Les bases de données relationnelles avec Access

TABLE DES MATIERES

1 Introduction

1.1 Définition

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. Aujourd'hui, les données sont de plus en plus 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.

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 BDD, et le terme anglais correspondant est Data Base.

Définition : une base de données est un ensemble structuré de données, géré à l'aide d'un ordinateur.

Certains auteurs restreignent la définition précédente en précisant "...un grand ensemble de données...". Cela n'a aucun sens, pour les deux raisons suivantes : la définition d'une BDD se réfère à la manière dont sont gérées les données (elles forment un ensemble structuré). Cette manière n'a aucun rapport avec le volume des dites données ; d'un point de vue théorique, la définition d'un "grand" ensemble de données est parfaitement arbitraire. Nul ne sait à partir de quelle taille un ensemble devient "grand".

On rajoute parfois deux conditions supplémentaires à la définition précédente : exhaustivité : la base contient toutes les informations requises pour le service que l'on en attend ; unicité : la même information n'est présente qu'une seule fois (pas de doublons).

Reprenons l'exemple de la base de données du personnel. Elle est utilisée pour la paye mensuelle, pour l'avancement, les mutations, 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 personnel les salaires et les charges correspondantes le commerce électronique, etc.

Les bases de données se sont introduites plus tardivement dans les établissements d'enseignement, qui n'ont pas les mêmes besoins que les entreprises.

Bien entendu, les bases de données existaient avant l'introduction de l'informatique au milieu du vingtième siècle, mais elles ne portaient pas encore ce nom. Pour stocker l'information, on utilisait des fiches, regroupées dans des boites appelées fichiers. Initialement, les fiches étaient triées manuellement. Avec l'introduction des perforations, le tri devint mécanique, puis électromécanique. Le développement des bases de données gérées par des moyens informatiques a rendu obsolètes ces anciennes techniques.

1.2 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 Sexe Adresse Ville Code postal

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

etc.

Dans une table, les termes ligne et enregistrement sont synonymes. Il en est de même pour les termes colonnes et champs. En anglais : row et column. 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 par importation. Par contre, 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.

1.3 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.

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, etc. 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, et utilisée par le service marketing). Le nombre d'utilisateurs utilisant une base de données est également extrêmement variable. Une BDD 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 principaux éditeurs (avec leurs parts de marché en l'an 2002, calculées sur le chiffre d'affaires) sont :

IBM (36 %), éditeur des SGBD DB2 (développé en interne - mis sur le marché en 1984) et Informix (obtenu par rachat de l'entreprise correspondante en 2001 ; la société Informix avait été créée en 1981. Une version bridée de DB2 vient d'apparaître sur le marché, où elle concurrence SQL Server de Microsoft ; Oracle (34 %), éditeur du SGBD qui porte le même nom. Cette entreprise a été créée en 1977 ; Microsoft (18 %), éditeur de trois SGBD. SQL Server est destiné aux gros systèmes, Access est un produit de bureautique professionnelle. L'arrivée de Microsoft sur le marché des SGBD date du début des années 90.

Ces chiffres recouvrent des réalités contrastées, quand on les fractionne par plate-forme. Dans le monde Unix, Oracle est en tête avec 62 %, suivi d'IBM (Informix compris) avec 27 %, alors que Microsoft n'est pas présent sur ce marché. Dans le monde Windows, Microsoft a pris la tête avec 45 %, suivi d'Oracle avec 27 % et d'IBM avec 22 %. 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. ! Les analystes pensent que le marché des SGBD fonctionnant sous Windows rejoindra celui des SGBD fonctionnant sous Unix. De manière schématique, on peut dire qu'Oracle arrive en tête pour la technicité, et Microsoft pour la convivialité et la facilité d'emploi. A fonctionnalités comparables, Oracle a la réputation d'être plus cher que ses principaux concurrents. En une vingtaine d'années, le marché des SGBD s'est fortement consolidé. Ainsi dBase, le SGBD le plus utilisé des années 80-90, n'a plus qu'une importance mineure. Paradox, qui eut son heure de gloire, semble avoir totalement disparu, bien que son format soit encore utilisé. 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 commodément avec le logiciel. Pour dialoguer avec les SGBD qui ne sont pas équipés d'une interface graphique, il faut utiliser le langage SQL (Structured Query Language), et introduire les instructions à l'aide d'un éditeur de lignes. Langage normalisé de manipulation des bases de données, SQL est utilisable avec pratiquement tous les SGBD du marché. Cependant, chaque éditeur ayant développé son propre "dialecte" -- comme c'est toujours le cas en informatique -- il faut pouvoir disposer d'un "dictionnaire" pour transporter une BDD d'un SGBD à l'autre. Ce "dictionnaire" a été développé par Microsoft sous le nom ODBC (Open Data Base Connectivity).

1.4 Le matériel (serveur de BDD)

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. Exemple : le logiciel Access 2002 de Microsoft, installé sur un PC récent, doté de 1 Go de mémoire vive et fonctionnant sous Windows XP. Si ces chiffres sont dépassés, ou si le temps de traitement des données devient prohibitif, il faut viser plus haut. Le micro-ordinateur doit être remplacé par un serveur de BDD, 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), et le SGBD bureautique par un SGBD prévu pour les grosses BDD multi-clients. 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 BDD. 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 BDD 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). En observant un serveur de BDD en cours de fonctionnement, on peut observer les trois cas de déséquilibre suivants :

la machine fait du "swapping", c'est à dire qu'elle passe son temps à promener des données entre la mémoire vive et la mémoire virtuelle (laquelle réside sur disque). Le remède consiste à augmenter la mémoire vive -- si la chose est matériellement possible ; si l'unité centrale est sous-occupée, alors que le disque dur ne cesse de tourner, la machine est sous-dimensionnée quant à sa mémoire de masse. Les remèdes : utiliser une interface disque plus performante (SCSI), un disque dur plus rapide, un système RAID 0. Ce cas est le plus fréquemment rencontré ; si l'unité centrale est utilisée à fond, alors que les disques durs sont peu sollicités, la machine est sous-motorisée. Les remèdes : utiliser une machine possédant des processeurs plus rapides, ou un plus grand nombre de processeurs.

Jusqu'à une date récente, 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 grosses

BDD, 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 la dernière décennie. Avec l'éclatement de la bulle Internet, les cordons de la bourse se sont resserrés, si bien que les services informatiques des entreprises commencent à recourir - de gré ou de force - à des systèmes plus décentralisés et de taille plus raisonnable.

1.5 L'administration de la base de données

L'ensemble "serveur de BDD + 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 la base de données. Dès que l'entreprise atteint la taille d'une grosse PME, l'administration de la BDD peut nécessiter la présence d'une personne à temps plein, voire plus.

Être administrateur de BDD 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. On remarque que, dans l'entreprise toujours, la spécialisation des informaticiens s'accroît. Pour être complet, le développement des sites Web contribue à créer un troisième pôle de compétences dans l'entreprise.

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

"Access", fait partie de la suite bureautique "Office" (version professionnelle) de l'éditeur Microsoft. Access est un SGBD de milieu de gamme, commercialisé à un coût très abordable, et conçu pour être utilisable à la fois par le développeur professionnel et l'utilisateur courant de micro-ordinateur (d'où son abord convivial).

La version 1 d'Access a été lancée au COMDEX, à l'automne de 1992. La version 2 date de 1994. Les versions suivantes portent le nom de l'année : 95, 97, 2000 (publiée en 1999) et 2002 (publiée en 2001). La version 2002 possède les caractéristiques suivantes : Elle peut contenir 2 Go de données au maximum. Ce chiffre est quelque peu trompeur car, en cours d'exploitation, la taille de la base de données grossit : de nouvelles données sont générées, d'autres ne sont plus utilisées, etc. Le SGBD ne récupère la place perdue que si on lui en donne l'ordre (compactage). Or la limite de 2 Go concerne la taille de la BDD avant compactage. La limite pratique (la taille après compactage) peut donc être notablement inférieure à 2 Go ; elle peut servir 5 à 10 utilisateurs simultanément ; chaque table peut comporter 2 millions d'enregistrements au maximum.

Si on installe Access sur un serveur de fichiers d'entrée de gamme (qui ne coûte pas plus cher qu'un PC musclé, mais dont les accès disques sont nettement plus rapides), on peut atteindre des volumes de données convenant à une petite PME. Au delà, il faut utiliser un SGBD gérant de plus gros volumes de données, et l'installer sur un serveur plus puissant. En fait, dans l'entreprise, c'est surtout la limitation à 5-10 utilisateurs qui restreint les usages d'Access à des opérations de taille modestes. Mais cela ne veut pas dire qu'un logiciel bureautique comme Access ne soit utile que dans les petites PME. Il a sa place un peu partout, et c'est la raison pour laquelle il se répand aussi vite : d'abord,

toutes les données manipulées dans une entreprise ne vont pas s'entasser dans une BDD unique, et il y a place pour des SGBD de puissances diverses ; ensuite, lorsqu'on recherche de l'information dans une grande base, une première interrogation permet généralement d'isoler la petite fraction des données auxquelles on s'intéresse. Lorsque c'est possible, on a intérêt à transférer la suite des opérations dans un tableur comme Excel ou un SGBD comme Access, qui sont plus conviviaux et où il est plus facile de manipuler les données. Ce transfert n'est facilement praticable que si les deux conditions suivantes sont remplies : le "gros" SGBD fonctionne sous Windows, et il permet d'exporter les résultats d'une requête vers la suite Office ; enfin, même si l'on planifie la création d'une grosse BDD consultée par de nombreux utilisateurs, il peut être utile de créer d'abord une maquette dans Access où, grâce à l'interface graphique, les développements sont beaucoup plus rapides. Lorsque la maquette fonctionne correctement, on effectue une migration vers un SGBD plus puissant tel que SQL Server ou DB2. Les SGBD qui fonctionnent sous Unix, et/ou ceux qui ne connaissent que leur format propriétaire, n'offrent pas cette possibilité.

2 Les tables

2.1 Introduction

Nous avons vu que, dans les BDD, les données sont stockées dans des tables. Voyons l'étude des BDD 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 !

2.2 La création d'une table

La première opération consiste à créer d'abord une base de données vide. Le logiciel réclame un nom de fichier et un seul, car toutes les informations relatives à la BDD seront stockées dans le même fichier. 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 BDD 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).

Sélectionner l'objet table, s'il ne l'est pas déjà par défaut. Trois méthodes sont proposées pour créer une nouvelle table :

créer une table en **mode création**. C'est la méthode générale, la seule à recommander

créer une table **à l'aide de l'assistant**. Ce dernier vous 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 à déconseiller 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.

2.3 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 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 tronquées 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 votre travail, cliquez sur l'icône  "Enregistrer" dans la barre d'outils. Lorsque tous les champs sont définis, fermez 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, plus tard, 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.

2.4 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. Plus loin dans le cours nous verrons 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).

la propriété "mode IME" concerne l'usage d'Access en japonais, et ne nous intéresse donc pas ici.

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 ;

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.

==========

2.5 La taille du champ dans Access

2.5.1 Type de données "Texte"

Un champ "Texte" peut contenir au maximum 255 caractères. La valeur proposée par défaut est 50. Pour saisir un texte contenant plus de 255 caractères, utiliser le type de données "Mémo". Un champ "Mémo" peut contenir 65.535 caractères.

Attention ! Si la taille du champ de type "Texte" est modifiée alors que des données se trouvent déjà dans la table, ces dernières peuvent être tronquées à droite.

2.5.2 Type de données "Numérique"

Les divers choix proposés sont rassemblés dans le tableau ci-dessous. Attention ! une modification de propriété alors que la table contient des données entraîne la perte de celles qui ne satisfont pas à la nouvelle propriété.

	Propriété	**Valeur**	**Rem.**	**Taille**
Octet	Entier compris entre 0 et 255 (2^8)	Entier	1 octet	
Entier	Entier compris entre -32768 (-2^{15}) et 32767 (2^{15}-1)	Entier	2 octets	
Entier long	Entier compris entre -2^{31} et 2^{31} -1	Entier	4 octets	
Réel simple	Réel compris entre -3,402823E38 et -1,401298E-45 pour les valeurs négatives et 1.401298E-45 et 3.402823E38 pour les valeurs positives	Réel à 7 chiffres significatifs	4 octets	
Réel double	Réel compris entre -1,79769313486231E308 et -4,94065645841247E-324 pour les valeurs négatives et 4,94065645841247E-324 et 1,79769313486231E308 pour les valeurs positives	Réel à 15 chiffres significatifs	8 octets	
Décimal	Réel compris entre -10E28 et 10E28 environ	Réel à 18 chiffres significatifs	12 octets	

2.5.3 Type de données "NuméroAuto"

Le numéro automatique est un entier long. Il est attribué par le système, et l'utilisateur ne peut pas le modifier. L'incrémentation peut être régulière (+1), ou aléatoire (nombres signés). Le même numéro n'est jamais attribué deux fois.

le cas du numéro de réplication, qui concerne la synchronisation de plusieurs copies (réplicas) de la même base.

Attention ! lorsqu'une ligne est supprimée, le numéro automatique n'est pas réutilisé par le système.

==========

2.6 Les formats dans Access

format prédéfini ou format personnalisé

2.6.1 Introduction

La propriété "**Format**" caractérise la manière dont les données s'affichent dans une table, mais elle n'affecte pas leur enregistrement dans la base. On peut donc, en général, modifier le format (à condition de ne pas changer le type de données) sans détruire les données déjà présentes.

L'utilisateur peut définir ses propres formats, appelés formats personnalisés, ou utiliser des formats prédéfinis proposés par le SGBD Access. Les différentes possibilités sont rassemblées dans le tableau suivant, et nous les examinerons tour à tour. Des formats prédéfinis sont également disponibles pour le type de données NuméroAuto -- ce qui constitue une bizarrerie du logiciel -- et pour le type booléen, mais dans les formulaires et les états seulement. Dans les tables, seule la case à cocher traduit le booléen.

Type de données	Format personnalisé	Format prédéfini
Texte	Oui	Non
Mémo	Oui	Non
Numérique, monétaire	Oui	Oui
Date / Heure	Oui	Oui
Lien hypertexte	Oui	Non

Ainsi, la propriété "Format" est utilisable chaque fois que l'on manipule du texte, des nombres, des dates et des heures. Le format est appliqué par le SGBD au moment où l'on valide l'enregistrement (par passage à la ligne suivante, par exemple). Si vous revenez

dans le champ, le logiciel affiche la valeur telle que vous l'avez saisie.

Vous noterez que certains formats prédéfinis sont affectés par la définition des paramètres régionaux, que l'on peut modifier dans le Panneau de Configuration du système d'exploitation Windows. Ainsi, c'est le symbole de l'euro qui est affiché (type de données monétaire) pour un pays européen, et celui du dollar pour les États-Unis.

Vous noterez également que, définie au niveau d'une table, la propriété "Format" s'applique également aux contrôles des formulaires et des états construits à partir de cette table.

Attention ! quand vous définissez un format, effectuez une sauvegarde en cliquant sur l'icône "Enregistrer", et regardez si le logiciel n'a pas modifié votre définition ; cela peut vous éviter bien des surprises. N'hésitez pas à basculer en mode "Feuille de données" à chaque essai, pour voir si le résultat est conforme à ce que vous attendez.

Remarque : la propriété "Format" est prioritaire sur la propriété "Masque de saisie" (voir plus loin). Lorsqu'un format est défini, le masque de saisie est ignoré.

2.6.2 Format personnalisé pour le texte

Le paramétrage exposé ci-dessous s'applique aux types de données texte, mémo et lien hypertexte. Les symboles spécifiques sont rassemblés dans le tableau suivant.

Symbole	**Signification**
@	Caractère (ou espace)
&	Caractère (pas d'espace)
<	Affiche le texte en minuscules
>	Affiche le texte en majuscules
"abc"	Affiche les caractères compris entre guillemets (utiliser pour plus d'un caractère)
\	Affiche le caractère suivant sous forme littérale
[Rouge]	Affiche dans la couleur spécifiée. Disponibles : Noir, Bleu, Vert, Cyan, Rouge, Magenta, Jaune et Blanc
*	Utilise le caractère suivant pour compléter le champ

Les symboles @ et & ont un fonctionnement identique, sauf en ce qui concerne les espaces. Vous noterez que certains caractères (le tiret par exemple) n'ont pas besoin d'être mis entre guillemets. Des exemple d'utilisation sont rassemblés dans le tableau suivant.

Format	Saisie	Affichage
@-@@	a	- a
	ab	-ab
	abc	a-bc
&-&&	a	-a
	ab	-ab
	abc	a-bc
<	Paul	paul
>	Paul	PAUL
@[Rouge]	abc	abc
&\ &	AB	A B
"**"&	ABC	AB**C
&*-	ab	ab----------
@;"Vide"	abc	abc
		Vide

Les cases qui ne contiennent aucune information peuvent être l'objet d'un format particulier; comme le montrent les exemples contenus dans les deux dernières lignes du tableau ci-dessus. Après le point-virgule se trouve le texte qui sera affiché systématiquement dans les cases vides.

2.6.3 Format de données numériques et monétaires

Le SGBD Access propose des formats prédéfinis pour les données numériques et monétaires. Il suffit de cliquer dans le champ de définition de la propriété "Format" pour faire apparaître l'icône d'une liste déroulante, et de consulter cette dernière.

Si les formats proposés ne vous conviennent pas, n'hésitez pas à créer un format personnalisé. Les symboles utilisables sont rassemblés dans le tableau ci-dessous.

