sur lequel s'effectue le regroupement. Comme nous cherchons à calculer des CA quotidiens, ce champ ne peut être que la date. Nous introduisons donc le champ "Date" dans la grille de création de la requête.
3. Il faut signifier au SGBD que la requête implique un regroupement sur le champ "Date". Pour ce faire, nous cliquons sur l'icône Σ "Totaux". Une nouvelle ligne, baptisée "Opération :", apparaît dans la grille de définition de la requête, entre "Table :" et "Tri :" (figures ci-dessous). La valeur par défaut, pour le champ "Date", est justement "Regroupement" (si cette valeur n'apparaît pas, nous cliquons sur la ligne et nous sélectionnons "Regroupement" dans la liste déroulante). Ainsi, le regroupement sera effectué sur la date.
4. Il faut maintenant introduire le champ sur lequel s'effectue l'opération liée au regroupement. Dans le présent exemple, l'opération consiste à sommer les CA de chaque agence. Nous introduisons donc le champ "CA" dans la grille.
5. Il faut indiquer au SGBD à quelle opération il doit procéder sur le champ "CA". Nous cliquons sur la ligne "Opération :", nous utilisons la liste déroulante pour remplacer "Regroupement" (qui s'est inscrit par défaut) par “Somme".

La requête se présente ainsi comme le montre la figure ci-dessous à gauche.



Quand nous basculons en mode feuille de données, nous obtenons le résultat représenté sur la figure ci-dessous (à gauche). Nous vérifions que le SGBD a bien calculé, pour chaque date, la somme des chiffres d'affaires des trois agences.



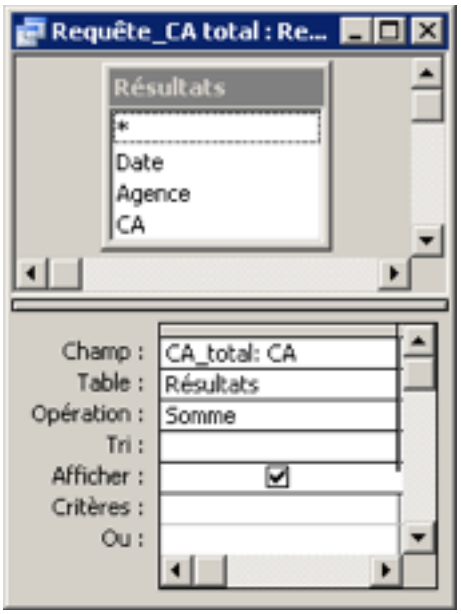
De la même façon, nous pouvons regrouper le chiffre d'affaires par agence. La requête et son résultat sont représentés dans les deux figures ci-dessus (à droite). Cette fois le SGBD a calculé, pour chaque agence, la somme des chiffres d'affaires quotidiens, pour les deux dates figurant dans la table.

Remarque 1 : l'icône Σ fonctionne comme un commutateur. Si nous cliquons dessus alors que la ligne "Opération :" est présente, celle-ci disparaît, et vice versa.

Remarque 2 : le nom du champ ("SommeDeCA") a été attribué par le SGBD, mais on peut le changer à volonté dans la grille de création de la requête. Pour l'appeler "CA agence", par exemple, il faut remplacer "CA" sur la ligne "Champ :" par "CA agence: CA".

Les opérations sur les colonnes

Dans le paragraphe précédent, nous avons créé des requêtes impliquant un regroupement sur une colonne donnée, ce qui nous a permis de calculer le chiffre d'affaires par date (pour chacune des trois agences) ou par agence (pour chacune des deux dates). Mais nous pouvons également avoir besoin du chiffre d'affaire total, pour toutes les agences et toutes les dates de la table. Pour ce faire, nous créons de nouveau une requête de regroupement, mais sans déclarer sur quel champ nous regroupons, comme le montre la figure ci-dessous à gauche. Vous noterez que nous avons utilisé la syntaxe qui nous permet d'imposer le nom du champ (CA_total).



Le résultat figure ci-dessus à droite ; vous pourrez vérifier qu'il est bien exact. Bien entendu, vous pouvez utiliser d'autres fonctions que la somme dans ces opérations effectuées verticalement. L'étude de ces fonctions fait l'objet du paragraphe suivant.

Ainsi vous pouvez effectuer des opérations sur les colonnes d'une table, comme vous le feriez dans un tableur. La différence avec un tableur -- outre la syntaxe -- provient du fait que le résultat ne peut pas être enregistré dans la table. Il apparaît systématiquement dans une nouvelle "feuille de données" volatile, à moins que vous ne demandiez que la requête crée une nouvelle table.

Les fonctions

La fonction "Somme" (qui ne s'applique qu'aux données numériques et monétaires) n'est pas la seule qui puisse être utilisée lors du regroupement. Dans Access, on trouve également les fonctions suivantes :

- Moyenne : calcule la moyenne. S'applique uniquement aux données numériques ou monétaires ;
- Min : retient seulement la valeur la plus basse. S'applique aussi au texte (classement alphabétique) ;

- Max : retient seulement la valeur la plus haute. S'applique aussi au texte (classement alphabétique) ;
- ÉcartType : calcule l'écart type. S'applique uniquement aux données numériques ou monétaires ;
- Var : calcule la variance. S'applique uniquement aux données numériques ou monétaires ;
- Premier : retient la première valeur rencontrée (en parcourant la table du haut en bas). S'applique aussi au texte ;
- Dernier : retient la dernière valeur rencontrée (en parcourant la table du haut en bas). S'applique aussi au texte ;
- Compte : compte le nombre de doublons dans le regroupement. Nous dédierons un chapitre particulier à l'étude de cette fonction, qui s'applique à tous les types de données.

La liste des fonctions utilisables lors du regroupement varie d'un SGBD à l'autre. La somme, la moyenne et le comptage sont présents dans tous les SGBD.

Si nous essayons de faire opérer une fonction sur un type de données incompatible, le SGBD Access affiche le message d'erreur suivant : "Type de données incompatible dans l'expression du critère". Et qu'en termes galants ces choses-là sont dites ! Évidemment, il ne s'agit pas d'un critère, mais d'une fonction.

On trouvera ci-dessous des exemples illustrant l'application de ces différentes fonctions, à l'exception de l'écart type et de la variance, qui n'auraient guère de sens vu le petit nombre de données (trois par regroupement) contenues dans la table "Résultats" qui nous sert d'exemple.

Date	MoyenneDeCA
15/10/2015	862.26
16/10/2015	1156.07

Agence	MoyenneDeCA
Est	621.19
Nord	1156.29
Sud	1250.01

Date	MinDeCA
15/10/2015	561.29
16/10/2015	681.09

Agence	MinDeCA
Est	561.29
Nord	927.02
Sud	1098.46

Date	MaxDeCA
15/10/2015	1098.46
16/10/2015	1401.56

Agence	MaxDeCA
Est	681.09
Nord	1385.55
Sud	1401.56

Le filtrage d'une requête de regroupement

Une requête de regroupement peut être filtrée avant et /ou filtrée après le regroupement. Il faut indiquer au SGBD dans quel cas on entend se trouver.