Symbole	Signification
, (virgule)	Séparateur décimal
0	Affiche un chiffre ou zéro
#	Affiche un chiffre ou rien
\	Affiche le caractère suivant (un seul caractère)
" € HT"	Affiche les caractères entre guillemets (plus d'un caractère)
%	Pourcentage. La valeur saisie est multipliée par 100 et suivie du signe %
E- ou e-	Notation scientifique (pas de signe pour les exposants positifs)
E+ ou e+	Notation scientifique (signe + pour les exposants positifs)
!	Force l'alignement à gauche
[Rouge]	Affiche avec la couleur spécifiée. Disponibles : Noir, Bleu, Vert, Cyan, Rouge, Magenta, Jaune et Blanc
*	Utilise le caractère suivant pour compléter le champ

Voici quelques exemples :

Format	Saisie	Affichage
#,##	127	127,
	127,758	127,75
	0,12	,12
	0	,
0,00 (fixe)	127	127,00
	127,758	127,76
	0,12	0,12
	0	0,00
# ###	127	127
	127,758	128
	0,12	
	0	
# ##0,00 (standard)	1248,159	1 248,16

	127	127,00
	0	0,00
#[Vert]	253	253
!	127	127
0*-	127,8	127----------
0,00" € HT"	127,758	127,76 € HT

Un format numérique peut comporter de une à quatre sections, et utilise le point-virgule comme séparateur. Le rôle de chaque section est défini dans le tableau ci-dessous.

Section	Rôle
Première	Format des nombres positifs
Seconde	Format des nombres négatifs
Troisième	Format de la valeur zéro
Quatrième	Format d'un champ vide

Exemples :

Format	Saisie	Affichage
"$ "# ##0,00[Vert];"$ -"# ##0,00[Rouge];;"néant"	127,758	$ 127,76
	-2541	$ -2 541,00
	0	$ 0,00
		néant
0,0;(0,0);"zéro";"A remplir"	127,758	127,8
	-127,758	(127,8)
	0	zéro
		A remplir
0,000E+;;\0	127,758	1,278E+2
	-127,758	-1,278E+2
	384	3,840E+2
	,001	1,000E-3
	0	0

Vous noterez qu'il considère 0,00 $ comme une somme positive.

2.6.4 Format de données de type date ou heure

Comme pour les données numériques, le SGBD Access propose des formats prédéfinis pour les données de type date ou heure. Il suffit de cliquer dans le champ de définition de la propriété "Format" pour faire apparaître l'icône d'une liste déroulante, et de choisir dans cette dernière.

Si les formats proposés ne vous conviennent pas, n'hésitez pas à créer un format personnalisé. La majeure partie des symboles utilisables est rassemblée dans le tableau ci-dessous.

Symbole	Signification
:	Séparateur d'heure (se règle dans le panneau de configuration de Windows)
/	Séparateur de date (id)
j	Affiche le jour du mois (un ou deux chiffres suivant besoin)

jj	Affiche le jour du mois (deux chiffres)
jjj	Affiche le jour de la semaine (nom abrégé)
jjjj	Affiche le jour de la semaine (nom entier)
e	Affiche le jour de la semaine (de 1 à 7)
ee	Affiche la semaine de l'année (de 1 à 53)
m	Affiche mois de l'année (un ou deux chiffres suivant besoin)
mm	Affiche le mois de l'année (deux chiffres)
mmm	Affiche le mois de l'année (nom abrégé)
mmmm	Affiche le mois de l'année (nom entier)
t	Affiche le trimestre
a	Affiche le numéro du jour dans l'année
aa	Année (deux derniers chiffres)
aaaa	Année (quatre chiffres)
h	Affiche l'heure (un ou deux chiffres suivant besoin) (de 1 à 24)
hh	Affiche l'heure (deux chiffres) (de 1 à 24)
n	Affiche les minutes (un ou deux chiffres suivant besoin) (de 0 à 59)
nn	Affiche les minutes (deux chiffres) (de 0 à 59)
s	Affiche les secondes (un ou deux chiffres suivant besoin) (de 0 à 59)
ss	Affiche les secondes (deux chiffres) (de 0 à 59)

Si vous saisissez une date dans un format reconnu par le SGBD, ce dernier n'en retiendra que la partie définie par ces symboles. Exemples :

Format	**Saisie**	**Affichage**	**Signification**
j	2/09/2002	2	Jour
jj	id.	02	Jour
jjj	id.	lun	Jour
jjjj	id.	lundi	Jour
e	id.	1	N° jour / semaine
ee	id.	36	Semaine
m	id.	9	Mois
mm	id.	09	Mois
mmm	id.	sept	Mois
mmmm	id.	septembre	Mois
t	id.	3	Trimestre
a	id.	245	N° jour / année
aa	id.	02	Année
aaaa	id.	2002	Année

h	5:8:3	5	Heure (/24)
hh	id.	05	Heure (/24)
n	id.	8	Minutes
nn	id.	08	Minutes
s	id.	3	Secondes
ss	id.	03	Secondes

Il suffit de combiner ces symboles pour constituer un format personnalisé complet. Exemples :

Format	Saisie	Affichage
jjjj", le "j\ mmmm\ aaaa	2/09/2002	lundi, le 2 septembre 2002
"Semaine n° "ee	id.	Semaine n° 36
"Nous sommes "jjjj	id.	Nous sommes lundi
"Année "aaaa", le "jj" / "mm	id.	Année 2002, le 02 / 09
jjjj\ j\ mmm\ aaaa	id.	lundi 2 sept 2002
h" heures "n" min "s" sec"	2:21:5	2 heures 21 min 5 sec

Les formats relatifs à la date et à l'heure sont surtout utilisés quand on imprime des données, c'est à dire quand on utilise les états. Dans les tables, on se contente généralement des formats prédéfinis.

==========

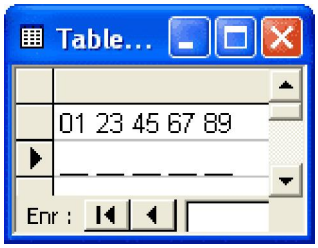
2.7 Les masques de saisie(prédéfinis ou personnalisés)

2.7.1 Introduction

Le **masque de saisie** permet de guider la saisie des données dans une table ou un formulaire, lorsque ces données ont des propriétés communes (ex : un nom doit commencer par une majuscule, "Tél." précède un numéro de téléphone, "mailto:" précède une adresse e-mail, etc.).

Le SGBD Access permet l'utilisation d'un masque de saisie pour les types de données texte, numérique, monétaire et date / heure.

Un assistant "Masque de saisie" est à votre disposition, mais il ne fonctionne pas pour les types de données numérique et monétaire. Vous pouvez lancer l'assistant en cliquant dans le champ de la propriété "Masque de saisie" (lors de la création ou de la modification de la table), puis sur le bouton ... Une liste de masques prédéfinis vous est alors proposée : code postal français ou international, numéro de sécurité sociale avec ou sans clé, etc. La figure ci-dessous montre l'utilisation d'un masque pour la saisie de numéros de téléphone français.



Les données que vous saisissez dans un masque peuvent être stockées :

Telles qu'elles sont saisies. Dans l'exemple précédent, cela signifie que les espaces situés entre les chiffres sont stockés avec ces derniers ;

Sans les caractères littéraux. Dans l'exemple précédent, cela signifie que seuls les chiffres seront enregistrés.

Le choix entre ces deux solutions ne prend de l'importance que si vous envisagez de supprimer le masque ultérieurement. Dans la première alternative, vos données garderont leur structure (01 23 45 67 89), dans la seconde tous les chiffres seront accolés (0123456789).

Si aucun élément de la liste des masques prédéfinis ne vous convient, vous avez la possibilité de créer vous-même un masque personnalisé.

2.7.2 Structure d'un masque

La définition d'un masque de saisie peut comporter jusqu'à trois sections séparées par des points-virgules. Seule la première section est impérative. Le tableau ci-dessous précise le rôle de chaque section.

Section	Rôle
Première	Définition du masque de saisie à l'aide de symboles
Seconde	Stockage des caractères littéraux : 0=oui, 1=non
Troisième	Caractère matérialisant la saisie

Par défaut, le caractère matérialisant la saisie est le soulignement (_), mais n'importe quel autre caractère est utilisable. Si on choisit l'espace, il faut le placer entre guillemets : " ".

2.7.3 Symboles

Les symboles utilisés pour définir un masque de saisie personnalisé sont rassemblés dans le tableau ci-dessous.

Symbole	Signification
0	Chiffre (0 à 9, saisie obligatoire ; signes plus [+] et moins [-] non autorisés)
9	Chiffre ou espace (saisie facultative ; signes plus et moins non autorisés)
#	Chiffre ou espace (saisie facultative ; espaces supprimés, signes plus et moins autorisés)
L	Lettre (A à Z, saisie obligatoire)
?	Lettre (A à Z, saisie facultative)
A	Lettre ou chiffre (saisie obligatoire)
a	Lettre ou chiffre (saisie facultative)
&	Tout caractère ou espace (saisie obligatoire)
C	Tout caractère ou espace (saisie facultative)
<	Conversion en minuscules de tous les caractères qui suivent
>	Conversion en majuscules de tous les caractères qui suivent
\	Le caractère qui suit s'affiche de manière littérale

2.7.4 Exemples

Le tableau ci-dessous rassemble quelques exemples de masques appliqués à un champ de texte. La première colonne contient la définition du masque, telle qu'elle est introduite dans la zone de texte de la propriété "Masque de saisie". La seconde colonne montre ce qui apparaît lorsqu'on clique dans le champ correspondant en mode feuille de données. La troisième colonne représente un exemple valide de saisie.

Définition du masque	Avant saisie	Ex. de saisie
00\ 00\ 00\ 00\ 00;0;_	__ __ __ __ __	01 23 45 67 89
LLL"--"??;;@	@@@--@@	aze--r
\(000") "000\-0000;1;*	(***) ***-****	(206) 555-0248
>L<CCCCCCCCCCCCCCC	_____	Jean-claude sohm
"ISBN "0\-&&&&&&&&&\-0	"ISBN "_-_____-_	ISBN 5-126795111-8
####	____	-26

Comme pour les formats, il est fortement conseillé de sauvegarder la définition de la table en cliquant sur l'icône "Enregistrer", puis de regarder si le SGBD n'a pas modifié le masque proposé, avant de passer en mode feuille de données pour faire des essais.

A l'usage, si les données saisies ne correspondent pas au masque, le SGBD ne proteste que lorsqu'on valide par passage à la ligne suivante, ou fermeture de la table.

En conclusion, les masques sont fort utiles pour limiter les erreurs de saisie lorsque les données ont un format bien défini.

==========

voir valeurs par défaut ;

voir règles de validation ;

voir traitement d'un champ vierge ;

voir indexation d'un champ (plus loin dans le cours) ;

2.7.5 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". 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'enregistrement 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... donc changer ce message si possible.

2.7.6 Conclusion

Il est essentiel de bien réaliser que, dans les BDD, les tables 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. Lors des opérations d'import/export, le système nous demandera si seule la structure est transportée, ou si les données doivent suivre.

3 Les Index

3.1 Introduction

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, célèbre dans l'Antiquité. 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 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 (quelque peu 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.

3.2 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 BDD 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	etc.
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	2001	4-1985-5891-3	...
4	etc.					

En informatique, un index est une table à une seule colonne, représenté par 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 3 1 ..	2 1 3 ..	2 1 3 ..	1 3 2 ..	3 1 2 ..
Index titre	Index auteur	Index éditeur	Index année	Index 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 dupliqués 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 :

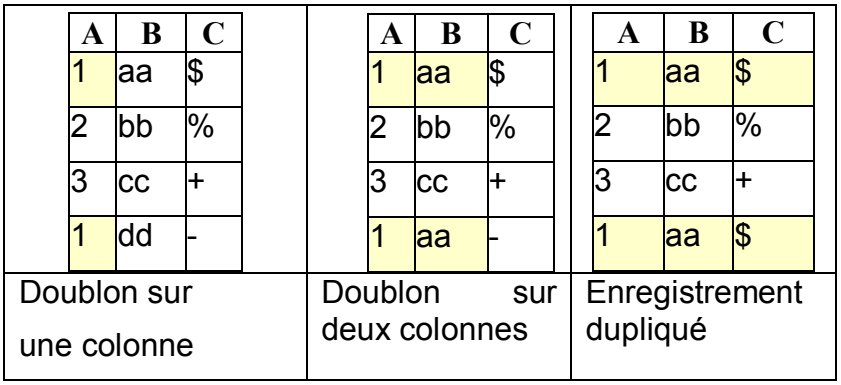
chaque fois que nous demandons au système de créer (et de maintenir) un index, nous augmentons sa charge de travail, et par conséquent nous le freinons. Ainsi, les opérations de saisie et de maintenance sont ralenties par la présence d'index, car ces derniers doivent être mis à jour immédiatement ;

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 le nom et le prénom.

3.3 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 ce dernier 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	aa	-

Doublon sur deux colonnes

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

Enregistrement dupliqué

Dans une BDD, 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 BDD 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 facilement si cette nouvelle information constitue un doublon sur les champs concernés. Il est donc 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.

3.4 L'indexation d'un champ

Dans le chapitre consacré aux tables, nous avons rencontré 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.

3.5 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 :

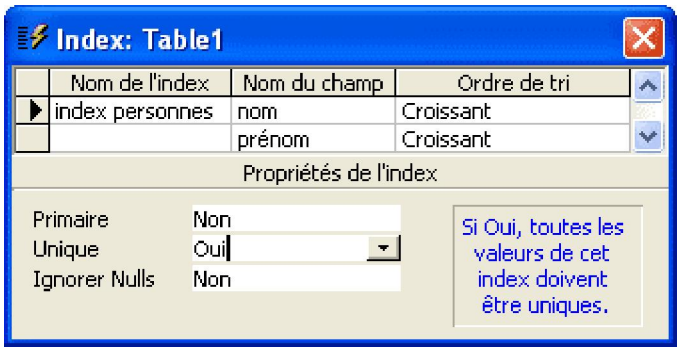
dans la colonne de gauche, on donne un nom à l'index multi-champ ;

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.

3.6 Conclusion

Les index jouent un rôle discret mais important dans la gestion des BDD.

La décision de créer un index résulte de l'examen des avantages et inconvénients de cette opération. L'index ralentit les saisies et consomme un peu de place, mais il rend les tris et les recherches d'information plus rapides. Bien entendu, il est inutile de créer un index sur quelque champ que ce soit dans une table qui renferme très peu d'enregistrements.

L'index joue un rôle important dans l'élimination des doublons résultant d'erreurs de saisie.

4 Les listes de choix

4.1 Introduction

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

Nom	Prénom	Titre	Adresse	Commune	Code postal
Durand	Pierre	M.	31 rue des champs	Uriage	38410
Chose	Stéphanie	Melle	2 place Stanislas	Nancy	54000
Trombe	Jean	M.	18 cours de la libération	Grenoble	38001
Machin	Andrée	Mme	10 cours Berriat	Grenoble	38000
etc.					

Il est tout à fait fastidieux de saisir de nombreuses fois la même information, telle que celle du titre (Mme, Melle, 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..

4.2 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.

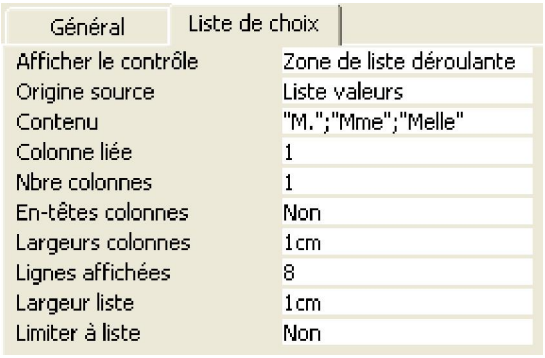
Lors de la création de la table, lorsque nous arrivons au type de données du champ "Titre", nous sélectionnons "Texte", puis "Assistant liste de choix..." dans la liste déroulante qui est à notre disposition. Nous procédons alors aux opérations suivantes :

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 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.

4.3 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.

Remarque : on ne peut rendre la liste obligatoire que si la propriété "Afficher le contrôle" est à "Zone de liste déroulante". Une "Zone de liste" ne le permet pas

4.4 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" ;

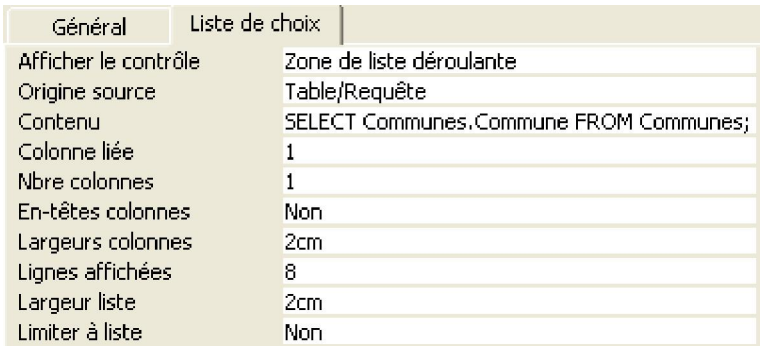
nous sélectionnons son unique champ, intitulé "Commune" ;

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.

4.5 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, 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.

4.6 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.

4.7 L'usage d'un code

Nous désirons maintenant associer le code postal au nom de la commune. Un problème naît immédiatement du fait que, en France du moins, plusieurs codes postaux peuvent être attribués à une même commune, lorsque cette dernière est une grande ville. A Paris, par exemple, chaque arrondissement possède son code postal, lequel varie donc de 75001 à 75020. A Grenoble, bien qu'il n'y ait pas d'arrondissement, il existe deux codes postaux (38000 et 38001). 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 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 NuméroAuto (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* que nous avons créée au paragraphe 4. 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 NuméroAuto. 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.

4.8 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.

4.9 Conclusion

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.

Nous reviendrons, au chapitre 9, sur l'usage des listes, et sur les rapports complexes qui existent entre liste et relation.

5 Le mécanisme relationnel

5.1 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.

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 pour créer les SGBD relationnels.

Il est de bon ton, dans l'enseignement des BDD, 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.

5.2 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.