Le filtrage après regroupement. Ce filtrage ne pose pas de problème particulier, puisque les champs sur lesquels nous pouvons opérer sont présents dans la grille de définition de la requête, et que cette dernière comporte une ligne "Critères :". Nous y inscrivons les critères de filtrage, en respectant les règles de syntaxe propre au SGBD, comme nous avons appris à le faire au chapitre précédent.

Dans la colonne "CA", par exemple, le critère : “>2000” limitera l'affichage aux regroupements conduisant à des chiffres d'affaires supérieurs à 2.000.

Dans la colonne "Agence", le critère : comme “Nord" limitera le calcul du chiffre d'affaire cumulé par agence à la seule agence Nord.

Dans la colonne "Date", le critère : <#30/09/2015# limitera le calcul du chiffre d'affaire cumulé par date aux jours précédant le 30 septembre 2015.

Le filtrage avant regroupement. Pour filtrer une requête avant regroupement, nous procédons ainsi :

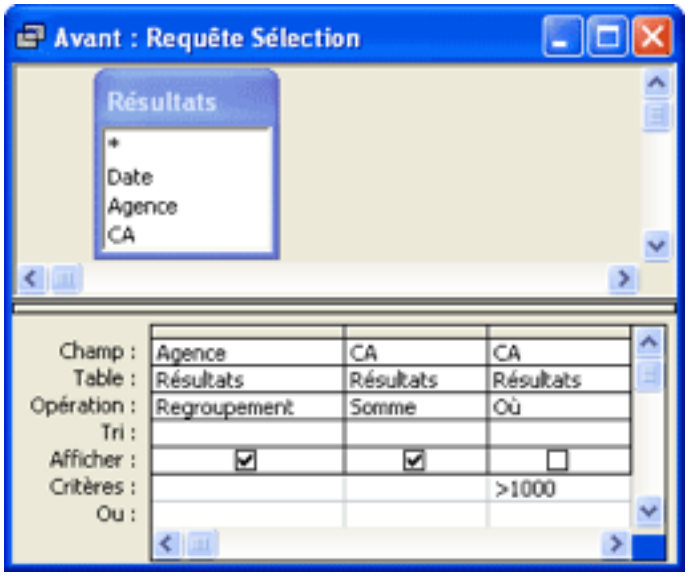
• nous introduisons le champ dans la grille ;

- nous cliquons sur la ligne "Opération :", où "Regroupement" s'est inscrit par défaut ;
- nous sélectionnons "Où" tout en bas de la liste déroulante. Nous constatons alors que la coche disparaît de la case qui se trouve sur la ligne "Afficher :". Attention ! si nous oublions d'inscrire "où" sur la ligne "Opération :", la requête ne fonctionnera pas ;
- nous introduisons le(s) critère(s) de filtrage sur la ligne "Critères :".

Avant de filtrer une requête de regroupement, il faut bien réfléchir au résultat que l'on veut obtenir. Car un même critère, appliqué au même champ, ne donnera pas le même résultat suivant qu'il opère avant ou après le regroupement. Pour le montrer, nous avons créé deux exemples dans lesquels le filtre consiste à ne conserver un CA que s'il est supérieur à 1000 euros, et où le regroupement s'effectue sur l'agence.

1. (figures de gauche). Nous appliquons le filtre aux CA regroupés par agence, c'est à dire que nous ne les affichons que s'ils dépassent 1000 euros ;
2. (figures de droite). Nous appliquons le filtre aux CA quotidiens, et nous ne les prenons en compte que s'ils dépassent 1000 euros.

Les requêtes et leurs résultats sont représentés sur les figures ci-dessous.



Si le critère porte sur le champ de regroupement, l'application du filtre avant ou après le regroupement donne le même résultat. Faites l'expérience !

5

IMPRIMER L'INFORMATION RECHERCHÉE

Dans une base de données, l'objet état est utilisé pour mettre en forme les données destinées à être imprimées. Lorsque la matérialisation de données issue d'un SGBD présente un caractère répétitif, et plus encore lorsqu'il est envisagé d'automatiser cette opération, le recours à un état constitue la meilleure solution. A ce titre, l'état constitue donc le troisième objet des SGBD par ordre d'importance décroissante, après les tables et les requêtes.

Mais la création d'un état qui présente correctement les données imprimées est souvent une opération longue et quelque peu fastidieuse. C'est pourquoi l'état n'est pas toujours considéré comme indispensable, et il existe deux façons de s'en passer.

Première solution. Nous pouvons imprimer directement une table ou une feuille de données, à condition de limiter considérablement nos ambitions en matière de présentation. Nous sommes maîtres de la largeur des colonnes (ne pas lésiner sur ce point, sinon l'information risque d'être tronquée), de la couleur de fond de cellule, de la taille et du type de la police, et c'est à peu près tout. Le SGBD pagine, affiche la date et le nom de la table (ou de la feuille de données), et met en page à sa façon -- c'est à dire qu'il commence en haut et à gauche, tout simplement. Imprimer directement une table ou une feuille de données est une solution de dépannage, mais ce n'est pas vraiment le moyen de réaliser un document bien présenté.

Deuxième solution. Une table créée dans Access et dans d'autres SGBD fonctionnant sous le système d'exploitation Windows peut facilement être exportée vers un tableur, et en particulier vers Excel qui est le plus utilisé. Dans un tableur, la mise en page avant impression est facile et intuitive, et nous disposons là d'un bon moyen pour obtenir un imprimé correctement présenté. L'exportation vers Excel des résultats d'une requête constitue une technique de plus en plus utilisée, non seulement pour mettre en forme des données avant impression, mais aussi pour profiter des diverses fonctions qu'offre le tableur. L'opération est particulièrement facile si nous nous trouvons dans Access : la table à analyser ou à imprimer étant sélectionnée, nous cliquons dans le menu sur "Outils", puis sur "Liaisons Office"et enfin sur "Analyse avec Microsoft Excel". Le tableur s'ouvre, et la table y est aussitôt exportée. De plus, un fichier au format Excel est enregistré sur le bureau.

Un état est pratiquement toujours construit sur le résultat d'une requête, et ce pour les raisons suivantes :

- les bases de données contiennent souvent des quantités considérables d'information, et il n'est pas question de tout imprimer. Il faut donc commencer par sélectionner l'information particulière que l'on veut reproduire avant d'imprimer ;
- dans une BD relationnelle, l'information est répartie dans des tables multiples, et il faut la rassembler avant de l'imprimer. On peut, cependant, introduire dans un même état des champs provenant de plusieurs tables, à condition que ces dernières soient liées par des relations ;
- on peut désirer que l'information imprimée se présente dans un certain ordre. Il faut donc opérer un tri plus ou moins complexe avant d'imprimer. Ceci dit, on peut également demander un tri complexe (jusqu'à quatre niveaux) lors de la création de l'état.

Bien qu'il ne soit pas question, en général, d'imprimer la totalité du contenu d'une BD, un état s'étale souvent sur plusieurs pages. C'est le SGBD qui se charge de gérer les sauts de page (si l'utilisateur ne donne pas d'instructions particulières à ce sujet), d'imprimer l'en-tête et le pied de chaque page. L'en-tête de la première page, et la fin de la dernière page, sont généralement différents de ceux des autres pages.

Comme pour les autres chapitres de ce tutoriel (ou tutorial, ou cours en ligne), nous utiliserons le SGBD Access comme support pratique. Nous prévenons cependant le lecteur que la mise en forme d'un état dans Access est une tâche quelque peu pénible, parce que l'outil mis à notre disposition par le SGBD est malcommode. Pour une impression occasionnelle, nous recommandons donc d'utiliser plutôt Excel.

Signalons au passage que, si vous utilisez Access 2002 comme SGBD, Windows XP comme système d'exploitation, et si vous n'êtes pas administrateur de votre machine, un bug peut faire capoter la création de l'état en fin d'opération. Si vous vous trouvez dans ce cas, prévenez l'administrateur de votre service informatique, et priez le ciel pour qu'il résolve le problème.

Les préliminaires

La finalité de l'état étant la réalisation d'une sortie imprimée, il est indispensable d'indiquer au SGBD :

- l'imprimante utilisée ;
- la taille de la zone imprimable, c'est à dire celle du papier, diminuée de celle des marges.

Nous rencontrons ici l'une des différences importantes qui existent entre l'état et le formulaire : le premier opère dans une zone imprimable, le second dans une fenêtre de l'écran du moniteur.

L'imprimante. Le SGBD construit l'état en fonction des caractéristiques de l'imprimante par défaut. Si vous utilisez un poste de travail sur lequel aucune imprimante n'a été déclarée, vous allez au-devant de bien des ennuis (lenteur, plantage...). Vérifiez donc ce point avant d'entreprendre la création d'un état.

La zone imprimable. Ensuite, le SGBD tient compte des options que vous avez choisies -- ou, plus généralement, conservées par défaut -- en ce qui concerne la taille du papier (A4, sauf exception) et les marges d'impression. Si vous manquez de

place en largeur -- ce qui est souvent le cas lorsqu'un état est présenté en colonnes -- vous avez intérêt à réduire les marges à gauche et à droite. Mais attention ! vous devez effectuer cette opération avant de commencer à construire votre état. Une fois ce dernier créé, les changements de marge que vous effectuez sont sans effet sur lui.

Dans Access, pour régler les marges, cliquez dans le menu sur "Outils", puis "Options..." : la fenêtre "Options" s'ouvre. Choisissez l'onglet "Général" : les quatre marges sont réglées par défaut à un pouce (2,54 cm). En pratique, 2 cm à gauche et 1 cm à droite suffisent largement. Quelles que soient les valeurs que vous choisissez, elles resteront valables quelle que soit la BD dans laquelle vous travaillez, tant que vous ne les modifierez pas à nouveau.

La sélection des données. Mettez au point la (ou les) requête(s) qui vous permettent de sélectionner les données à imprimer. Si vous avez besoin d'un tri, incorporez-le à ce stade, car les tris que l'on demande au niveau des états ne fonctionnent pas toujours très bien. Enfin, vous n'êtes pas obligé de créer une table, un état pouvant être construit directement sur le résultat d'une requête.

La présentation des données. Il existe deux modes de présentation d'un état :

- la présentation tabulaire : les données sont disposées dans des colonnes verticales, comme dans une table. Le nom des colonnes (l'étiquette) figure une seule fois par page. Cette présentation, qui convient particulièrement bien aux données numériques, économise de la place ;
- la présentation verticale : les données relatives à chaque champ sont imprimées les unes en dessous des autres, le nom du champ étant rappelé à chaque fois. Cette présentation est la seule possible lorsque le contenu de certains champs est volumineux. Elle peut faire l'objet d'un regroupement sur une colonne donnée, si cette dernière contient de nombreux doublons. Elle présente l'inconvénient de requérir beaucoup de place.

Il est bon de réfléchir à la présentation avant de commencer à créer l'état.

Les barres d'outils. Il en existe deux qui sont spécifiques des états. Elles s'appellent respectivement "Créer un état" et "Mise en forme (Formulaire/État)", et certaines de leurs icônes nous seront indispensables. Pour faire apparaître ces deux barres, nous cliquons sur "Affichage" dans le menu, puis sur "Barres d'outils", et nous cochons le nom de la barre désirée. Par la même occasion, nous demandons l'affichage de la boîte à outils.

La création de l'état

Ouvrons une BD contenant au moins une table pourvue de données et, dans la fenêtre "Base de données", sélectionnons l'objet "États". Deux options nous sont présentées :

- Créer un état en mode Création
- Créer un état à l'aide de l'Assistant

Les manuels, en général, conseillent d'utiliser l'assistant en toutes circonstances, car créer un état à partir de rien est assez fastidieux. Mais, dans le cas où l'on choisit la présentation tabulaire, et où il y a de nombreux champs, l'assistant conduit à un mauvais résultat, qu'il est ensuite difficile de corriger complètement. Dans ce cas, il faut confier à l'assistant le soin de créer une partie des champs seulement (en particulier ceux sur lesquels on envisage un regroupement), et utiliser ensuite la fonction "Modifier" pour introduire les autres champs au fur et à mesure que la mise en forme de l'état progresse. Obtenir un bon résultat lors de l'impression demande une bonne dose de patience.

Il existe aussi une icône  "Nouveau", qui redonne le mode création et l'assistant, mais propose de plus des fonctions dont l'intérêt est plutôt mince :

- État instantané : Colonnes. Cette fonction crée un état en présentation verticale dès que nous lui fournissons le nom de la table ou de la requête. Nous ne sommes pas maître des paramètres, et le résultat est sommaire ;
- État instantané : Tableau. Cette fonction opère comme la précédente, mais en présentation tabulaire ;

- Assistant graphique. Cette fonction permet de créer des graphiques, mais leur qualité est très mauvaise. Pour créer un graphique correct, il faut exporter les données vers un tableur ;
- Assistant étiquette. Comme son nom l'indique, cette fonction permet d'imprimer des étiquettes.

Lançons l'assistant, choisissons une table ("Personnes", avec les champs "Nom" et "Prénom"), sélectionnons ses deux champs, ignorons le regroupement et le tri, adoptons la disposition tabulaire, l'orientation portrait et le style "Informel" (valeurs par défaut), donnons un titre à l'état ("Liste des personnes"), et cliquons sur "Terminer". Nous obtenons un état qu'il est souhaitable d'améliorer.