5.3 L'informatique arrive

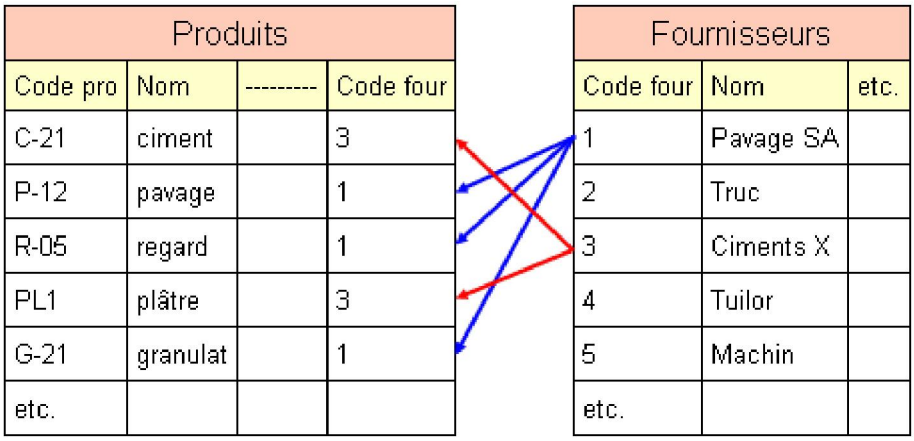
Le temps passe, le coût de la main d'oeuvre ne cesse d'augmenter, et les décideurs de l'entreprise font leurs comptes : informatiser la gestion des données courantes de l'entreprise va permettre de faire des économies. D'ailleurs, les concurrents suivent le même chemin, pas question de rester à la traîne.

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.

5.4 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	Tuilor	
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.

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.

5.5 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 ?

Bons de livraison							
N°	Date	Code ent.	Code_prod	Quantité	Code_prod	Quantité	Etc.....?
1225	15/02/2001	SGTDM	---------	------	---------	------	
1226	15/02/2001	XYZT	----------	---	----------	---	
1227	15/02/2001	CQFD	--------	------	--------	------	
etc.							

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". 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.

Table de jonction		
N° bon	Code_prod	Quantité
1225	C-22	30
1225	P-5	2
1225	R-10	5
1226	etc.	

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. Donc, 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".

5.6 Conclusion

Nous avons tenté de montrer, à l'aide d'exemples concrets, que les bases de données relationnelles tirent leur origine dans la manière dont les entreprises géraient leurs données avant la grande vague d'informatisation de ces 20 dernières années. Les tables et leurs enregistrements sont issus des fichiers et de leurs fiches. Les clés et les relations proviennent directement de l'usage des codes. Le problème de la relation n-n, par contre, est spécifique de l'introduction de l'informatique, mais la création d'une table de jonction résout le problème sans grande difficulté.

Les premiers SGBD mis sur le marché ne géraient pas les relations, et les entreprises les trouvaient fort malcommodes. Le succès des SGBD relationnels provient du fait qu'ils répondent bien aux besoins des entreprises -- parce qu'ils ont été conçus pour cela.

6 Le schéma relationnel de la base

6.1 Introduction

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 BDD 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 BDD citent cet article, mais que fort peu l'ont lu.

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 BDD.

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 BDD 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".

6.2 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.

6.3 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.

6.4 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
etc.		

Table "Personnes"

Code_org	Organisme
1	ABCD
2	XYZ
3	CQFD
4	TAGADA
etc.	

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**.

6.5 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 TAGADA.

Nom	Code_per
Durand	1
Chose	2
Machin	3
Truc	4
etc.	

Table "Personnes"

Code_per	Code_org
2	1
1	3
1	4
3	3
etc.	

Table "Affiliation"

Code_org	Organisme
1	ABCD
2	XYZ
3	CQFD
4	TAGADA
etc.	

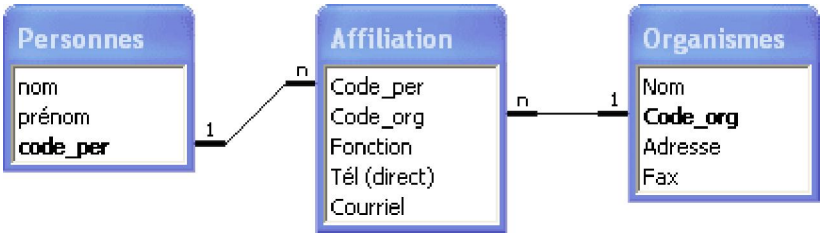
Table "Organismes"

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**.

6.6 Le 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.

6.7 Conclusion

Dans le processus de création d'une base de données, l'établissement du schéma relationnel de la base de données représente l'étape fondamentale. Il est inutile d'aller plus loin, et de se ruer sur l'ordinateur, tant que cette étape n'est pas parfaitement maîtrisée.

Comme vous pouvez le constater, on n'utilise pas de moyens informatiques au cours de cette étape. Il existe certes des logiciels d'aide à la création du schéma relationnel, qui rendent service dans les cas très complexes, mais les cas que vous rencontrerez nécessiteront surtout de la réflexion, de la méthode et du bon sens. Vos outils seront du papier, un crayon... et une bonne gomme !

Lorsque le schéma relationnel vous parait bon, testez-le par simulation sur papier. Suivez les relations et vérifiez que pouvez remplir les tables sans problème. Alors, mais alors seulement, vous pouvez vous asseoir devant l'ordinateur, et lancer le SGBD. Mais là encore, soyez prudent : dès que vous avez introduit une petite quantité de données, testez le système et retestez-le. Car corriger le schéma relationnel d'une BDD qui est déjà remplie de données est presque toujours une opération douloureuse.

7 Les relations

7.1 Introduction

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.

7.2 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 dans un chapitre précédent. 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".

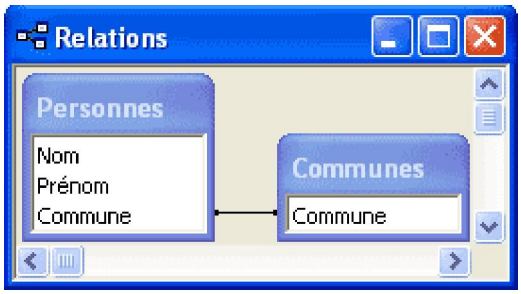
Cet exemple simple nous montre qu'une liste de choix externe n'est pas différente, dans son principe, d'une relation 1-n entre deux tables. Les différences se manifestent sur le plan pratique, car les techniques utilisées pour créer une liste externe et une relation ne sont pas exactement les mêmes.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

7.3 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.

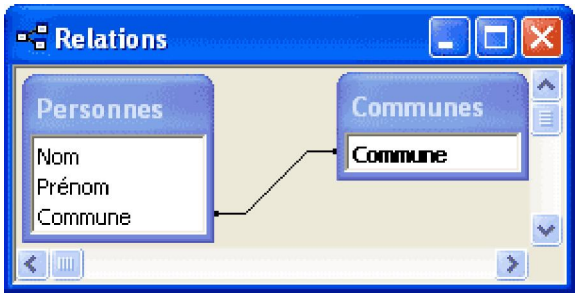
7.4 L'utilisation de la clé

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. Comment pose-t-on 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.

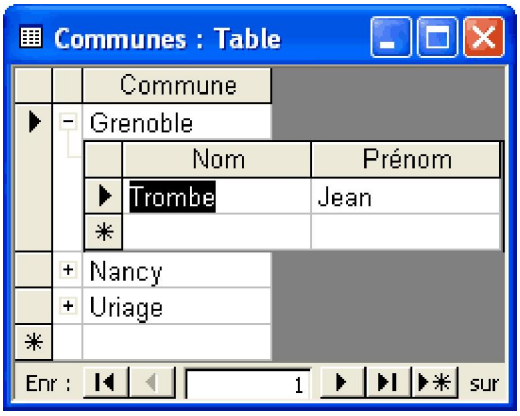
Dans la fenêtre "Relations", la présence de la clé est révélée par le fait que le nom du champ correspondant est écrit en caractères gras, comme le montre la figure ci-dessous.



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

7.5 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.

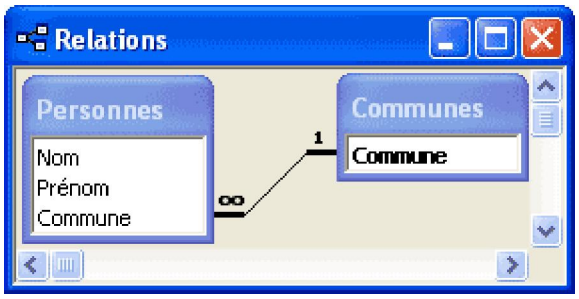
7.6 L'intégrité référentielle

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 BDD. 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 tard, dans le cadre des requêtes.

7.7 Conclusion

Nous avons vu, sur un exemple simple (deux tables liées par une relation 1-n), comment créer une relation et la doter des propriétés qui assurent la cohérence de la base de données (intégrité référentielle). Nous avons reconnu au passage des notions que nous avions déjà rencontrées à propos des listes: l'usage de la clé, les sous-tables. Il serait maintenant bon que nous étudiions ce que les listes et les relations ont en commun, et ce qui les différencie.

8 Les listes comparées aux relations

8.1 Introduction

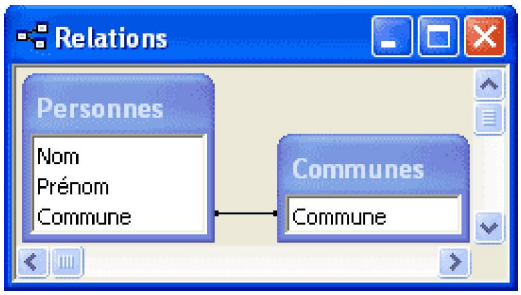
Arrivés à ce stade de notre étude des SGBD, nous sommes amenés à nous poser la question suivante : quelle est la différence entre une liste externe (basée sur une table) et une relation ? Pourquoi ne pas toujours utiliser l'une et ignorer l'autre ?

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.

8.2 La relation sous-jacente à une liste

Nous réutilisons l'exemple 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".

8.3 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 permettant 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 ?

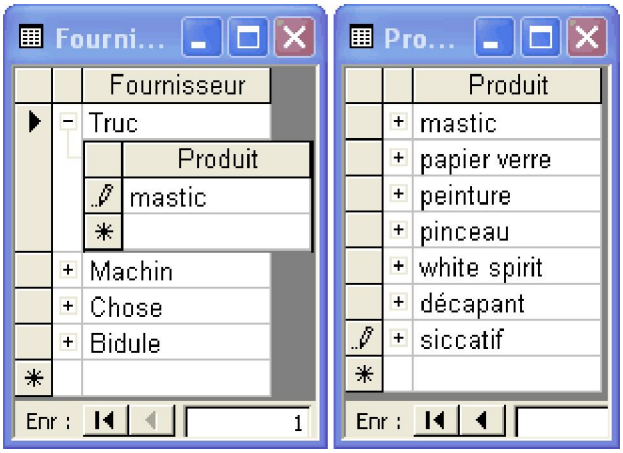
Bref, une liste fonctionne à la fois comme une liste et comme une relation. Il est des cas où nous n'en avons pas vraiment besoin, mais il est aussi des cas où cela peut nous rendre service -- celui des tables de jonction, par exemple.

8.4 Le cas des tables de jonction

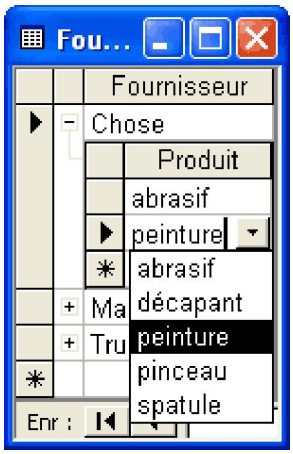
Rappelons qu'une table de jonction est une table que l'on crée pour scinder une relation n-n en deux relations 1-n. Pour illustrer ce cas, nous utiliserons l'exemple d'une liste de fournisseurs et de leurs produits.

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.

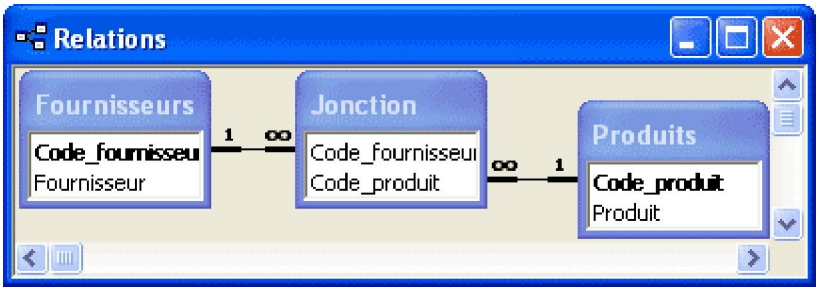


Supprimons les relations, reconstruisons-les sous forme de liste, puis dotons-les de l'intégrité référentielle. La nouvelle situation est représentée par la figure ci-dessous. Si la table "Produit" est renseignée, on peut remplir la table "Fournisseurs" et la table "Jonction" (sous-table) dans une seule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



8.5 L'usage des codes

Tout ce que nous avons dit 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, on masque les codes, et tout se passe comme s'ils n'existaient pas quand on remplit les tables.

8.6 Conclusion

Une liste de choix externe (c'est à dire basée sur une table) est une relation 1-n dotée d'un mécanisme particulier. Il est souvent intéressant de créer une relation sous forme de liste d'abord, puis de doter ensuite la relation sous-jacente des propriétés requises (intégrité référentielle). Cette technique est particulièrement intéressante pour la saisie des informations dans les tables de jonction.

Il ne faut pas hésiter à utiliser des codes pour régler des problèmes d'homonymie. Il faut absolument confier la gestion de ces codes au SGBD, et nous conseillons de faire en sorte que ces codes soient cachés à l'opérateur -- à moins qu'il ne tienne absolument à les voir, bien entendu.

9 La recherche manuelle

9.1 Introduction

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**.

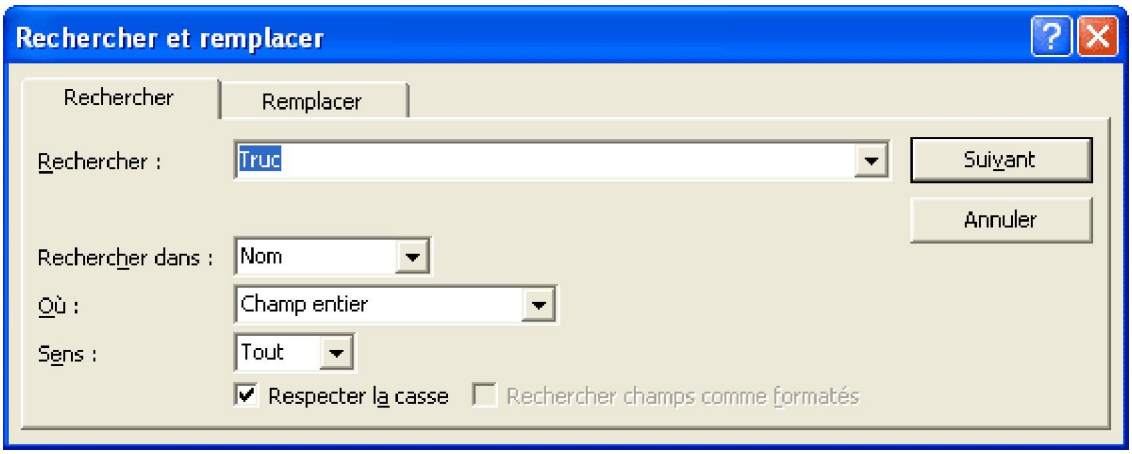
Ces techniques de recherche "manuelle" ne permettent pas d'effectuer des opérations très sophistiquées, mais elles ont le mérite de la rapidité et de la simplicité. Quand elles s'avèrent insuffisantes, il faut utiliser l'outil de recherche dont sont dotés tous les SGBD, et qui s'appelle la requête. Voir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

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.

Comme pour les autres chapitres de ce cours, nous utiliserons le SGBD Access (version 2002) comme support pratique. On notera que, dans ce logiciel, tout ce qui concerne le tri et les filtres s'applique non seulement aux tables, mais aussi aux formulaires.

9.2 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 :" !

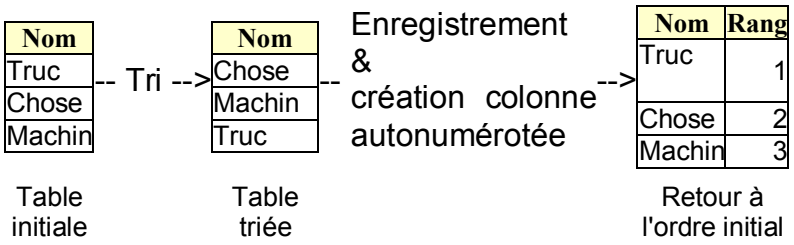
9.3 Le tri

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 PC 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 SCSI (un serveur de fichiers, par exemple) se montrera beaucoup plus rapide qu'un PC 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 -- ce que confirme l'expérience suivante.

Créons une table à une seule colonne de type texte, saisissons quelques chaînes de caractères, puis trions la table en ordre croissant et enregistrons-la en confirmant la modification. Créons alors une seconde colonne de type NuméroAuto, enregistrons cette modification, et consultons la table. Surprise : la numérotation de la second colonne ne suit pas l'ordre alphabétique, mais l'ordre de création initial, comme le montre la figure ci-dessous.



Conclusion : pour changer définitivement l'ordre des informations enregistrées dans une table, il faut utiliser une requête permettant de recréer la table sous un autre nom, ou de l'insérer dans une autre table (vide).

9.4 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 contient 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 :

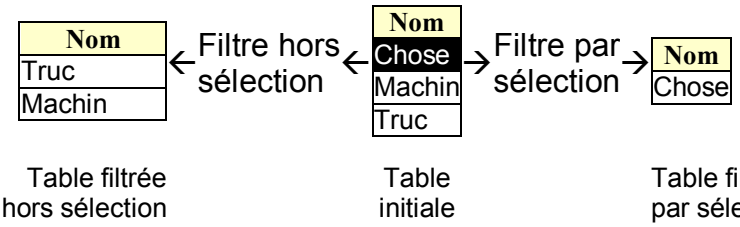
N'est pas en contact avec les extrémités du champ, le SGBD retient les enregistrements qui contiennent cette chaîne (n'importe où dans le champ) ;

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".

La figure ci-dessous illustre le fonctionnement des deux filtres (par sélection et hors sélection) dans un cas fort simple :



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.

9.5 Le filtre par formulaire

Le **filtre par formulaire** permet de filtrer une table en utilisant simultanément plusieurs chaînes de caractères, liées par des opérateurs logiques ET et OU. Il peut être considéré comme un perfectionnement du filtre par sélection, avec cependant une réserve : les chaînes choisies représentent obligatoirement le contenu exact du champ.

Pour illustrer le fonctionnement de ce filtre sur un exemple, nous créons une table ("Table5") contenant quatre colonnes, et nous y introduisons des données comme représenté sur la figure ci-dessous. Puis nous cliquons sur l'icône "Filtrer par formulaire". S'ouvre alors une fenêtre intitulée "Table5: Filtrer par formulaire". Une liste déroulante (dédoublonnée) nous permet, pour chaque colonne, de choisir une donnée : le SGBD filtrera en appliquant l'opérateur logique ET entre ces termes. Nous pouvons également effectuer un OU, en cliquant sur l'onglet du même nom. Pour appliquer le filtre, nous cliquons sur l'icône "Appliquer le filtre", et nous obtenons le résultat représenté ci-dessous. Pour supprimer le filtre, nous cliquons une seconde fois sur l'icône qui s'appelle maintenant "Supprimer le filtre".

Table "Table5" initiale

Commercial	Produit	Nombre	Date
Chose	Lave-vaisselle	3	03/03/2003
Machin	Aspirateur	5	04/03/2003
Chose	Cocotte-minute	6	03/03/2003
Truc	Réfrigérateur	4	03/03/2003
Machin	Mixer	4	04/03/2003
Chose	Lave-vaisselle	1	04/03/2003

Filtre par formulaire

Table "Table5" filtrée par formulaire

Commercial	Produit	Nombre	Date
Chose	Lave-vaisselle	3	03/03/2003
Chose	Cocotte-minute	6	03/03/2003

A notre connaissance, le filtre par formulaire est peu utilisé. On peut lui reprocher son manque de souplesse : les chaînes que l'on choisit représentent exactement le contenu du champ. De plus, si on pratique un OU, le premier terme de l'alternative disparaît de l'écran. Le filtre par sélection, plus simple, mais plus souple, rend de meilleurs services.

9.6 Le filtre/tri

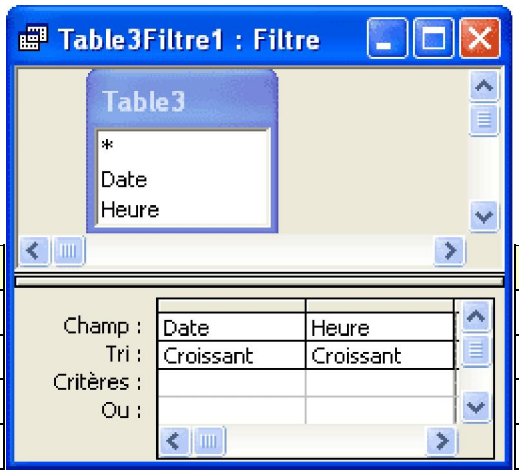
Le tri que nous avons considéré précédemment ne concerne qu'une seule colonne, c'est un tri simple. Dans certains cas, nous avons besoin d'effectuer un tri sur plusieurs colonnes, encore appelé tri multiple. Le **filtre/tri** est l'outil qui nous permet d'arriver à nos fins sans avoir besoin de créer 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
02/03/2003	16:21:36
03/03/2003	10:00:25
02/03/2003	12:23:18
01/03/2003	13:40:01



Date	Heure
01/03/2003	13:40:01
02/03/2003	12:23:18
02/03/2003	16:21:36
03/03/2003	10:00:25

Table "Table3" initiale — Filtre/tri — Table "Table3" finale

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.

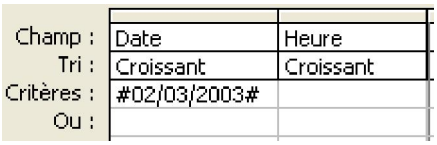
La ligne "Critères :" permet, comme dans une requête, de sélectionner les enregistrements à afficher sur des critères plus ou moins complexes. Les techniques correspondantes sont exposées dans les chapitres relatifs aux requêtes.

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.

9.7 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.



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 requête". 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.

9.8 Conclusion

Ce chapitre traite de techniques modestes, mais qui ont leur utilité. La fonction "Rechercher" permet de vérifier si une donnée figure ou non dans une table. Dans la négative, il faut tout de suite vérifier que l'on ne s'est pas trompé de colonne, ni de place dans le champ...

Tout utilisateur de SGBD qui manipule des tables se sert du tri à tour de bras. Le tri par sélection est une technique très simple qui rend bien des services. On l'utilise souvent couplée à la fonction "Rechercher". Cette dernière permet de trouver la première occurrence d'une chaîne donnée, avant d'appliquer le tri par sélection qui fournit immédiatement les autres.

Le filtre/tri, enfin, mérite le détour, car il permet de présenter une table triée comme on a envie de la voir et de l'utiliser.

10 Introduction aux requêtes

10.1 Préambule

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 BDD 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 BDD après la table.

10.2 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 BDD. 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 BDD. 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 BDD. 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.

10.3 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. Quatre chapitres (18-21) sont consacrés à la création des divers types de requêtes.

Dans le SGBD Access, la création d'une requête union n'est possible qu'en SQL. Nous évoquerons ce point au chapitre 20. On notera par ailleurs que deux opérations ensemblistes (intersection et différence) ne sont pas implémentées dans Access.

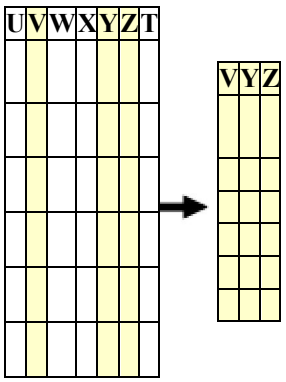
10.4 Conclusion

La recherche d'information est , à notre humble avis, l'aspect le plus intéressant -- pour ne pas dire le plus passionnant -- de l'étude des bases de données. En enchaînant astucieusement un petit nombre de requêtes bien choisies, on peut souvent faire des merveilles dans la recherche d'information, sans avoir besoin de recourir aux outils plus compliqués (et combien plus onéreux !) du "data mining", ou autres techniques à la mode.

10.5 Introduction

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.

10.6 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	prénom	nom_organisme	fonction
Turlutu	Jean	Chose et Cie	technicien
Surpont	Yvette	TAGADA	secrétaire
Lechant	Paul	Société Machin	directeur
Durand	Nathalie	Entreprise Truc	ingénieur
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 BDD 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 BDD. 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.

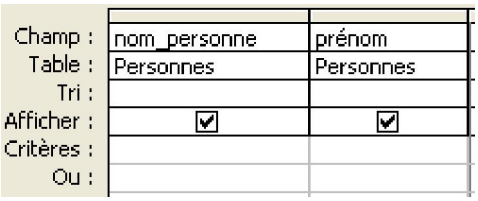
La moitié basse contient la grille de définition de la requête. Pour y introduire un champ (on notera au passage que l'astérisque représente la totalité des champs), nous disposons de trois méthodes :

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 :



Champ :	nom_personne	prénom
Table :	Personnes	Personnes
Tri :		
Afficher :	☑	☑
Critères :		
Ou :		

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

nom_personne	prénom
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").

10.7 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.

10.8 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.

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.

nom_personne	prénom
Durand	Nathalie
Lechant	Paul
Lechant	Paul
Surpont	Yvette
Turlutu	Jean
Verseau	Pierre

10.9 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.

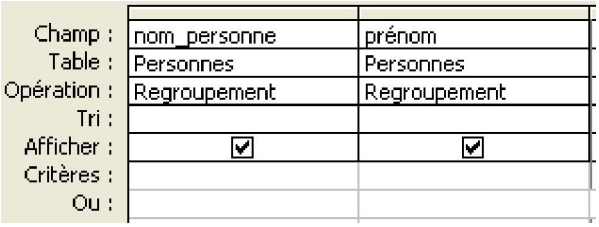
Première méthode. Ouvrons la requête "Requête1" en mode création. Effectuons un clic droit dans la fenêtre de définition de la requête et sélectionnons "**Propriétés**" dans la liste déroulante, ou cliquons sur l'icône "Propriétés". La boite de dialogue "Propriétés de la requête" s'ouvre. Modifions la 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	prénom
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.

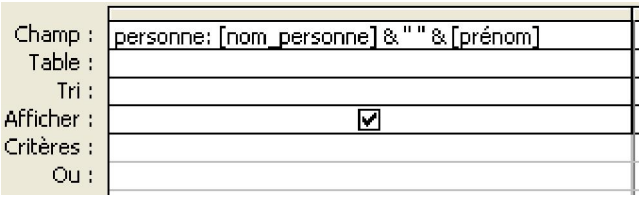


Basculons ensuite en mode "feuille de données" : les doublons ont disparu et la feuille de données est triée par ordre alphabétique croissant.

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

10.10 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, comme en Visual Basic, 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 BDD 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.

10.11 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.

10.12 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 :

Champ :	personne: [nom_personne] & " " & [prénom]
Table :	
Opération :	Regroupement
Tri :	
Afficher :	☑
Critères :	
Ou :	

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.

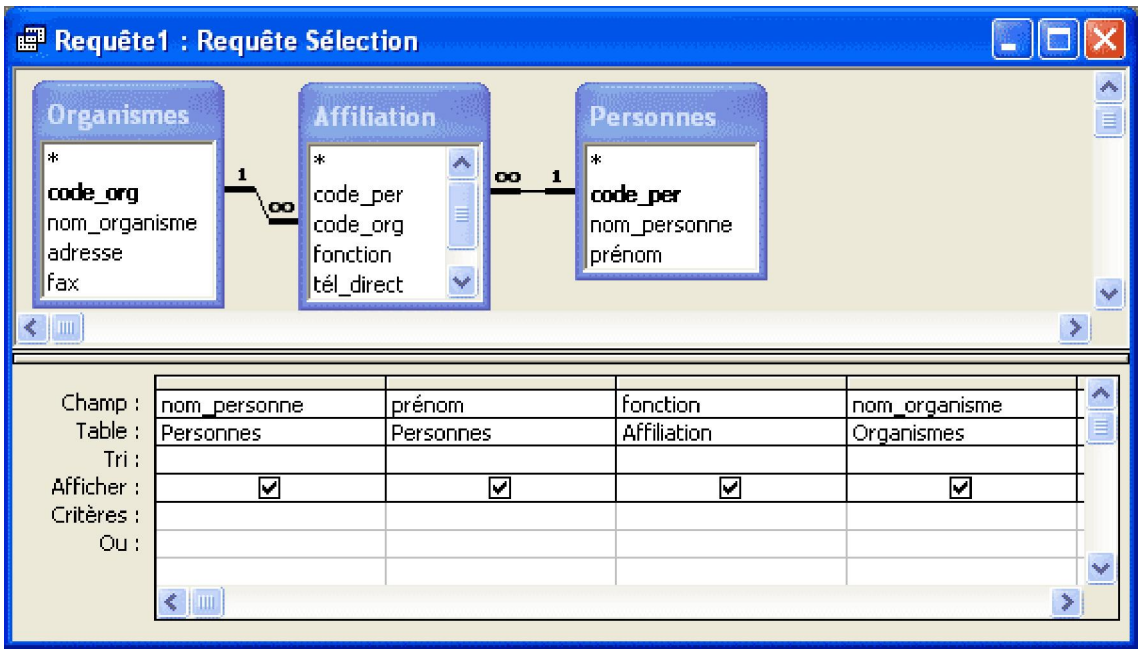
10.13 La requête multi-table

Dans une BDD 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.

10.14 Conclusion

Les opérations de sélection simple (ou projection) s'effectuent sur les colonnes des tables. Les lignes ne sont pas modifiées.

Les opérations de sélection simple (ou projection) sont fort utiles lorsqu'on désire :

obtenir une vue simplifiée d'une table, en ne conservant que les colonnes contenant les informations désirées ;

rassembler des informations dispersées parmi plusieurs tables liées par des relations. On crée ainsi une vue partielle ou globale du contenu de la base de données.

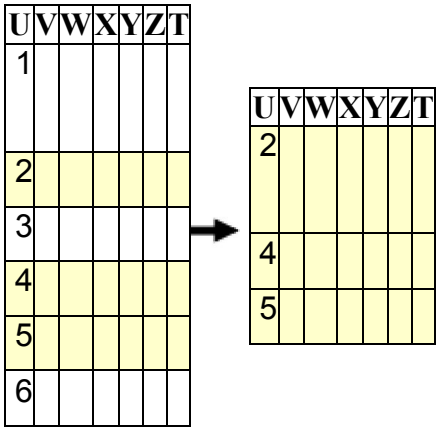
11 La requête de sélection

11.1 Introduction

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 BDD, un rôle très important.

11.2 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 11 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 1.000 € le mois dernier. Dans une des tables de notre BDD 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 1.000 €, ce qui met en jeu l'**opérateur de comparaison** "supérieur à".

11.3 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.

11.4 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.

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 par le SGBD Access. 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 (_). Une annexe traitera de l'usage des caractères génériques plus en détail.

11.5 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) :

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

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

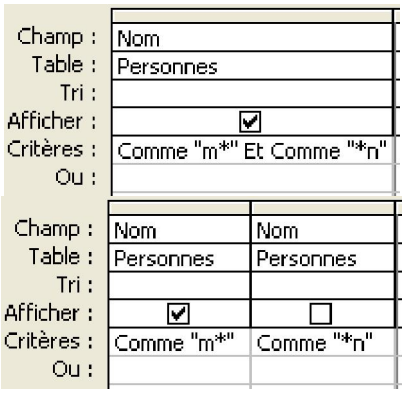
Nous voyons que l'opérateur OU peut être écrit explicitement (à droite), ou traduit graphiquement (à gauche). Le résultat de la requête est, bien entendu, le même dans les deux cas.

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 !

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

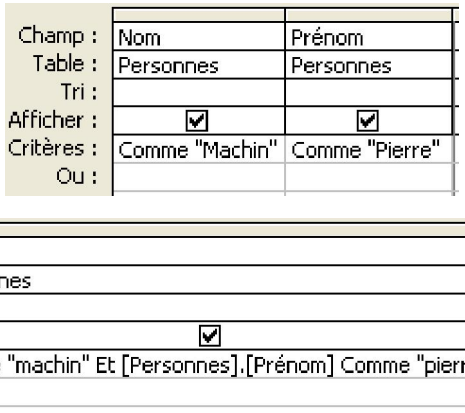
Champ :	Nom	Prénom
Table :	Personnes	Personnes
Tri :		
Afficher :	☑	☑
Critères :	Comme "Machin" Ou [Personnes].[Prénom] Comme "Jacques"	

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".



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.



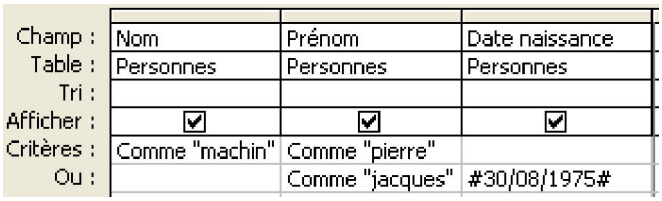
Conclusion : dans la grille de création d'une requête,

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.



La requête fonctionne selon l'expression ensembliste suivante :

((Personnes.Nom Comme "machin") ET (Personnes.Prénom Comme "pierre")) OU
((Personnes.Prénom Comme "jacques") ET (Personnes.[Date naissance]=#8/30/1975#))

c'est à dire 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.

11.6 Les opérateurs de comparaison

Les opérateurs de comparaison arithmétiques :

= (égal), **<** (inférieur), **<=** (inférieur ou égal), **>** (supérieur), **>=** (supérieur ou égal), **<>** (différent)

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.

11.7 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) issues de VBA 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.

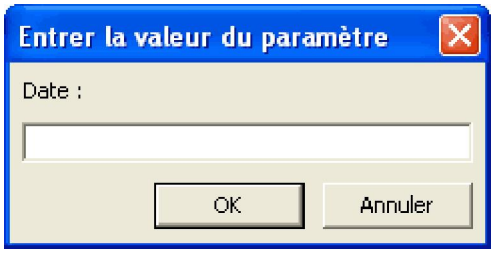
Attention ! Notez bien que, lorsqu'une fonction possède plusieurs arguments, le caractère séparateur est le point-virgule, alors que c'est la virgule dans la version anglophone d'Access. C'est un détail de syntaxe ridicule, mais il est à l'origine de bien des mauvaises surprises.

11.8 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 :

11.9 Conclusion

Les SGBD mettent à la disposition des utilisateurs des outils extrêmement variés pour exprimer les critères utilisés dans l'élaboration d'une requête. En emboîtant, si nécessaire, plusieurs requêtes, les utilisateurs peuvent extraire des BDD toutes les informations qu'ils désirent -- ou presque. Il est fort rare que l'on se trouve dans l'impossibilité réelle de transcrire un critère donné.

Pour qui fait l'effort d'apprendre à créer des requêtes, et obtient de son entreprise l'autorisation de s'en servir, il y a là une mine d'or... au sens de l'information tout au moins.