La structure de l'état

L'état que nous venons de créer apparaît dans la fenêtre "Base de données" (l'objet "État" étant sélectionné). Nous le sélectionnons, puis nous cliquons sur l'icône  "Modifier". Une fenêtre s'ouvre, qui nous permet de modifier l'état.

Procédons d'abord à une petite vérification. Notre imprimante utilise du papier A4 de 21 cm de large, et nous avons choisi des marges gauche et droite importantes (5 cm) afin de limiter la taille de la figure ci-dessous. Nous vérifions que la largeur utile de l'état (la partie claire et quadrillée) mesure effectivement 11 cm, comme le montre la figure. Même si nous modifions par la suite les marges dans la fenêtre "Options", celles de notre nouvel état ne varieront pas.

Nous observons que l'état est divisé en cinq zones :

- En-tête d'état. Cette zone est imprimée en haut de la première page uniquement ;
- En-tête de page. Cette zone est imprimée en haut de toutes les pages (à condition qu'elle contienne de l'information) ;
- Détail. Ce terme désigne la zone qui s'étend entre l'en-tête et le pied de page. Cette zone permet d'imprimer le contenu de la table (ou de la feuille de données) sous-jacente ;
- Pied de page. Cette zone est imprimée en bas de chaque page ;
- Pied d'état. Cette zone s'imprime uniquement sur la dernière page, après le détail et avant le pied de page. Par défaut, l'assistant lui attribue une hauteur nulle.

De manière quelque peu schématique, on peut dire que les diverses zones sont utilisées de la manière suivante :

- la première zone (en-tête d'état) contient le titre, de telle sorte qu'il n'apparaisse qu'une seule fois, et qu'il se trouve en tête de l'état ;
- dans une présentation tabulaire, les noms des champs (on les appelle les "étiquettes") sont placés dans la seconde zone (en-tête de page), de manière à être reproduits en haut de chaque page. Dans une présentation verticale, les étiquettes sont regroupées avec les champs correspondants dans la zone "Détail", et l'assistant ne prévoit pas d'utiliser l'en-tête de page ;

- les champs (on les appelle "zones de texte", ou parfois "contrôles" comme dans les formulaires) sont placés dans la troisième zone (détail) ; leurs valeurs sont imprimées ligne après ligne, tant qu'il reste de la place entre l'en-tête et le pied de chaque page. Le SGBD passe ensuite à la page suivante, et le processus recommence ;
- dans la quatrième zone (pied de page), le système pagine et inscrit la date. Ces informations apparaissent donc en bas de chaque page.
- la cinquième zone (pied d'état) contient les résultats de calculs éventuels, ou un message de fin d'état.

Dans la fenêtre graphique de modification de l'état, les zones de texte et les étiquettes apparaissent comme des rectangles contenant du texte.

On distingue trois catégories de zone de texte :

- la zone de texte dépendante. Son contenu provient soit d'un champ de la table ou de la requête sous-jacente, soit d'une instruction SQL. Les zones de texte dépendantes se trouvent généralement dans la zone "Détail" de l'état ;
- la zone de texte indépendante. Son contenu ne provient pas d'un objet de la BD. On l'utilise pour afficher un texte informatif (exemple : le titre de l'état), une image (exemple : le logo de l'entreprise), ou des éléments de décoration, principalement dans l'en-tête de l'état ;
- la zone de texte calculée. Elle contient le résultat d'un calcul (exemples : somme, moyenne, fonctions statistiques, date, page, etc.). Ce résultat est remis à jour chaque fois que les données utilisées dans le calcul sont modifiées. Les zones de texte calculées se trouvent généralement dans les pieds de page ou dans le pied de l'état.

La mise en forme de l'état

Un état étant ouvert en mode "Modification", les outils qui nous permettent de procéder à sa mise en forme sont dispersés à cinq endroits différents. Nous pouvons en effet utiliser :

- les icônes des barres d'outils "Créer un état" et "Mise en forme (Formulaire/État)". Nous avons déjà indiqué au paragraphe 2 comment les faire apparaître ;
- la feuille de propriétés que possède chaque objet de l'état. Pour la faire apparaître, nous sélectionnons l'objet (ex : le titre), nous cliquons sur l'icône  "Propriétés", puis nous sélectionnons l'onglet "Format" ;
- la fenêtre graphique elle-même, dans laquelle il est possible de modifier l'état en tirant ou en glissant-déplaçant à l'aide de la souris ;
- le menu, par ses rubriques "Format" et "Insertion" ;
- la boite à outils. Pour la faire apparaître, nous cliquons sur l'icône  "Boîte à outils". Cette boîte est commune aux états et aux formulaires, mais elle est nettement moins utilisée pour les premiers que pour les seconds.

Les barres d'outils. Nous noterons d'abord que la barre de mise en forme est active lorsqu'une zone de texte est sélectionnée. Cette barre contient les outils usuels du traitement de texte (propriété de la police, alignement du texte, couleur de fond et de premier plan, bordures). Elle contient en outre (tout à fait à gauche) la liste des objets de l'état, liste dont nous avons déjà parlé au paragraphe 4 précédent.

Lorsque nous modifions la présentation d'un état, il nous faut régulièrement vérifier le résultat obtenu, car ce qui s'affiche est souvent différent de ce qui s'observe en mode "Modifier". Pour cela, nous cliquons sur l'icône  "Aperçu" de la barre d'outils "Créer un état".

Cette barre contient également deux autres icônes fort utiles. L'icône  "Liste des champs" permet de rajouter de nouveaux champs à l'état, par simple glisser-déplacer (de la liste vers l'état). L'icône  "Propriétés" permet d'afficher la feuille de propriétés de l'objet sélectionné dans l'état.

Ces feuilles de propriétés nous sont fort utiles, car dans leur onglet "Format" nous pouvons :

- régler la position des quatre coins du rectangle contenant une zone de texte. L'origine des coordonnées correspond au coin situé en haut et à gauche de la zone dans laquelle se trouve la zone de texte ;
- régler les marges et l'interligne (à l'intérieur de la zone de texte) ;
- colorer le fond ou le rendre transparent ;
- régler la bordure (couleur, épaisseur) et l'apparence (3D, ombré, etc.) ;
- fixer tous les attributs du texte, comme dans un traitement de texte.

Si nous avons affaire à une zone de l'état, la liste des attributs est évidemment plus réduite.

La fenêtre graphique nous permet d'effectuer de multiples opérations à l'aide de la souris :

- modifier la taille des différentes zones qui constituent l'état, en tirant sur la barre qui les sépare, lorsque le curseur prend la forme ✢ d'une croix. Appliquée à la zone "Détail", cette opération a pour effet de modifier l'interligne à l'impression (dans la présentation tabulaire) ;
- déplacer les zones de texte et les étiquettes, soit séparément, soit en les regroupant par sélection multiple (touche shift enfoncée). Lorsque le curseur prend la forme d'une main ouverte, toutes les zones sélectionnées sont simultanément déplacées. Lorsque le curseur prend la forme d'un index pointé, seule la zone concernée est déplacée. Attention ! ne positionnez pas une zone de texte contre la limite de la zone imprimable, sinon l'imprimante débitera une page blanche pour chaque page imprimée ;
- modifier la taille d'une étiquette ou d'une zone de texte, en tirant sur les sommets ou les milieux des côtés du rectangle correspondant ;
- modifier la largeur de la zone imprimable. Réduisez-la si vous le désirez, mais ne l'élargissez pas au-delà de sa taille initiale, même si vous avez réduit les marges entre temps. Sinon, une page blanche accompagnera chaque page imprimée. Un bon conseil : ne touchez pas à la largeur de la zone imprimable.

Le positionnement à l'aide de la souris ne permet pas d'obtenir un résultat précis. Il est donc recommandé de perfectionner le positionnement en réglant les coordonnées dans les feuilles de propriétés.

Attention ! Si nous plaçons une zone de texte dépendante dans l'en-tête de l'état, seule la première valeur du champ correspondant sera imprimée (parce que l'en-tête est unique). Ceci peut être mis à profit pour imprimer une information qui ne varie pas dans la table sous-jacente (une date par exemple).

Le menu. Les fonctions de la rubrique "Format" deviennent actives lorsqu'une étiquette ou une zone de texte est sélectionnée. Elles nous permettent :

- d'aligner (plusieurs étiquettes ou zones de texte) ;
- de régler la taille (au contenu, par exemple) ;
- de fixer l'espacement horizontal (plus grand ou plus petit) entre étiquettes et/ou zones de texte ;
- de placer en premier plan ou en arrière plan (une image par rapport à une zone de texte par exemple).

Les fonctions de la rubrique "Insertion" nous permettent :

- de numéroter les pages et d'imprimer la date et l'heure (si nous avons créé l'état sans l'aide de l'assistant) ;
- d'insérer des graphiques, des images ou des objets divers, que nous pouvons ensuite placer où nous voulons dans l'état.

La boite à outils. Seuls sont réellement utiles :

- l'outil "Sélection", qui est présent par défaut lorsque la boite à outils n'est pas affichée ;
- l'outil "Étiquette", qui permet en fait de créer une zone de texte indépendante ;
- l'outil "Zone de texte". Lors de l'affichage ou de l'impression de l'état, le SGBD nous demandera de saisir le contenu de cette zone ;
- l'outil "Image", qui permet d'insérer une image dans l'état ;

- l'outil "Saut de page". Dans un état en présentation verticale, l'insertion d'un saut de page en bas de la zone Détail permet de n'afficher qu'un seul enregistrement par page (gare à la consommation de papier !) ;
- l'outil "Sous-formulaire". Les sous-formulaires seront étudiés au chapitre 24 suivant.

Les autres outils -- en particulier ceux qui permettent d'insérer des boutons, des boutons radio, des cases à cocher, des listes, etc. -- n'ont pas d'utilité pour mettre en forme un état qui est, par vocation, destiné à être imprimé.

6

SAISIE D'INFORMATION

Les formulaires simples

Les formulaires sur deux tables

Les formulaires simples

Le formulaire est souvent considéré comme le troisième objet des bases de données, par ordre d'importance décroissante, après la table et la requête. En fait, son importance réelle dépend de la manière dont on utilise le SGBD.

Le formulaire est avant tout un outil de saisie d'information au clavier. A ce titre, il entre en concurrence avec :

• l'écriture directe dans les tables ;

• l'importation des données.

La facilité d'écriture directe dans les tables peut varier de très pratique à parfaitement impraticable suivant les cas. Un formulaire peut rendre la saisie de certaines informations plus facile, principalement dans les SGBD qui, au contraire d'Access, n'affichent pas les sous-feuilles de données. Enfin, les formulaires permettent l'ajout de boutons, menus, etc. qui donnent à l'application un aspect très "fini". Les SSII

(sociétés de service en informatique) soignent donc les formulaires pour donner la meilleure impression possible à leur client -- surtout si ce dernier n'est pas capable de juger sur autre chose que la présentation.

Certaines bases de données sont principalement alimentées en données par importation des données : le formulaire ne sert alors plus à rien. C'est, par exemple, le cas des magasins à grande surface, qui alimentent leur BD directement et en temps réel depuis les caisses enregistreuses. C'est aussi le cas des sites web qui déversent quotidiennement leur fichier journal dans la BD qui sert au suivi du site et à la mesure d'audience. C'est encore le cas de tous ceux qui font de l'acquisition de données via des capteurs couplés à des ordinateurs, etc.

Accessoirement, le formulaire sert aussi d'outil de visualisation, c'est à dire de consultation du contenu de la base à l'écran. On reproche parfois au formulaire de montrer les enregistrements un par un, alors qu'une table en montre un grand nombre à la fois, mais on peut concevoir le formulaire de telle sorte que sa présentation soit très proche de celle d'une table.

Cette discussion peut en fait se résumer ainsi :

- lorsque la BD est utilisée par des personnes très diverses sans expérience particulière en matière de SGBD, ou lorsque l'accès aux tables est interdit aux utilisateurs, ou lorsque la saisie directe dans les tables est malaisée, les formulaires constituent un passage obligé ;
- lorsque la BD est utilisée par un petit groupe de professionnels formés à l'usage des SGBD, ou lorsque les données sont importées au lieu d'être saisies, les formulaires constituent un simple habillage de la BD et n'ont guère d'utilité.

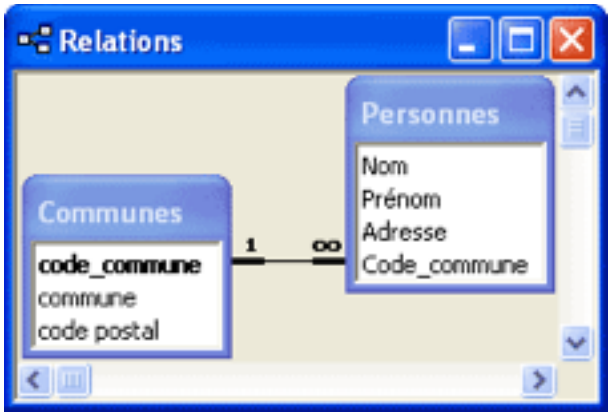
Comme pour l'ensemble de ce tutoriel (encore appelé "cours en ligne" ou tutorial), nous utiliserons le SGBD Access comme support pratique.