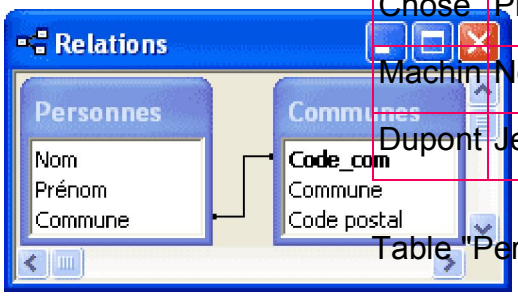
12 La notion de jointure

12.1 Introduction

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 au chapitre 4. 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.

Les deux tables sont liées par une relation, assurée via un code masqué. Cette relation apparaît dans la fenêtre "Relations", comme le montre la figure ci-dessous.



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
SMH	38402

Table "Communes"

12.2 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.

12.3 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.

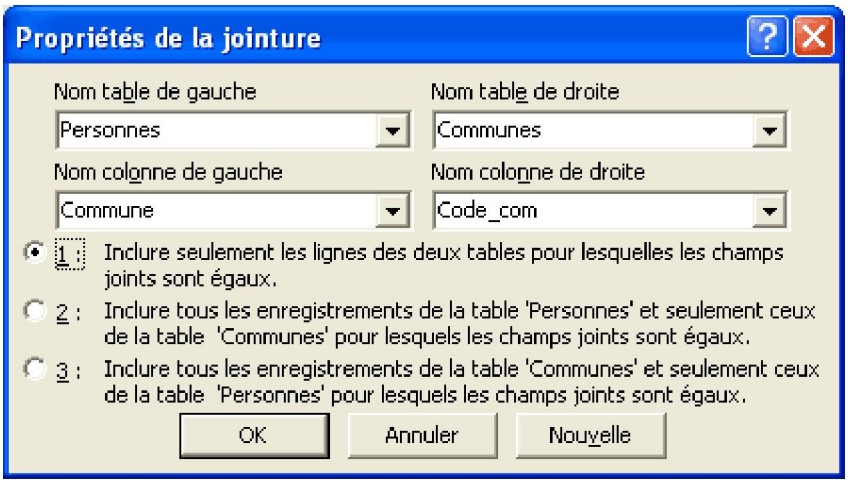
12.4 La jointure interne

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) et SMH, auxquelles ne correspond aucune personne, sont également absente. Le SGBD a traité la relation entre les deux tables comme une **jointure interne**.

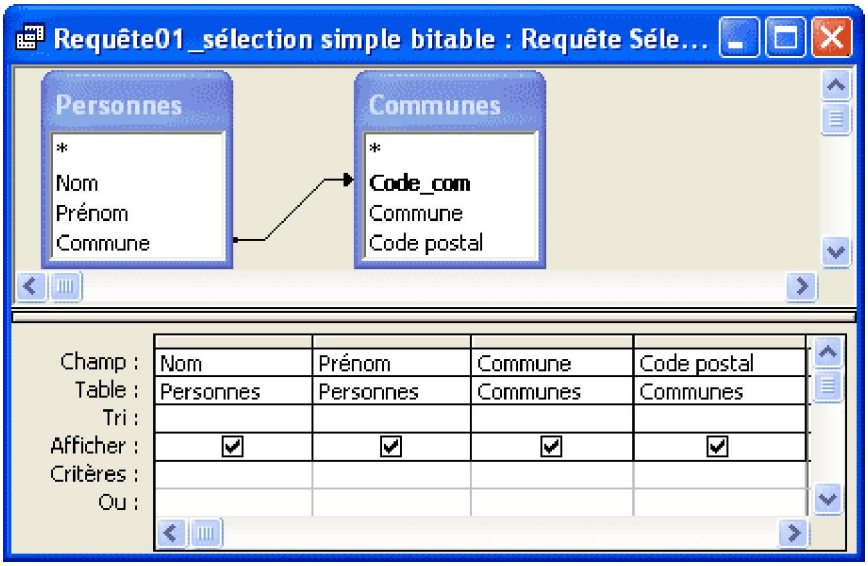
Effectuons un clic droit sur la relation, et sélectionnons "Propriétés de la jointure". La fenêtre du même nom s'affiche ; elle se présente comme le montre la figure ci-dessous. Bien que le terme ne soit pas présent, l'option 1 de la fenêtre correspond effectivement à la jointure interne.



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.

12.5 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.

12.6 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
		SMH	38402

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.

12.7 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 :

Prénom
Paul
Jean
Marie
Henri
Claude

"Prénoms1"

Prénom
Henri
Patrick
Paul

"Prénoms2"

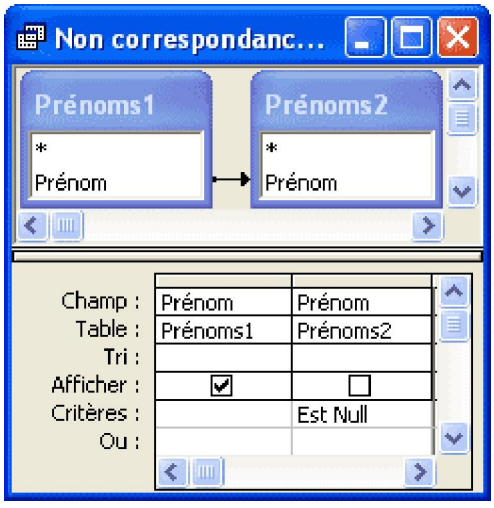
Nous recherchons les prénoms de la première table qui sont absents de la seconde. Pour ce faire, nous devons passer en revue tous les prénoms de la première table, et regarder s'ils sont ou non dans la seconde. Pour créer la requête correspondante, nous songeons donc à utiliser une jointure gauche.

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 :

Prénom1	Prénom2
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 :

Prénom
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 :

Prénom1	Prénom2
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 :

Prénom1	Prénom2
Paul	Paul
Henri	Henri
	Patrick

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.

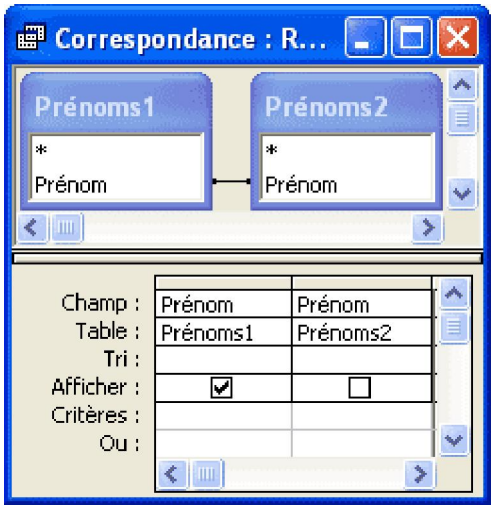
Si le raisonnement relatif à la requête de non correspondance vous paraît ardu, ne vous inquiétez pas : vous n'êtes pas le seul. C'est la raison pour laquelle Microsoft a créé un assistant pour ce type de recherche. Vous le trouverez dans la fenêtre "Base de données", l'objet "Requêtes" étant sélectionné.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

12.8 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 :

Prénom
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 d'Access 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.

12.9 Conclusion

La notion de jointure joue un rôle important dans les requêtes multi-tables, en particulier dans les requêtes de correspondance et de non correspondance.

Si l'opérateur ne précise pas le type de jointure, c'est la jointure interne que le SGBD applique par défaut.

13 La requête de regroupement

13.1 Introduction

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.

13.2 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
06/01/2003	Nord	927,02 €
06/01/2003	Sud	1 098,46 €
06/01/2003	Est	561,29 €
07/01/2003	Nord	1 385,55 €
07/01/2003	Est	681,09 €
07/01/2003	Sud	1 401,56 €

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 au minimum deux colonnes :

une sur laquelle s'effectue le **regroupement** (elle doit contenir des doublons) ;

une 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).

13.3 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.

Première étape. 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").

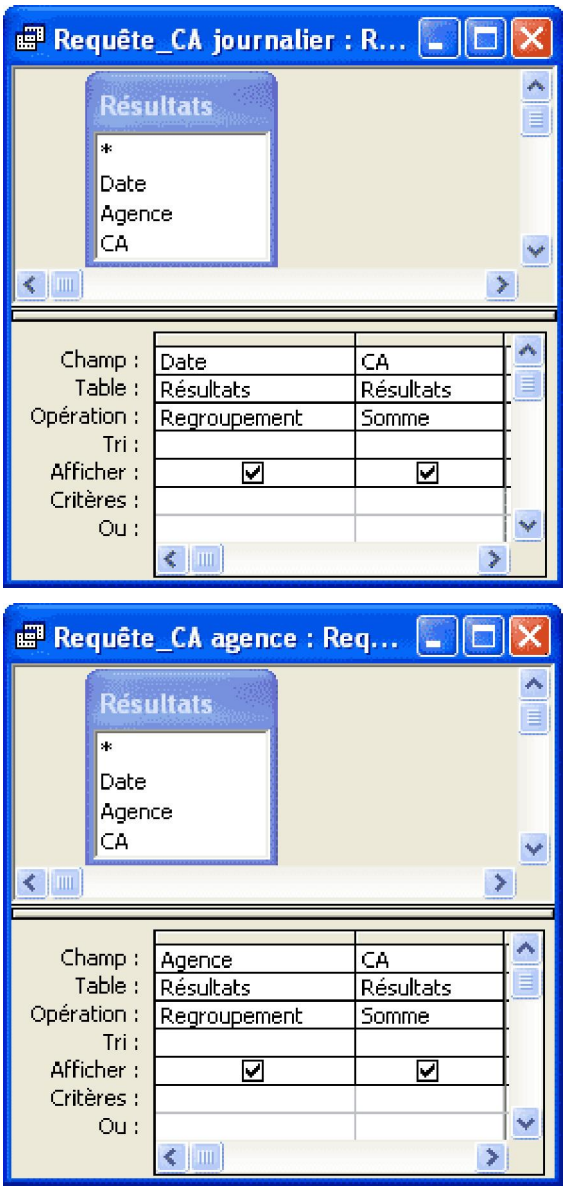
Seconde étape. Elle consiste à introduire dans la grille le champ sur lequel s'effectue le regroupement. Comme nous cherchons à calculer des CA quotidiens, ce champ ne peut être que la date. Nous introduisons donc le champ "Date" dans la grille de création de la requête.

Troisième étape. 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.

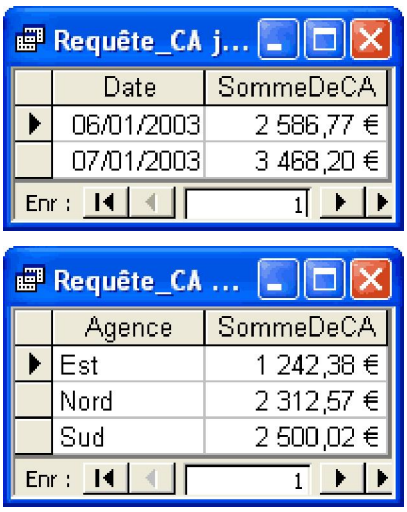
Quatrième étape. Il faut maintenant introduire le champ sur lequel s'effectue l'opération liée au regroupement. Dans le présent exemple, l'opération consiste à sommer les CA de chaque agence. Nous introduisons donc le champ "CA" dans la grille.

Cinquième étape. 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 (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".

13.4 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
06/01/2003	862,26 €
07/01/2003	1 156,07 €

Agence	MoyenneDeCA
Est	621,19 €
Nord	1 156,29 €
Sud	1 250,01 €

Date	MinDeCA
06/01/2003	561,29 €
07/01/2003	681,09 €

Agence	MinDeCA
Est	561,29 €
Nord	927,02 €

		Sud	1 098,46 €

Date	MaxDeCA	Agence	MaxDeCA
06/01/2003	1 098,46 €	Est	681,09 €
07/01/2003	1 401,56 €	Nord	1 385,55 €
		Sud	1 401,56 €

Date	PremierDeCA	Agence	PremierDeCA
06/01/2003	927,02 €	Est	561,29 €
07/01/2003	1 385,55 €	Nord	927,02 €
		Sud	1 098,46 €

Date	DernierDeCA	Agence	DernierDeCA
06/01/2003	561,29 €	Est	681,09 €
07/01/2003	1 401,56 €	Nord	1 385,55 €
		Sud	1 401,56 €

13.5 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 euros. 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 :

<#08/01/2003#

limitera le calcul du chiffre d'affaire cumulé par date aux jours précédant le 8 janvier 2003.

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 ;

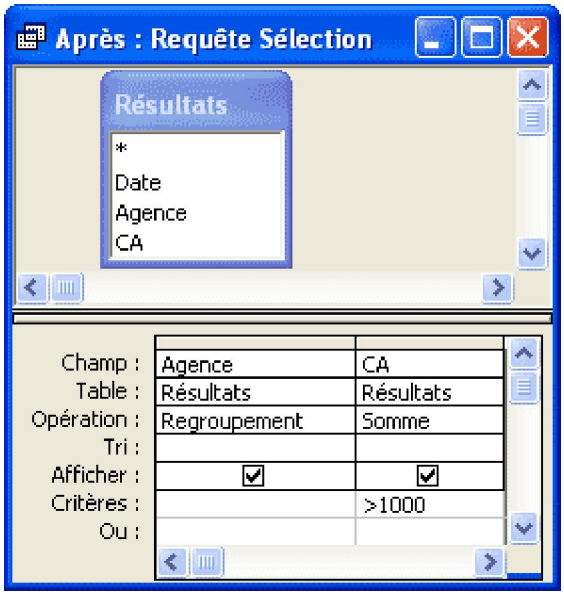
nous introduisons le(s) critère(s) de filtrage sur la ligne "Critères :".

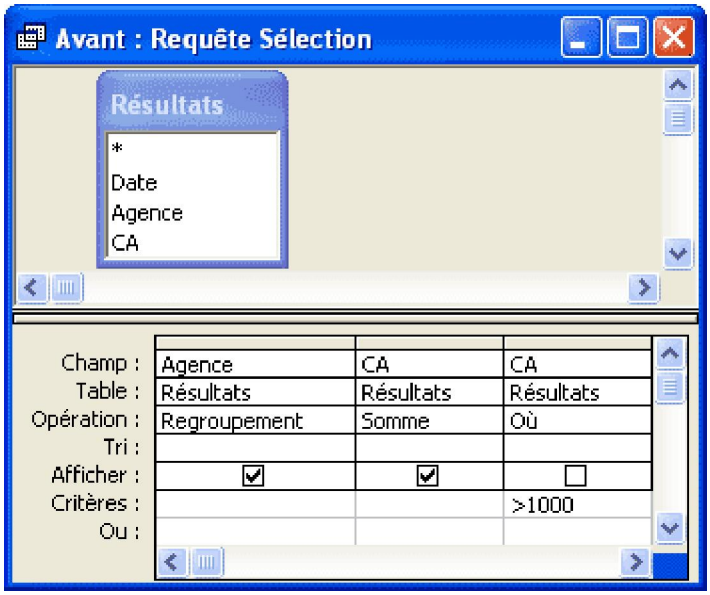
Avant de filtrer une requête de regroupement, il faut bien réfléchir au résultat que l'on veut obtenir. Car un même critère, appliqué au même champ, ne donnera pas le même résultat suivant qu'il opère avant ou après le regroupement. Pour le montrer, nous avons créé deux exemples dans lesquels le filtre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

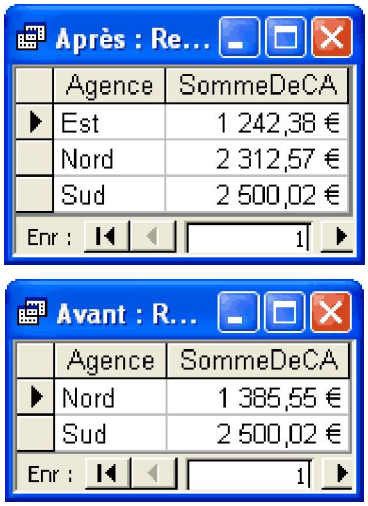
Premier exemple (figures de gauche). Nous appliquons le filtre aux CA regroupés par agence, c'est à dire que nous ne les affichons que s'ils dépassent 1000 euros ;

Deuxième exemple (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 !

13.6 Conclusion

La requête de regroupement est un outil fort utile. Mais, pour l'utiliser correctement, il faut prendre les précautions suivantes :

il faut appliquer le regroupement à un seul champ, ou à un très petit nombre de champs, sinon il n'y a plus de regroupement possible et la requête, bien évidemment, ne regroupe plus rien ;

il faut, avant d'appliquer un filtre, décider s'il doit agir avant ou après le regroupement. Le résultat, en effet, ne sera pas le même dans les deux cas, sauf si le filtre est appliqué au champ de regroupement.

Le regroupement avec comptage, et son application au traitement des doublons, sont abordés au chapitre suivant.

14 Le comptage et l'élimination des doublons

14.1 Introduction

Nous poursuivons notre étude du regroupement (commencée au chapitre précédent) dans le cas particulier où la fonction utilisée est "Compte". Cette fonction n'effectue pas d'opération ; comme son nom l'indique, elle fournit le nombre d'enregistrements regroupés, dans la colonne où on lui demande d'opérer.

Pour créer des exemples de requête, nous utilisons la table du chapitre précédent, dans laquelle nous avons supprimé une donnée (le chiffre d'affaires de l'agence Est pour la journée du 7 janvier 2003 n'a pas été communiqué au siège de l'entreprise -- quelqu'un va se faire tirer les oreilles). La table est représentée ci-dessous.