La création d'un formulaire simple

Un formulaire est avant tout un outil permettant de saisir au clavier des données qui sont immédiatement introduites dans une ou plusieurs tables. Le formulaire est

donc lié à une ou à plusieurs tables, et il hérite de leurs propriétés : types de données, propriétés des champs, listes de choix et protection contre les doublons via un index. A l'inverse, les propriétés du formulaire ne rejaillissent pas sur les tables sous-jacentes. Il arrive enfin que l'on puisse attribuer à un champ de formulaire une propriété qui modifie ou contredit celle du champ correspondant de la table.

Pour étudier les formulaires, nous utiliserons une table (nommée "Personnes") dotée d'une liste de choix (nommée "Communes"), comme le montre la figure ci-dessous. Nous faisons en sorte (comme expliqué au chapitre 3) que le nom de la commune, et non son code, s'affiche dans le champ "Code_commune" de la table "Personnes".



Pour travailler sérieusement, nous créons un index sans doublon sur les champs Nom+Prénom dans "Personnes", et sur les champs "commune+code postal dans "Communes". Nous interdisons de plus le Null et la chaîne vide dans tous les champs sauf "Adresse". Le champ "code_commune" de la table "Communes" est du type NuméroAuto.

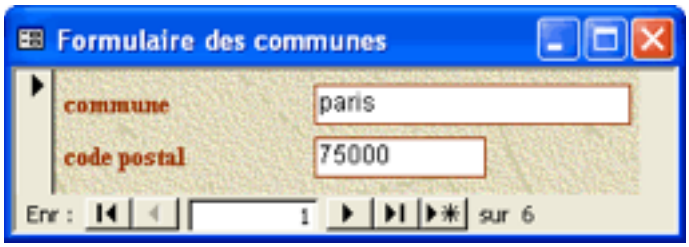
Créons, pour commencer, un formulaire simple qui nous permette de saisir les noms des communes et les codes postaux correspondants. Dans la fenêtre "Base de données", nous sélectionnons (colonne de gauche) l'objet "Formulaires". Nous double-cliquons sur "Créer un formulaire à l'aide de l'Assistant" et nous répondons aux demandes de ce dernier :

- d'abord, nous sélectionnons la table "Communes". Les champs disponibles s'affichent (code_commune, commune, code postal), et nous sélectionnons les deux

derniers. En effet le code, qui sert à faire fonctionner la mécanique relationnelle, est implémenté par le système lui-même (type de données NuméroAuto) ;

- ensuite, nous choisissons une disposition -- "Colonne simple" par exemple ;
- puis nous choisissons l'un des styles proposés ;
- enfin nous nommons le formulaire "Formulaire des communes", et nous cliquons sur "Terminer".

Voici comment se présente le formulaire, dont le nom apparaît désormais dans la fenêtre "Base de données" :



Le formulaire est constitué d'étiquettes (les noms des champs dans la table) et de contrôles correspondant aux champs de la table. Lorsqu'ils sont directement dérivés de la table, les contrôles sont appelés "contrôles dépendants", ou encore "champs". La table à partir de laquelle est construit le formulaire est la table sous-jacente.

Nous constatons d'abord que nous pouvons examiner le contenu de la table "Communes" enregistrement par enregistrement, le formulaire jouant alors son rôle d'outil de visualisation. Certes, l'aspect est plus joli que celui de la table, mais cette dernière présente l'avantage de nous donner une vue globale des informations. Si nous avions choisi la disposition "Tabulaire", nous aurions obtenu une présentation par lignes plus proche de la table sous-jacente. La disposition "Feuille de données", quant à elle, fournit une présentation pratiquement identique à celle de la table sous-jacente.

Nous constatons ensuite que nous pouvons nous placer sur la première ligne vide de la table sous-jacente, et saisir des données (noms et prénoms des personnes). Ces données sont automatiquement introduites dans la table sous-jacente, comme nous pouvons le constater en fermant le formulaire et en ouvrant la table. Le formulaire

joue alors son rôle d'outil de saisie des données. Mais les problèmes de synchronisation que nous avons déjà rencontrés demeurent : si nous laissons la table sous-jacente ouverte, nous constatons que les données saisies dans le formulaire n'y apparaissent pas. On peut fermer, puis rouvrir la table sous-jacente, pour que les nouvelles données s'y trouvent inscrites, mais il est plus simple de rendre la table active, de cliquer dans le menu sur "Enregistrements", puis sur "Afficher tous les enregistrements". La table se complète immédiatement.

Notons que le formulaire nous présente les informations dans l'ordre où elles se trouvent dans la table sous-jacente. Pour faire en sorte que les informations apparaissent triées, nous disposons de plusieurs méthodes :

- nous pouvons effectuer une requête de sélection simple sur la table (en sélectionnant tous les champs, en triant d'abord sur le nom et ensuite sur le prénom), puis créer le formulaire à partir de cette requête. Toutes les saisies que nous effectuons dans ce nouveau formulaire sont automatiquement transmises à la table "Personnes", au travers (si l'on peut dire) de la requête de tri ;
- si un tri simple nous convient, nous plaçons le curseur dans le champ désiré du formulaire, puis nous cliquons sur l'icône  du tri ;
- pour effectuer un tri multiple, nous cliquons dans le menu sur "Enregistrements", puis sur "Filtrer", puis sur "Filtre/tri avancé..." et nous intervenons dans la grille qui s'affiche.

La mise en forme du formulaire

Il est fréquent que les administrateurs de BD empêchent les utilisateurs de voir les tables (et par la même occasion, d'effectuer des requêtes -- quelle politique !). Les formulaires constituent alors (avec les états et les menus, que nous étudierons plus loin) la seule interface entre la base et ses utilisateurs. Le SGBD Access met à disposition des concepteurs de BD de multiples outils permettant de rendre cette interface aussi soignée que possible. Avec la mise en forme du formulaire, nous entrons quelque peu dans le domaine de l'informatique “cosmétique".

Sélectionnons le formulaire que nous venons de créer, et cliquons sur l'icône  "Modifier". Le formulaire s'ouvre en mode de création (on devrait dire plutôt en mode de modification, car il est déjà créé). Notons que, à partir de l'option "Affichage" du menu, nous pouvons passer du mode création au mode formulaire, ou au mode feuille de données, la présentation duquel est très proche de celle de la table sous-jacente. Nous obtenons le même résultat en cliquant sur l'icône située à gauche de la barre d'outils "Création de formulaire", à condition que cette dernière soit affichée.

Comme les tables et les requêtes, le formulaire se présente donc sous deux aspects :

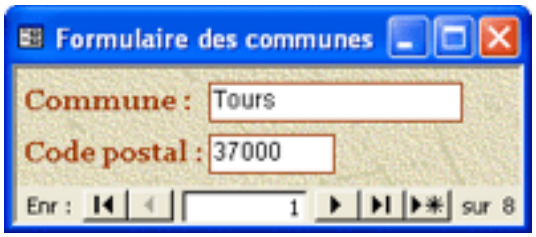
- la structure (titre, étiquettes, contrôles, etc.), que l'on définit en mode création ;
- l'outil de saisie et de visualisation des données, que l'on utilise en mode formulaire.