Date	Agence	CA
06/01/2003	Nord	927,02 €
06/01/2003	Sud	1 098,46 €
06/01/2003	Est	561,29 €
07/01/2003	Nord	1 385,55 €
07/01/2003	Est	
07/01/2003	Sud	1 401,56 €

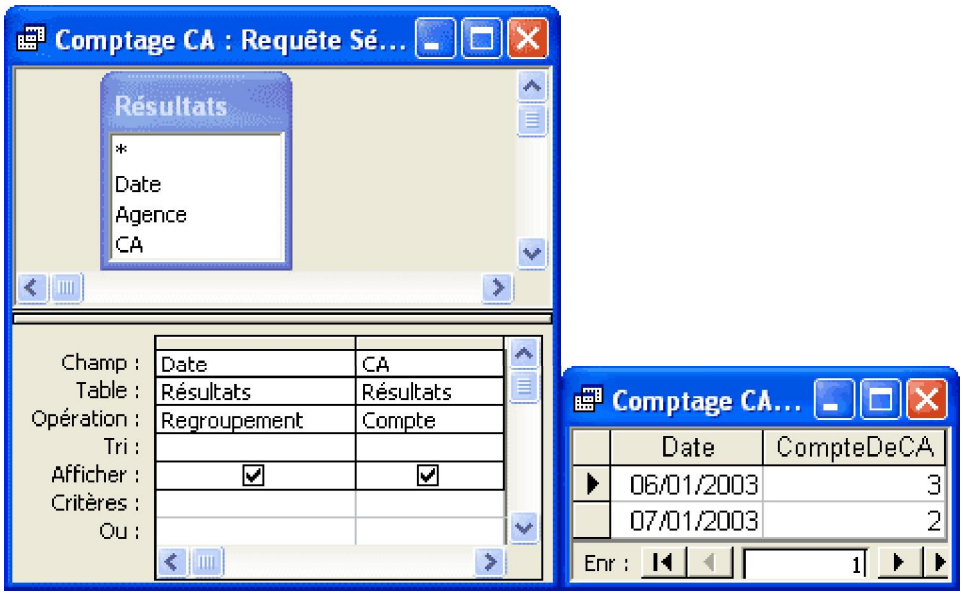
14.2 Le regroupement avec comptage

Nous créons une requête dans laquelle :

le regroupement est effectué sur le champ "Date" ;

l'opération de comptage concerne le champ "CA".

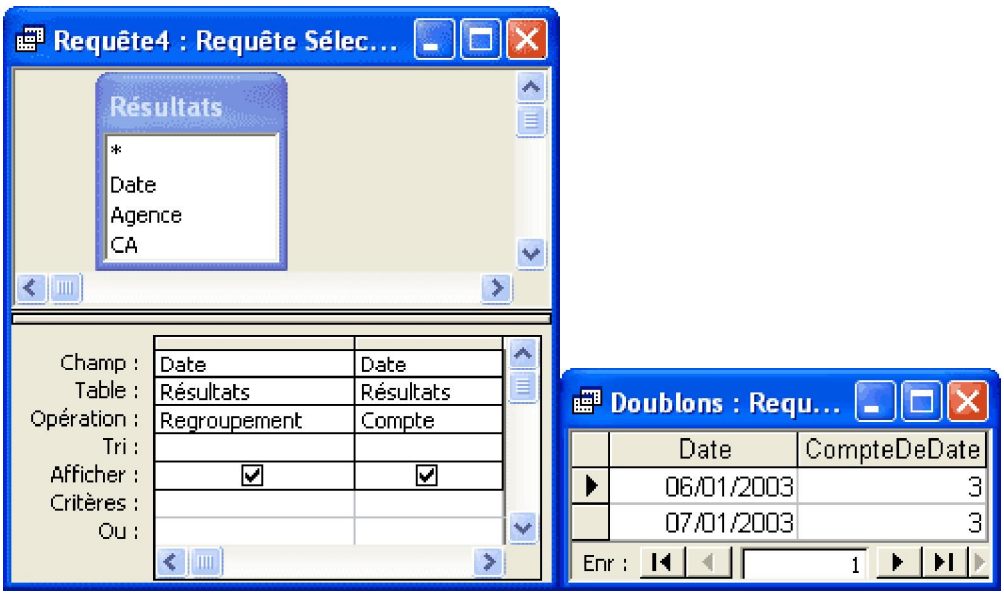
Les deux figures ci-dessous représentent la requête de regroupement (à gauche), et la feuille de données résultante (à droite).



Nous remarquons immédiatement que la fonction "Compte" ne prend pas en compte les enregistrements vides. Il est d'usage de dire que la fonction "Compte" ignore les "Null".

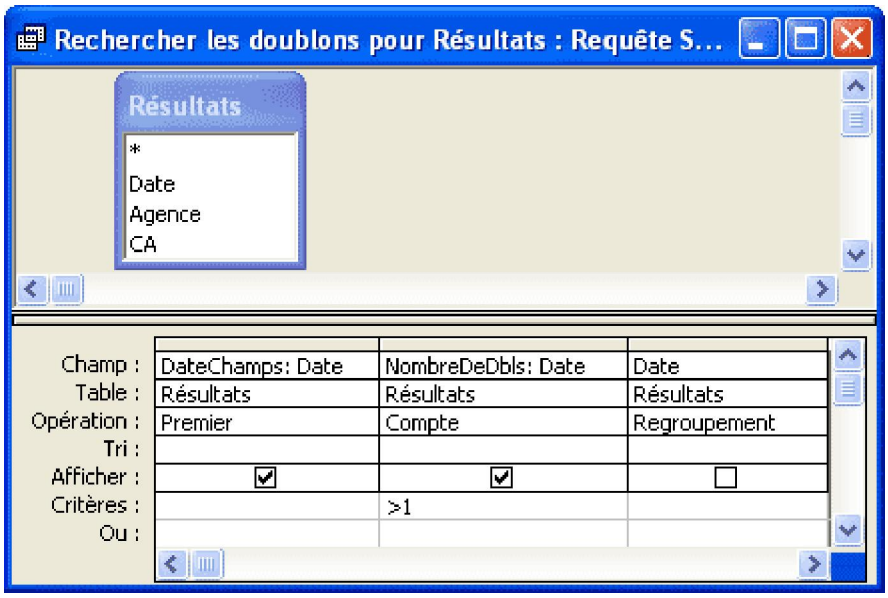
14.3 Le comptage des doublons

Nous pouvons appliquer la fonction "Compte" au champ de regroupement lui-même. Dans le cas du champ "Date", les deux figures ci-dessous représentent la requête de regroupement (à gauche), et la feuille de données résultante (à droite).



Nous disposons ainsi d'un moyen de compter les doublons -- à condition d'admettre qu'une valeur qui n'apparaît qu'une seule fois (et sera comptée comme telle) fait partie des doublons. Si nous prenons le terme "doublon" au sens strict, nous devons rajouter, dans la colonne de comptage, sur la ligne "Critères :", un filtre qui ne conserve que les valeurs supérieures à l'unité (>1). Nous reconnaissons au passage un exemple de filtrage après regroupement.

Le SGBD Access possède un assistant de comptage des doublons, et il est instructif de regarder comment il fonctionne. Dans la fenêtre "Base de données", nous sélectionnons l'objet "Requêtes", nous cliquons sur l'icône "Nouveau", nous sélectionnons "Assistant Requête trouver les doublons", et nous cliquons sur "OK". Nous demandons le comptage des doublons sur le seul champ "Date", et nous observons la manière dont la requête est formulée (figure ci-dessous).

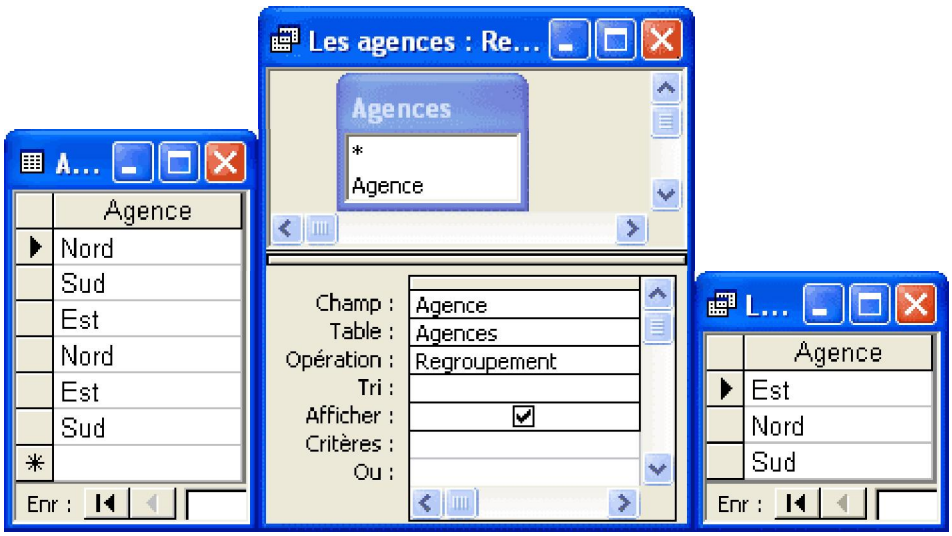


Surprise ! Le regroupement et le comptage sont bien effectués sur le même champ "Date", mais le regroupement et l'affichage du champ sont disjoints. Cette complication, qui apparaît comme inutile, trouverait-elle sa source dans le désir d'assurer la compatibilité avec une ancienne version d'Access, ou avec d'autres SGBD ? L'auteur de ces lignes donne sa langue au chat.

14.4 L'élimination des doublons

De la table qui nous sert d'exemple, extrayons une nouvelle table ne contenant que la colonne "Agence". Cette table contient des doublons, puisque toutes les agences y sont citées deux fois. Il suffit d'opérer une sélection avec regroupement, mais sans comptage des doublons, pour éliminer les doublons de la table. Le regroupement est une bonne technique de dédoublonnage, que nous avons déjà utilisée au chapitre 11, pour un exemple de doublons sur deux champs.

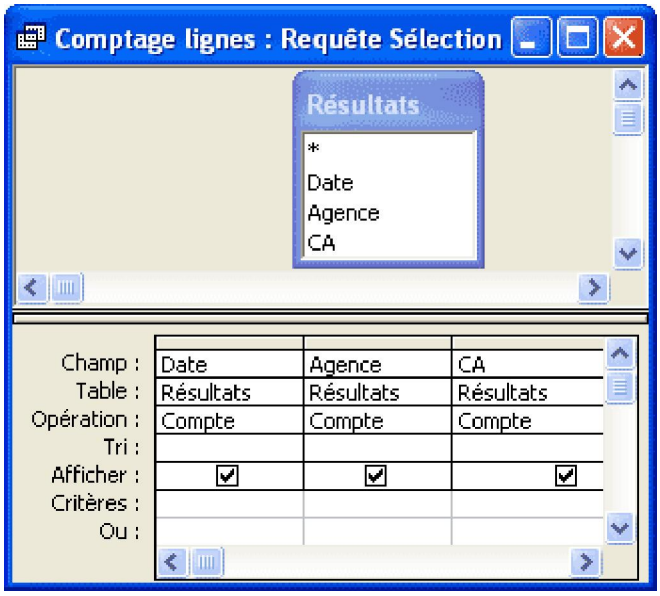
La table de départ, la requête et son résultat sont représentés ci-dessous.

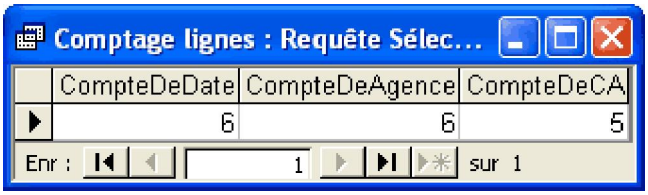


Soyons concis : pour dédoublonner, il suffit de regrouper sans compter.

14.5 Le comptage sans regroupement

En faisant exactement l'inverse, c'est à dire en comptant sans regrouper, nous déterminons le nombre d'enregistrements (non Null) présents dans une ou plusieurs colonnes. La figure ci-dessous représente la requête correspondante (en haut) et la feuille de données résultante (en bas). Au passage, nous vérifions que, dans la colonne CA, la fonction "Compte" a négligé la case vide.





Nous pouvons ainsi déterminer le nombre d'enregistrements d'une table, à condition que nous soyons sûrs que la colonne utilisée pour l'opération ne contient pas de case vide. Nous pouvons songer à utiliser une colonne dans laquelle le Null est interdit, mais il n'en n'existe pas toujours. Pour nous affranchir du comptage sur une colonne particulière, nous procédons comme suit :

nous ajoutons la table "Résultats" dans la fenêtre de création de la requête, mais nous n'introduisons aucun champ dans la grille ;

sur la ligne "Champ :", nous inscrivons le nom de la colonne dans laquelle figurera le résultat du comptage, soit par exemple "Comptage" ;

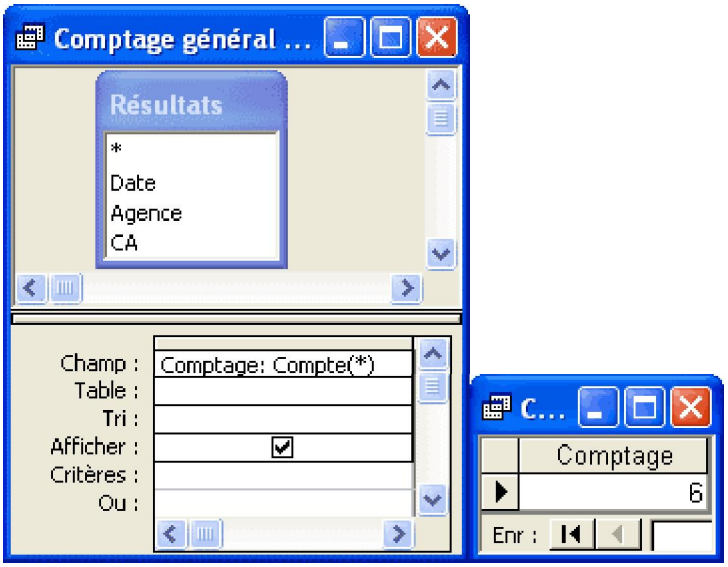
nous faisons suivre ce nom de deux points et d'un espace. Le SGBD sait alors que la suite définit le contenu du champ ;

à la suite, nous écrivons la fonction Compte(*). L'astérisque indique au SGBD qu'il doit compter les lignes de la table, sans utiliser de colonne particulière ;

sur la ligne "Table :", nous n'inscrivons rien. En effet, la colonne que nous allons créer n'appartient à aucune table. Par contre, le SGBD sait que la fonction "Compte(*)" s'applique à la table "Résultats", qu'il était donc indispensable d'introduire dans la fenêtre.

Le principe de l'opération est simple. Nous créons une requête sélection sans introduire aucun des champs de la table sélectionnée. Nous demandons à cette requête de créer un nouveau champ, auquel nous donnons un nom (indispensable pour l'affichage). Nous définissons le contenu de ce nouveau champ à l'aide la fonction "Compte()" qui compte des lignes. Nous utilisons l'astérisque comme argument, de telle sorte que la fonction compte les lignes de la table. La requête affiche une feuille de données possédant une seule colonne (puisque nous l'avons définie ainsi), et une seule ligne (puisque la fonction fournit une valeur unique).

Les figures ci-dessous représentent la requête (à gauche) et la feuille de données résultante (à droite).



Cette méthode compte effectivement les lignes d'une table, même si elles sont toutes vides. Vous pouvez faire l'expérience en effaçant tout ce que contient la table "Résultats" (après en avoir gardé copie).

Remarque : si nous remplaçons l'astérisque par le nom de l'un des champs de la table "Résultats" (mis entre crochets pour respecter la syntaxe), nous obtenons le résultat du paragraphe précédent, c'est à dire le comptage des enregistrements (non vides) du champ considéré.

14.6 Conclusion

Comme vous pouvez le constater, cette page est pleine de ressources. Elle vous montre comment :

compter les enregistrements regroupés ;

compter les doublons ;

éliminer les doublons ;

compter les enregistrements (non vides) dans une colonne ;

compter les lignes dans une table.

Le comptage des doublons, et leur élimination, sont des techniques importantes à connaître, car elles sont souvent utilisées pour la maintenance des bases de données et l'interrogation de leur contenu.

15 Les requêtes ajout et analyse croisée

15.1 Introduction

Nous avons consacré 5 chapitres à la requête de sélection et à ses divers développements. Nous avons d'abord étudiée sa forme élémentaire (la sélection simple), puis sa forme générale (la sélection avec critères). Nous avons ensuite découvert la notion de jointure, qui s'introduit naturellement lorsque la sélection porte sur plusieurs tables, et de là nous sommes passés à la correspondance et à la non-correspondance. Nous avons enfin perfectionné la sélection grâce à la notion de regroupement, ce qui nous a permis d'effectuer des synthèse, et de manipuler les doublons. Bref, comme nous pouvons le constater, la sélection est la reine des requêtes !

Cependant, la sélection ne peut pas tout faire, et sous la pression des besoins, d'autres types de requête ont été créés. Nous en avons rassemblé deux dans ce chapitre (l'ajout et l'analyse croisée), et deux dans le chapitre suivant (la suppression et la mise à jour). La requête analyse croisée est une spécificité d'Access, et on ne la retrouve généralement pas dans les autres SGBD. Pour les afficionados du SQL, la requête ajout n'est pas connue sous ce nom ; elle est simplement considérée comme un cas particulier d'utilisation de la commande INSERT.

Il reste une grande absente, la requête union, qu'on ne peut pas créer dans la fenêtre graphique d'Access, mais que nous traiterons lorsque nous étudierons le langage SQL.

15.2 Le fonctionnement de la requête ajout

La requête ajout permet d'insérer les enregistrements d'une table n° 2 dans une table n° 1. L'opération ne peut se faire que si les deux tables ont au moins un champ commun (même nom, même type de données ou conversion de type possible -- cela dépend du SGBD). Comme le montre la figure ci-dessous, les champs de la table n° 2 qui ne sont pas communs avec ceux de la table n° 1 sont ignorés ou refusés (ex : le champ "T"). Les champs de la table n° 1 qui n'existent pas dans la table n° 2 ne sont pas renseignés (ex : le champ "W") -- à moins que le champ ne soit du type NuméroAuto, auquel cas le système le remplira lui-même, comme nous le constaterons dans un prochain exemple.

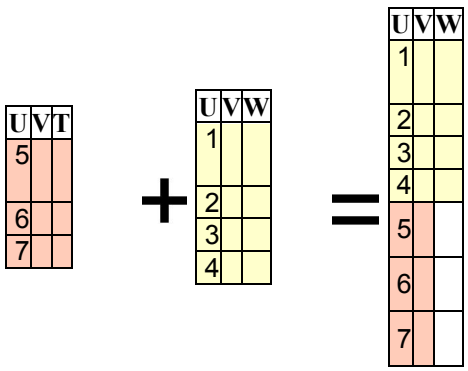
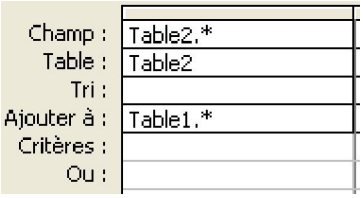


Table n°2 Table n°1 Résultat

Attention ! la requête ajout modifie irréversiblement la table à laquelle on ajoute des données (la table n° 1 dans la terminologie du paragraphe ci-dessus). L'opération une fois effectuée, il n'est plus possible de revenir en arrière. Il est donc très fortement recommandé de créer une copie de la table n° 1 avant de procéder à l'ajout. La table que l'on ajoute (la table n° 2) n'est ni modifiée, ni supprimée, au cours de l'opération.

Pour créer une requête ajout dans le SGBD Access, nous introduisons la table à ajouter (la table n° 2 dans notre terminologie) dans la fenêtre de création/modification d'une requête, et nous

sélectionnons les champs que nous voulons -- ou que nous pouvons -- ajouter. Puis nous cliquons sur la petite flèche qui borde l'icône "Type de requête" et, dans la liste déroulante qui s'affiche, nous choisissons "Requête Ajout...". Dans la boite de dialogue "Ajout" qui s'ouvre, nous précisons quelle est la table dans laquelle doit s'effectuer l'ajout (la table n° 1 dans notre terminologie). La grille de définition de la requête acquiert alors une ligne supplémentaire intitulée "Ajouter à :", comme le montre la figure ci-dessous.



Les données de la table n° 2 seront effectivement ajoutées à la table n° 1 lorsque nous exécuterons la requête. Des messages nous avertiront de ce qui se passera -- à moins que nous n'en ayons décidé autrement dans les options (onglet "Modifier/Rechercher", cadre "Confirmer").

Diverses sophistications sont possibles. Nous pouvons :

sélectionner une partie seulement des champs de la table n° 2 ;

sélectionner à l'aide de critères les enregistrements de la table n° 2 qui doivent être ajoutés à la table n° 1 ;

remplacer la table n° 2 par une requête ;

faire en sorte qu'une requête (mono ou multi-table) effectue également un ajout dans une autre table.

Il nous faut cependant bien veiller à ce que les colonnes qui sont utilisées pour définir les opérations de sélection, mais qui ne sont pas concernées par l'ajout, ne contiennent aucune information sur la ligne "Ajouter à :", sinon le SGBD Access nous gratifiera d'un message d'erreur qui nous plongera dans des abîmes de réflexion (exemple à méditer : "Destination de sortie 'requête' répliquée").

Voici une liste non limitative des diverses utilisations de la requête ajout :

rassembler dans une même table des enregistrements provenant de tables séparées. Dans cette application, la requête ajout entre en concurrence avec la requête union, que nous étudierons dans l'un des chapitres consacrés au SQL (chapitre 22). Attention : les deux requêtes n'imposent pas les mêmes contraintes, et ne fournissent pas forcément le même résultat (problème des doublons) ;

imposer à une table des propriétés particulières, en l'ajoutant à une table modèle,

initialement vide et dont les propriétés sont soigneusement définies (largeur et visibilité des colonnes, tri, police de caractères, etc.) ;

garder trace d'un classement dans une table ;

etc.

Dans le paragraphe suivant, nous examinerons quelques exemples d'utilisation de la requête ajout.

15.3 L'utilisation de la requête ajout (exemples)

Notre **premier exemple** illustre simplement la procédure exposée ci-dessus. La figure suivante représente le contenu des deux tables avant et après l'ajout.

Nom	Prénom
Durand	Oscar
Lechant	Anne

Nom	Prénom	Date
Machin	Pierre	12/6/1983
Truc	Nathalie	26/11/1985
Chose	André	5/2/1980

Nom	Prénom	Date
Machin	Pierre	12/6/1983
Truc	Nathalie	26/11/1985
Chose	André	5/2/1980
Durand	Oscar	

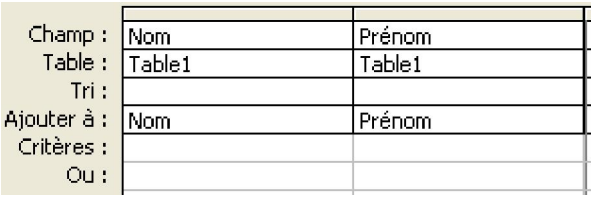
Table n° 2 (avant & après)

Table n° 1 (avant)

Lechant	Anne	

Table n° 1 (après ajout de la table n° 2)

Attention ! Le résultat de l'ajout dépend de l'ordre dans lequel on effectue les opérations. Si nous permutons les rôles des tables 1 et 2, nous redéfinissons notre requête ajout comme suit :



Champ :	Nom	Prénom
Table :	Table1	Table1
Tri :		
Ajouter à :	Nom	Prénom
Critères :		
Ou :		

car le SGBD Access n'acceptera pas que nous tentions d'introduire dans une table des champs qui n'y sont pas initialement présents. Nous notons que seuls les noms des champs de la table n° 2 figurent dans la grille, mais nous pourrions écrire « Table2.Nom » et « Table2.Prénom » à la place « Nom » et de « Prénom », sans que le système ne proteste.

La figure ci-dessous représente le contenu des deux tables avant et après l'ajout.

Nom	Prénom	Date
Machin	Pierre	12/6/1983
Truc	Nathalie	26/11/1985
Chose	André	5/2/1980

Table n° 1 (avant & après)

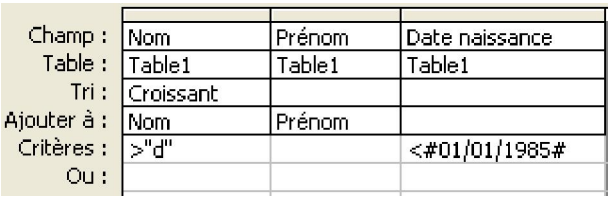
Nom	Prénom
Durand	Oscar
Lechant	Anne

Table n° 2 (avant)

Nom	Prénom
Machin	Pierre
Truc	Nathalie
Chose	André
Durand	Oscar
Lechant	Anne

Table n° 2 (après ajout table n° 1)

Notre **second exemple**, directement inspiré de la fin du précédent, illustre l'introduction de critères aux enregistrements de la table que l'on ajoute à l'autre. Voici comment se présente la grille de la requête, si l'on impose des critères à deux champs de la table n° 1 avant de l'ajouter à la table n° 2 (pour faire bonne mesure nous avons également ajouté un tri) :



Champ :	Nom	Prénom	Date naissance
Table :	Table1	Table1	Table1
Tri :	Croissant		
Ajouter à :	Nom	Prénom	
Critères :	>"d"		<#01/01/1985#
Ou :			

Et voici le résultat :

Nom	Prénom
Durand	Oscar
Lechant	Anne
Machin	Pierre

Notre **troisième exemple** montre comment on peut modifier les propriétés d'une table en l'ajoutant à une table vide. La table n° 1 que nous avons utilisée ci-dessus comporte un champ « Date », pour lequel nous avons choisi le format jj/mm/aaaa, appelé « Date, abrégé » en mode création. Nous

voulons maintenant obtenir la date dans le format complet (exemple : dimanche 19 juin 1944) pour avoir connaissance du jour. La méthode la plus simple consiste, bien entendu, à changer manuellement de format en mode création. Mais si l'opération doit être répétée souvent, il faut trouver un moyen pour l'automatiser. Une requête ajout, qu'il est facile d'exécuter depuis une macro, nous fournit la solution.

Par copier/coller (structure seulement) à partir de la table n° 1, nous obtenons une table « modèle » qui contient les mêmes champs (mais vides). Nous modifions le format du champ « Date », initialement « Date, abrégé », en « Date, complet » et nous enregistrons la modification. Grâce à une macro (cet objet est étudié dans les chapitres 26 et suivant), nous créons une copie de la table « modèle » que nous appelons table n° 3, puis nous lui ajoutons tous les champs de la table n° 1. Nous constatons que, dans la table n° 3, la date s'affiche en format complet, comme le montre la figure ci-dessous.

Nom	Prénom	Date
Machin	Pierre	12/6/1983
Truc	Nathalie	26/11/1985
Chose	André	5/2/1980

Table n° 1 (avant & après)

Nom	Prénom	Date

Table n° 3 (copie du modèle)

Nom	Prénom	Date
Machin	Pierre	dimanche 12 juin 1983
Truc	Nathalie	mardi 26 novembre 19685
Chose	André	mardi 5 février 1980

Table n° 3 (après ajout de la table n° 1)

Il existe en principe une méthode beaucoup plus simple pour modifier les propriétés d'un champ de manière automatisable. Elle consiste à écrire une requête SQL utilisant la commande ALTER TABLE avec la clause MODIFY (chapitre 18). Malheureusement, la clause MODIFY ne fonctionne pas dans Access... et on utilise la requête ajout pour pallier cette déficience.

Notre **quatrième exemple** montre comment on peut garder la trace du classement d'une table. Pour ce faire, nous traitons l'exemple d'une entreprise qui veut établir la liste de ses produits classés par ordre décroissant de chiffre d'affaires (CA) au cours de l'année écoulée. Le point de départ est une table contenant la liste des produits (classés par ordre alphabétique) avec leur CA. La méthode la plus simple consiste à trier la table par ordre de CA décroissant, à l'enregistrer, puis à la doter (en mode création) d'une colonne supplémentaire du type de données NuméroAuto. Mais si nous avons besoin d'automatiser l'opération, il nous faut recourir à une autre solution.

La figure ci-dessous montre la méthode utilisée. Nous trions la table de départ (Table1) par ordre de CA décroissant. Nous l'ajoutons à une table vide contenant les mêmes champs, plus un champ de type NuméroAuto (Table2). Puis, à l'aide d'une requête de sélection simple, nous trions la table Table2 par ordre alphabétique du premier champ. Le résultat final est une table des produits classés par ordre alphabétique, avec une colonne indiquant le rang de classement par ordre de CA décroissant. On notera que le champ "Classement" a été rempli par le SGBD (l'opérateur ne peut pas écrire dans ce champ).

Produit	CA
prod01	12 345,00 €
prod02	67 890,00 €
prod03	527,12 €
prod04	92 187,55 €

Table1

Produit	CA
prod04	92 187,55 €
prod02	67 890,00 €
prod01	12 345,00 €
prod03	527,12 €

Table1 (triée)

Produit	CA	Classement
		(NuméroAuto)

Table2 (avant ajout)

Produit	CA	Classement
prod04	92 187,55 €	1

Produit	CA	Classement
prod01	12 345,00 €	3

prod02	67 890,00 €	2
prod01	12 345,00 €	3
prod03	527,12 €	4

Table2 (après ajout de Table1)

prod02	67 890,00 €	2
prod03	527,12 €	4
prod04	92 187,55 €	1

Table2 (triée)

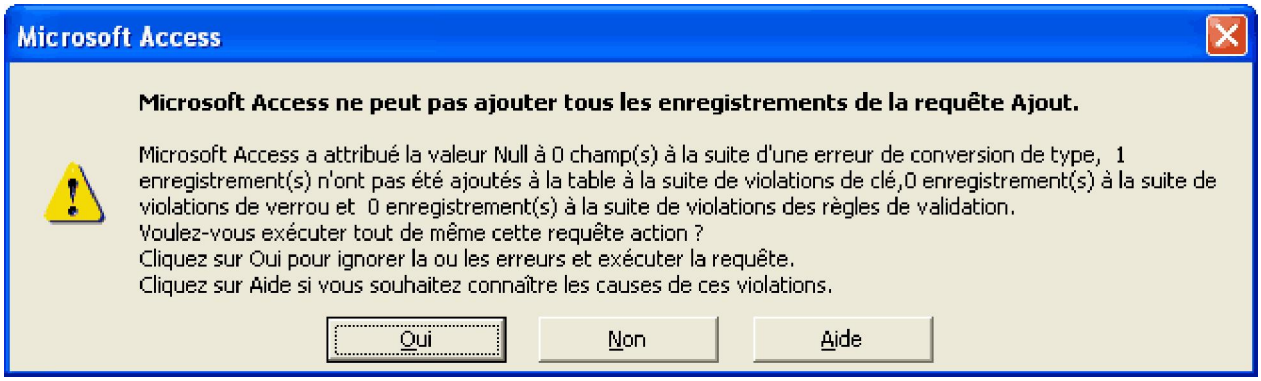
Une autre solution à ce problème consiste à utiliser la commande ALTER TABLE en SQL, avec la clause ADD COLUMN et le type de données COUNTER, qui correspond à NuméroAuto (voir le chapitre 18).

15.4 L'ajout sans doublons

La requête ajout crée des doublons si la seconde table contient des enregistrements identiques à ceux de la première. Lecas le plus flagrant résulte de l'ajout d'une table à elle-même, opération qui est tout à fait licite dans Access, même si son intérêt parait à peu près nul. On notera que la plupart des SGBD interdisent cette opération.

Le premier correctif auquel nous songions consiste à basculer de "Non" à "Oui" la propriété "Valeurs distinctes" de la requête ajout. Ainsi modifiée, la requête n'élimine pas les doublons qui résultent de l'ajout, mais évite de transporter dans la première table des enregistrements qui constituent des doublons dans la seconde. C'est mieux que rien, mais ce n'est pas suffisant.

Le second correctif auquel nous songions consiste à créer un index sans doublons sur les champs de la première table communs avec ceux de la seconde table. Lorsque nous lançons la requête ajout, nous recevons l'alerte suivante, qui constitue un morceau d'anthologie en matière de message informatique. Mais si nous admettons que par "violation de clé" il faut entendre "violation d'indexation sans doublons", tout s'éclaire :



Cliquons sur "Oui" et le tour est joué : le SGBD n'ajoute que les enregistrements qui ne créent pas de doublon. Si nous cliquons sur "Non", la requête est annulée. Si nous cliquons sur "Aide", nous obtenons une aide qui n'a rien à voir avec le contexte.

Deux autres solutions peuvent être pratiquées :

faire suivre la requête Ajout d'une requête qui élimine les doublons. Nous avons appris à créer une telle requête dans le chapitre précédent ;

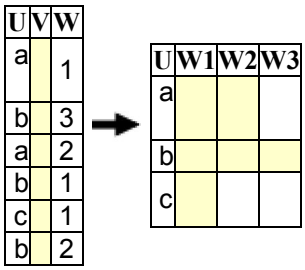
créer une requête Union en langage SQL. Nous apprendrons à nous servir du langage SQL dans un chapitre ultérieur.

Il faut cependant bien noter que l'ajout et l'union fonctionnent de manière distincte. Dans la requête union, les deux tables que l'on réunit jouent des rôles identiques. Comme nous venons de le constater, ce n'est pas le cas dans la requête ajout.

15.5 L'analyse croisée

La requête analyse croisée s'applique à une table comportant au moins trois colonnes, et possédant des caractéristiques particulières. L'une des colonnes doit comporter des doublons, sur lesquels sera effectuée l'opération de **regroupement** (la colonne "U" de la figure ci-dessous). Une autre colonne

(la colonne "W" de la figure ci-dessous) doit comporter un nombre restreint de valeurs distinctes, qui serviront à créer les nouvelles colonnes. Un assistant facilite la création de ce type de requête, dont la conception n'est pas aisée.



Considérons l'exemple de la table (nommée « Table1 ») représentée ci-dessous. Une entreprise a dressé la liste de ses fournisseurs et, pour chacun d'entre eux, la liste des produits fournis ainsi que le classement par ordre de chiffre d'affaires.

Société	Prod	Rang
Machin	prod1	1
Machin	prod4	2
Machin	prod12	4
Truc	prod2	1
Truc	prod5	3
Machin	prod21	3
Truc	prod6	2
Chose	prod2	2
Chose	prod30	1

Dans la fenêtre "Base de données", l'objet "Requêtes" étant sélectionné, nous cliquons sur le bouton nouveau, nous choisissons "Assistant Requête analyse croisée" dans la liste qui s'affiche, et nous cliquons sur "OK". Le dialogue suivant s'établit avec l'assistant :

nous indiquons d'abord sur quelle table nous voulons opérer. Dans le cas présent, il s'agit de la table "Table1" ;

nous choisissons le champ "Société" comme "en-tête de ligne". Dans le jargon de l'éditeur, cela signifie que ce champ sera le premier de la nouvelle table ;

nous choisissons le champ "Rang" comme "en-tête de colonne". Cela signifie que le SGBD va créer les colonnes "1", "2", etc. ;

nous choisissons "premier" et nous décochons la case "Oui, inclure les sommes des lignes" car les données sont du type texte et non du type numérique ;

nous cliquons sur le bouton "Terminer" et nous basculons en mode feuille de données pour examiner le résultat (figure ci-dessous).

Société	1	2	3	4
Chose	prod30	prod2		
Machin	prod1	prod4	prod21	prod12

Truc	prod2	prod6	prod5	

La requête analyse croisée est surtout utilisée dans le domaine financier, où elle sert à créer des bilans à partir de données comptables. Les nouvelles colonnes qui sont crées correspondent alors à des périodes de temps données (jours, semaines, mois, etc.).