Le mode création met à notre disposition de multiple outils (entre lesquels il existe une redondance partielle) pour modifier le formulaire que nous avons créé. Nous pourrions même les utiliser pour créer le formulaire de toutes pièces, mais il est plus simple de passer d'abord par l'assistant. Ces outils peuvent être regroupés ainsi :

- une intervention via un clic droit sur une partie spécifique du formulaire, et choix de "Propriétés" dans la liste déroulante. Une boite de dialogue s'ouvre, dans laquelle nous sélectionnons l'onglet "Format". Il nous est alors possible de régler de nombreux détails de présentation ;
- une intervention graphique directe dans la fenêtre. Nous pouvons développer les parties "en-tête" et "pied de page", régler la hauteur et la largeur du formulaire, déplacer les contrôles et les étiquettes, étendre le formulaire sur plusieurs pages (séparées ou en onglet), etc. ;
- une boite à outils qui s'affiche en même temps que le formulaire en mode création. Si cette boite n'apparaît pas, cliquer sur "Affichage" dans le menu, et sélectionner "Boite à outils".

La mise en forme d'un formulaire, avec ses très nombreuses possibilités, fera l'objet d'une annexe (décembre 2002). Nous reviendrons plus loin sur certains usages par-

ticuliers de la boite à outils. Voici ce que devient la figure précédente après un léger lifting :



Le perfectionnement du formulaire

Pour modifier les propriétés d'un formulaire, il faut ouvrir sa feuille de propriétés de la manière suivante :

- sélectionner le formulaire ;
- cliquer sur l'icône "Modifier" ;
- cliquer sur l'icône "Propriétés" ou double-cliquer sur le carré noir qui se trouve en haut et à gauche de la fenêtre du formulaire ;
- sélectionner l'onglet "Données", qui donne accès à diverses propriétés du formulaire.

Dans le formulaire tel qu'il est, nous pouvons non seulement saisir de nouvelles données, mais aussi modifier ou supprimer des données existantes. Cette possibilité peut être fort dangereuse, et nous pouvons la supprimer en basculant la propriété "Modif autorisée" de "Oui" à "Non". En revenant en mode formulaire, nous constatons que désormais, les données enregistrées (par fermeture du formulaire) ne peuvent plus être modifiées. Cette interdiction est très gênante en cas d'erreur de saisie, mais elle ne s'étend pas à la table sous-jacente.

Nous pouvons aller plus loin, et cacher complètement les enregistrements déjà saisis. Les enregistrements nouvellement saisis seront cachés à leur tour dès que nous refermerons le formulaire. Pour ce faire, nous basculons la propriété "Entrée données" de "Oui" à "Non". En revenant en mode formulaire, nous constatons qu'aucune donnée ne s'affiche. Nous pouvons, par contre, saisir de nouvelles données.

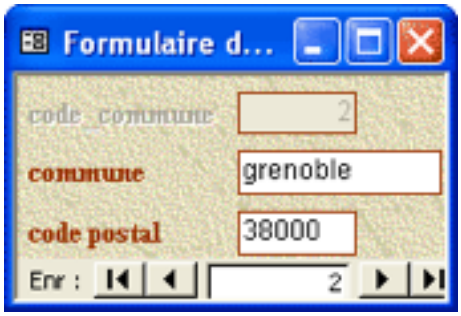
Nous pouvons aussi faire en sorte que le formulaire ne permette pas la modification des données, c'est à dire qu'il serve uniquement de dispositif de visualisation. Pour ce faire, nous basculons la propriété "Ajout autorisé" de "Oui" à "Non", puis nous revenons au mode formulaire. Nous pouvons ainsi donner à un utilisateur la possibilité de consulter la base, mais sans pouvoir en modifier le contenu.

Recréons le formulaire des communes en incluant le champ "code_commune". Nous constatons que nous pouvons introduire le curseur dans ce champ, mais que nous ne pouvons pas le modifier. Le type de données est NumératoAuto, et seul le SGBD peut écrire dans ce champ. Dans le formulaire, cette propriété est héritée de la table sous-jacente. Pour ne pas agacer la personne qui utilise le formulaire, nous pouvons faire en sorte que le curseur ne passe plus dans le champ "code_commune". Plusieurs solutions s'offrent à nous.

Pour modifier les propriétés d'un contrôle ou d'une étiquette, nous pouvons faire apparaître la feuille de données comme ci-dessus, et sélectionner l'objet dans la liste déroulante qui se trouve tout en haut. Nous pouvons aussi sélectionner l'objet en mode création , effectuer un clic droit, et choisir "Propriétés".

Sélectionnons l'onglet "Autres", et basculons la propriété "Arrêt tabulation" de "Oui" à "Non". Quand nous revenons au mode formulaire, nous constatons que le curseur ne peut plus être placé dans le champ "code_commune", bien que la valeur de ce dernier continue à s'afficher.

Nous pouvons aussi sélectionner l'onglet "Donnée", et basculer la propriété "Activé" de "Oui" à "Non" (la propriété "Verouillé" étant sur "Non"). Quand nous revenons au mode formulaire, nous constatons que le contrôle "code_commune" et son étiquette sont grisés, comme lorsqu'une fonction n'est pas disponible dans un menu. De plus, le curseur ne pénètre plus dans le champ. La figure suivante illustre cette situation :



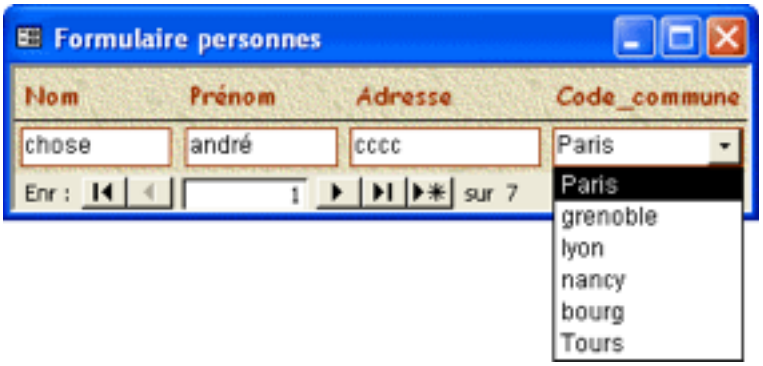
Basculer la propriété "Verrouillé" à "Oui" (la propriété "Activé" étant sur "Oui") a pour conséquence d'empêcher la modification du contenu du champ, sans en modifier l'aspect et sans empêcher le curseur d'y pénétrer. Cela ne nous est d'aucune utilité dans le cas présent, puisque le champ "code_commune" hérite déjà de cette propriété de par son type de données NuméroAuto.

Pour consulter la suite, qui traite des formulaires basés sur deux tables, cliquez sur la flèche droite ci-dessous.

Les formulaires sur deux tables

Le formulaire avec liste de choix

Lorsque la table sous-jacente est dotée d'une liste (laquelle provient le plus souvent d'une table auxiliaire), le formulaire en hérite automatiquement. Pour le montrer, nous créons un formulaire sur la table "Personnes". Après un peu de mise en forme, il se présente comme le montre la figure ci-dessous :



Dans la table "Communes", qui sert de liste, nous avons déclaré nulle la largeur de la colonne "code_commune". Comme pour une table, c'est alors le nom de la commune qui s'affiche à la place du code. Nous devrions donc modifier l'étiquette, en remplaçant "Code_commune" par "Commune", afin que l'utilisateur du formulaire ne soit pas troublé.

Le formulaire avec liste indépendante

Le logiciel Access nous permet de créer une liste indépendante, contenant par exemple les noms et prénom déjà saisis. Pour ce faire, nous ouvrons le formulaire en mode création, puis nous agrandissons la zone d'en-tête pour y placer la liste. Si la boite à outils n'est pas apparente, nous cliquons sur "Affichage", puis "Boite à outils" (nous pouvons aussi cliquer sur l'icône ). Dans cette boite, nous activons l'icône  "Assistants contrôle". Comme son nom l'indique, cette icône active un certain nombre d'assistants relatifs à la création de contrôles.

Nous cliquons ensuite sur l'icône "Zone de liste déroulante", puis à l'endroit choisi pour implanter la liste. La boite de dialogue "Assistant de zone de liste déroulante" s'ouvre. Nous dialoguons ainsi avec l'assistant :

- nous choisissons la troisième option "Rechercher un enregistrement dans mon formulaire basé sur la valeur que j'ai sélectionnée..." ;
- nous sélectionnons ensuite le nom et le prénom ;
- nous réglons graphiquement la largeur des colonnes ;
- nous baptisons l'étiquette "Liste des personnes", et nous cliquons sur terminer ;
- nous ajustons graphiquement la position de la liste (lorqu'une petite main noire apparait) ;
- nous réglons les détails de présentation, puis nous basculons en cliquant sur l'icône "Ouvrir" pour observer le résultat.

Si nous avons pris la précaution de baser le formulaire sur une requête qui trie la table "Table01", la liste indépendante que nous venons de créer est triée par ordre alphabétique. Nous pouvons donc vérifier rapidement si le nom d'une personne donnée a déjà été saisi. De plus, si nous sélectionnons un nom dans cette liste, les informations correspondantes s'affichent immédiatement, comme le montre la figure ci-dessous. Par contre, le fait de changer d'enregistrement n'a pas d'influence sur la liste indépendante.

Une telle liste indépendante rend-elle réellement service ? A l'internaute de juger !

Le formulaire basé sur deux tables

L'assistant permet de créer un formulaire basé sur plusieurs tables. Il suffit de répéter l'opération de saisie des champs pour les tables considérées. Réalisons l'ex-

périence à l'aide des tables "Personnes" et "Communes". La procédure est la suivante :

- nous sélectionnons la table "Personnes", et nous introduisons tous les champs correspondants ;
- nous sélectionnons la table "Commune", et nous sélectionnons le champ "Code postal". Il est inutile que nous sélectionnions le champ "Code _commune", puisque nous l'avons déjà pris dans la table "Personnes". Il est également inutile que nous sélectionnions le champ "Commune", puisque le nom de la commune s'affichera dans le contrôle "Code_commune", comme nous l'avons vu dans le paragraphe précédent ;
- l'assistant nous demande alors si nous souhaitons afficher les données "par Communes" ou "par Personnes". Nous choisissons la deuxième option, et nous voyons que cela nous conduit à un formulaire unique. Nous ferons l'essai de la première option dans le paragraphe suivant ;
- nous choisissons la disposition et le style, nous renommons le formulaire, et nous examinons le résultat, qui est représenté sur la figure ci-dessous, après un peu de mise en forme.

Nous constatons que, si nous saisissons ou modifions le nom d'une commune, le code postal correspondant s'affiche automatiquement -- ce que nous ne pouvons pas faire dans une table. Il faut par contre éviter que l'opérateur puisse modifier le code postal. Pour cela, nous désactivons ou nous verrouillons le contrôle "Code postal".

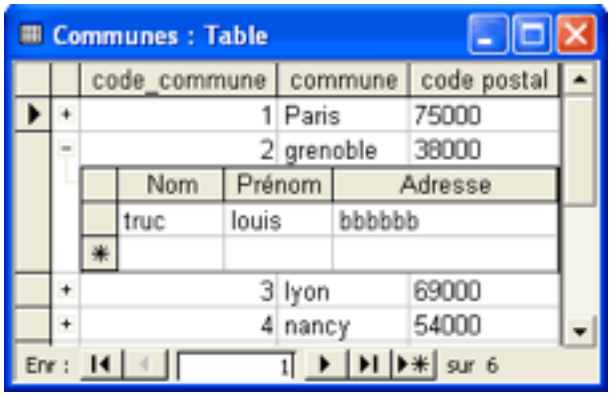
Le formulaire avec sous-formulaire

Reprenons la procédure précédente, mais demandons cette fois l'affichage des données par communes. Le dialogue avec l'assistant s'établit ainsi :

- nous sélectionnons la table "Communes", et nous introduisons les champs "commune" et "Code postal" ;
- nous sélectionnons la table "Personnes", et nous introduisons les champs "nom", "prénom" et "adresse" ;
- nous demandons l'affichage des données par Communes, dans un formulaire avec sous-formulaire, et non dans des formulaires attachés ;
- la meilleure disposition pour un sous-formulaire est "feuille de données" ;
- nous choisissons un style, nous renommons le formulaire et le sous-formulaire, et nous terminons l'opération.

Nous obtenons ainsi un formulaire principal dans lequel nous pouvons saisir des noms de communes avec leur code postal,et les utiliser dans le sous-formulaire associé. La figure ci-dessous illustre le résultat obtenu :

Nous voyons que le sous-formulaire correspond -- avec une présentation différente -- à la sous-table que nous avons rencontrée au chapitre 3, et dont nous reproduisons un exemple ci-dessous :



Il est possible d'introduire plusieurs sous-formulaires dans un même formulaire, mais l'assistant ne permet d'en créer qu'un seul. Les sous-formulaires supplémentaires peuvent être introduits avec l'aide de la boite à outils.

Table des matières

Gérer l'information

Stocker l'information

Les tables

Les index

Les listes de choix

Les relations

Le mécanisme relationnel

Les requêtes (retrouver des informations)

Imprimer l'information recherchée

Saisie d'information