15.6 Conclusion

Les requêtes de suppression et de mise à jour peuvent être regroupées sous la bannière unique de la maintenance des BDD.

16 Les requêtes de maintenance

16.1 Introduction

Une base de données évolue sans arrêt : de nouveaux enregistrements sont introduits, d'autres sont archivés, d'autres sont modifiés, d'autres enfin sont supprimés. Des contrôles, suivis éventuellement de corrections, sont effectués.

Deux types de requête sont particulièrement utilisés pour ces opérations de maintenance :

la **suppression**, qui permet de faire disparaître des enregistrements jugés obsolètes, erronés ou inutiles ;

la **mise à jour**, qui permet de modifier le contenu de certains enregistrements.

Sont supprimés ou modifiés les enregistrements qui répondent à certains **critères**. Ces opérations sont généralement effectuées sur une seule table à la fois. Cependant, par le jeu des relations, suppressions et modifications peuvent se répercuter en cascade dans d'autres tables, si l'option correspondante a été choisie lors de la création de la relation, après que l'intégrité référentielle eût été requise (Cf. le chapitre 5).

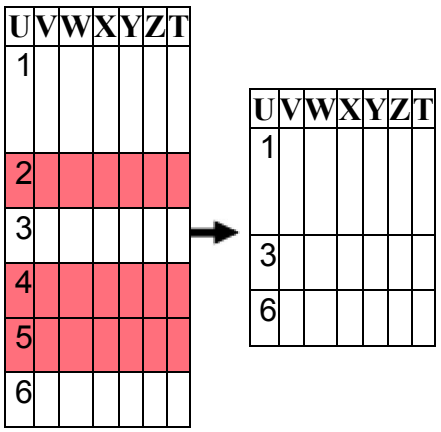
La suppression est une opération qui s'effectue au niveau de l'enregistrement. La mise à jour intervient souvent à un niveau plus fin : seuls certains champs, dans certains enregistrements, sont concernés.

Attention ! Une fois effectuées, suppressions et mises à jour sont irréversibles. Avant d'effectuer une requête de ce type, il est indispensable d'effectuer une copie des tables concernées, voire de la BDD toute entière.

Bien entendu, une opération de modification ou de suppression concernant un tout petit nombre d'enregistrements ne justifie pas la création d'une requête ; il suffit d'intervenir ponctuellement dans la table considérée. Par contre, si le nombre d'opérations à effectuer croit, la création d'une requête fait gagner du temps et diminuer le risque d'erreurs.

16.2 La suppression

La requête de suppression opère sur une table. Elle supprime les enregistrements (ou lignes) répondant à un ou plusieurs critères. Dans la figure ci-dessous, les enregistrements répondant à ces critères sont colorés en rouge ; la requête les fait disparaitre irrémédiablement.



A titre d'exemple, l'opération de suppression peut être utile dans les cas suivants :

retirer de la BDD d'une entreprise toutes les données économiques relatives à un exercice clos ;

retirer de la table des prêts (d'une bibliothèque) toutes les opérations terminées (le livre emprunté a été rendu) ;

retirer du fichier journal d'un système informatique tous les enregistrements âgés de plus d'un mois ;

retirer de la liste des clients tous ceux qui n'ont rien commandé depuis deux ans ;

éliminer d'une table tous les enregistrements incomplets (un champ donné n'a pas été renseigné) ;

retirer de la table du stock tous les articles qui se sont mal vendus au cours des douze derniers mois ;

etc.

Supposons, à titre d'exemple, que nous voulions éliminer d'une table "Factures" toutes les factures soldées avant le 1er janvier de cette année (2002). Dans la fenêtre "Base de données" nous sélectionnons l'objet "Requêtes", puis nous double-cliquons sur "Créer une requête en mode Création". Dans la boite "Afficher la table", nous sélectionnons la table "Factures", nous l'ajoutons et nous fermons. Dans la fenêtre "Microsoft Access", nous cliquons sur l'icône de la barre d'outils et, dans la liste déroulante qui s'affiche, nous choisissons "Requête Suppression". Nous introduisons le champ "Date_de_réglement" dans la grille et, sur la ligne "Critères :", nous écrivons :

<#01/01/2002#

l'opérateur < signifiant "antérieur à", et les signes # rappelant que nous opérons sur une date.

Attention ! Si nous basculons en mode feuille de données en cliquant sur l'icône , le logiciel affiche la liste des enregistrements qui seront supprimés, et non l'aspect de la table après suppression comme nous pourrions nous y attendre.

Si nous exécutons la requête (icône), tous les enregistrements antérieurs au 1er janvier 2002 sont éliminés. Si la table "Factures" est ouverte pendant la suppression, le mot "Supprimé" apparaît dans chaque champ de chaque enregistrement supprimé. Dès que l'on referme la table, les enregistrements supprimés disparaissent sans laisser de traces.

L'écriture d'un critère de suppression doit tenir compte du type de données du champ auquel il s'applique. Nous associerons sous peu à ce chapitre des annexes détaillant les règles d'écriture des critères pour les différents cas.

16.3 La mise à jour

Les informations contenues dans une BDD peuvent avoir besoin d'une mise à jour. Ainsi, un taux de TVA peut varier, de même que le prix de vente des produits et services de

l'entreprise, etc. Mais la mise à jour peut être soumise à un ou plusieurs critères, s'appliquant ou non au champ susceptible d'être modifié. Par exemple, une prime peut être versée à l'ancienneté, et donc dépendre de la date d'embauche du salarié (laquelle figure dans la table du personnel de l'entreprise). Par conséquent, la mise à jour s'applique à une table, et concerne soit une colonne complète, soit seulement certains enregistrements de la colonne (colorés en rose dans la figure ci-dessous).

U	V	W	X	Y	Z	T
1						
2						
3						
4						
5						
6						

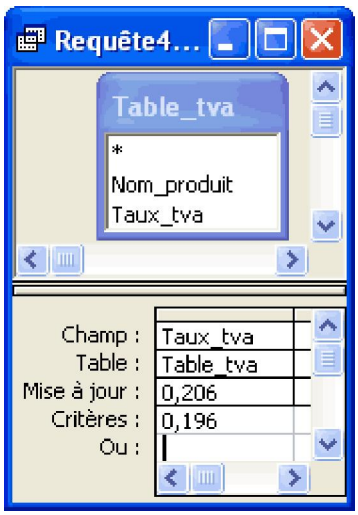
L'actualité proche nous fournit un excellent exemple mise à jour simple : lors du passage à l'euro, tous les prix contenus dans les base de données durent être convertis de francs en euros à l'aide du fameux coefficient 6,55957. Pour traiter ce cas, créons une table "Produits" contenant trois champs : le code, le nom, et le prix unitaire HT de chaque produit. Les types de données correspondants sont : NuméroAuto, texte, et numérique (réel, deux décimales, pas de format particulier). Saisissons quelques enregistrements, et créons une copie de la table ainsi remplie.

Créons une requête en mode création. Introduisons la table "Produits", cliquons sur l'icône  , et sélectionnons "Requête Mise à jour" dans la liste déroulante. Introduisons le champ "Prix_unitaire" dans la grille et, sur la ligne "Mise à jour :", saisissons l'expression représentée dans la figure ci-dessous. Laissons la ligne "Critères :" vide, puisque le passage des francs aux euros concerne tous les enregistrements de la colonne sans exception.

Attention ! si nous passons en mode feuille de données à l'aide de l'icône  , nous constatons que l'opération de mise à jour n'est pas effectuée. Un bogue de plus à signaler à l'éditeur !

Si nous exécutons la requête, nous constatons que tous nos prix ont bien été convertis en euros. Mais prenons garde de ne pas exécuter la requête une deuxième fois !

Cet exemple de requête "mise à jour" est particulièrement simple, parce qu'il s'applique à tous les enregistrements sans distinction. On peut créer facilement un exemple avec critère, comme le montre la figure ci-dessous. La requête portera à 20,6 % le taux de TVA de 19,6 %, mais ne touchera pas aux autres taux.



16.4 Conclusion

Les bases de données nécessitent des opérations de maintenance. Pour réaliser ces dernières, on utilise des requêtes, et plus particulièrement deux d'entre elles, la suppression et la mise à jour.

Ces requêtes ne créent pas de table, elles modifient les tables existantes. Ces modifications étant irréversibles, il est indispensable d'effectuer des copies de sauvegarde avant d'exécuter les requêtes.

Quelles que soient les précautions prises, il est parfois difficile d'éviter que des informations soient saisies deux fois. La recherche et l'élimination des doublons est donc également utilisée pour la maintenance des BDD.

17 Les tables en SQL

17.1 Introduction à SQL

Le sigle **SQL** signifie "Structured Query Language", soit en français "Langage de recherche structuré". SQL est un langage de gestion des bases de données relationnelles que presque tous les SGBD comprennent. Il a été développé par IBM dans le courant des années 70, et son nom actuel (il s'appelait initialement SEQUEL) date du début des années 80.

SQL a été normalisé par l'ANSI (American National Standards Institute) et par l'ISO (International Organization for Standardization). Voici les principales étapes de ce processus :

- première norme ANSI en 1986 ;
- première norme ISO (SQL1) en 1987, révisée en 1989 ;
- deuxième norme ISO (SQL2) en 1992 ;
- troisième norme (SQL3) en cours de rédaction depuis 1999 par l'ANSI et l'ISO, après une très longue gestation, et avec beaucoup de retard sur l'événement.

Malgré la normalisation ISO, l'implémentation du SQL par les différents éditeurs de SGBD comporte des différences plus ou moins marquées concernant :

- les détails de la syntaxe ;
- l'écriture des commandes ;
- le fonctionnement exact des commandes ;
- l'implémentation de nouveaux types de données (images, animations, vidéos, liens hypertexte, etc.).

Bref, il n'y a qu'un seul langage SQL, mais chaque éditeur de SGBD implémente son propre dialecte. Le "dictionnaire" qui permet de passer d'un dialecte à l'autre s'appelle ODBC (Open Data Base Connectivity). Il a été imaginé par Microsoft, et mis sur le marché en 1993.

Contrairement à ce que son nom indique, SQL ne sert pas qu'à écrire des requêtes. C'est un langage complet, qui permet de créer des BDD, des tables, de saisir des données et de les corriger, de créer des vues, des index et des états (parfois baptisés "rapports", par francisation de l'anglais "reports"). Sauf erreur de notre part, il ne permet pas de créer des formulaires, parce qu'il a été conçu à une époque ou l'interface graphique n'existait pas sur ordinateur, et parce qu'un formulaire sans interface graphique n'a guère d'intérêt.

Par contre, dans les SGBD sans interface graphique, le recours à SQL est obligatoire pour toutes les opérations, y compris la création de la BDD, celle des tables, et la saisie des données. La tentation est donc forte, pour un professeur qui ignore à quels SGBD ses étudiants seront confrontés lorsqu'ils entreront dans la vie active, de faire dans son enseignement une large part à SQL, qui représente l'outil universel de manipulation des données. Ceci dit, l'interface graphique est tellement entrée dans les moeurs, qu'il parait difficile qu'à terme tous les SGBD n'en soient pas dotés. En attendant, nous avons adopté une position mixte, en commençant par l'interface graphique, plus facile à appréhender, et en rajoutant à ce cours une partie consacrée au langage SQL.

17.2 Le langage SQL dans Access

Dans Access, le langage SQL est utilisé par le moteur du SGBD pour traduire en commandes exécutables les instructions que donne l'utilisateur à travers l'interface graphique. Mais l'utilisateur n'a pas accès à ce code, sauf pour la conception des requêtes, où il peut passer facilement du mode graphique au mode SQL et vice versa. Nous utiliserons largement cette possibilité dans les trois chapitres suivants.

Pour les tables, la situation est nettement moins satisfaisante. L'utilisateur qui se sert de l'interface graphique pour créer une table n'a pas accès au code SQL correspondant. Par contre, il dispose d'un éditeur de SQL qui reconnaît les principales commandes concernant la création et la modification des tables, et la saisie des données.

Cet éditeur, cependant, ne doit pas faire illusion, car il est loin d'être complet. Il ne permet pas de régler dans le détail les propriétés des champs, comme on peut le faire dans l'interface graphique. Il ne permet pas non plus de créer des listes. Il rend donc des services limités. Il présente cependant de l'intérêt dans les deux cas suivants :

- l'apprentissage initial du SQL, pour lequel il n'est pas utile d'entrer immédiatement dans les moindres détails ;

- l'automatisation (via les macros) de certaines opérations relatives aux tables. En effet, le code SQL que nous allons écrire sera enregistré sous forme de requête par le SGBD, et il est très facile de lancer une requête à partir d'une macro.

Évidemment, il est beaucoup plus facile de créer, remplir, modifier, et supprimer une table dans l'interface graphique d'Access qu'en utilisant des commandes SQL. Mais tous les SGBD ne sont pas dotés d'une interface graphique, et il est bon de savoir se débrouiller sans elle le cas échéant.

17.3 La procédure

Pour gérer les tables en langage SQL dans Access, il nous faut opérer de la manière suivante. Dans la fenêtre "Base de données", nous sélectionnons l'objet "Requêtes". Nous effectuons un double clic sur "Créer une requête en mode création", nous refermons la fenêtre "Afficher la table" sans introduire de table et, dans le menu, nous suivons le chemin suivant :

Requête --> Spécifique SQL --> Définition des données

S'ouvre alors une fenêtre intitulée "Requête1 : Requête Définition des données", dans laquelle nous pouvons écrire du code SQL.

Pour exécuter ce code, nous cliquons sur l'icône "Exécuter". Pour l'enregistrer, nous cliquons sur "Enregistrer". Nous constatons alors que le SGBD Access traite notre code comme une requête.

Pour modifier le code SQL, nous sélectionnons la requête enregistrée précédemment, et nous cliquons sur l'icône "Modifier". La fenêtre "Requête Définition des données" s'ouvre à nouveau.

Tous les exemples cités dans ce chapitre ont été transportés (par copier/coller) dans Access 2002, et nous avons vérifié leur bon fonctionnement. Vous ne devriez donc pas rencontrer de difficulté pour les reproduire.

17.4 La création et la suppression d'une table

Dans la fenêtre ouverte grâce à la procédure précédente, nous écrivons notre première **commande** (ou instruction) SQL, contenant la **clause** CREATE TABLE, pour créer la table "Personnes" (nous notons qu'un point-virgule marque la fin de la commande) :

```
CREATE TABLE Personnes
(Nom CHAR(20),
Prénom CHAR(20));
```

Nous exécutons cette commande en cliquant sur l'icône "Exécuter". Le fait qu'aucun message ne soit émis signifie que tout s'est bien passé. Nous sélectionnons l'objet "Table", et nous constatons que :

la table "Personnes" est effectivement créée ;

qu'elle possède deux champs de type texte (de 20 caractères au maximum) ;

qu'ils sont intitulés "Nom" et "Prénom".

Si nous enregistrons cette commande en cliquant sur l'icône "Enregistrer", le SGBD Access la traite comme une requête. Devant son nom, il place une icône particulière (, à ne pas confondre avec l'icône "Modifier") pour rappeler qu'il s'agit d'une commande SQL liée à la manipulation des tables.

Bien entendu, si nous n'exécutons pas la requête, la table "Personnes" ne sera pas créée. Par contre, si la table "Personnes" existe déjà, la commande ne s'exécute pas (la table existante n'est pas écrasée), et le SGBD affiche le message suivant :

Attention ! Si l'objet "Table" est sélectionné quand vous lancez l'exécution de la commande SQL, la table "Personnes" n'apparaîtra pas (c'est l'éternel problème de la synchronisation dans Access). Il suffit de cliquer, dans le menu, sur "Affichage", puis sur "Actualiser", pour que le nom de la table apparaisse.

Les commandes SQL s'expriment en **format libre**. Nous pouvons écrire les clauses en minuscules, et nous ne sommes pas tenus d'aller à la ligne pour détailler les champs. Bien que la précédente présentation (sur trois lignes) soit considérée comme plus lisible, l'expression suivante est parfaitement exacte et s'exécute normalement :

create table Personnes (Nom char(20),Prénom char(20));

Les conventions relatives aux noms des tables et des champs varient quelque peu d'un SGBD à l'autre. En ce qui concerne plus particulièrement les champs :

le nombre de caractères ne doit pas être trop grand (64 dans Access, 18 à 30 dans d'autres SGBD) ;

seuls les lettres, les nombres et le caractère de soulignement sont autorisés. Access admet les caractères accentués. Il admet aussi l'espace, mais le nom du champ doit alors être écrit entre crochets ;

certains SGBD requièrent que le nom d'un champ commence par une lettre, mais ce n'est pas le cas d'Access ;

les termes faisant partie du vocabulaire du langage SQL sont interdits ("date" par exemple). Ce sont les mots réservés.

Les types de données sont définis dans le DDL (Data Definition Language) de chaque SGBD, et ils varient beaucoup d'un logiciel à l'autre. Dans Access, les mêmes termes ne sont pas toujours utilisés dans l'interface graphique, en VBA et en SQL. Voici un échantillon représentatif des différentes façons d'exprimer un type de données lors de la création d'une table en SQL dans Access :

Booléen : BIT ;

Nombre entier : SHORT (entier), SMALLINT (entier), LONG (entier long), INTEGER (entier long), BYTE (octet) ;

Nombre réel : SINGLE (réel simple), DOUBLE (réel double), NUMERIC (réel double) ;

Monétaire : CURRENCY, MONEY ;

Date/Heure : DATE, TIME, DATETIME ;

Texte : VARCHAR (255 caractères), CHAR(n) ou TEXT(n) (n caractères), LONGTEXT (mémo, 32K max.) ;

Fichier binaire : LONGBINARY (Objet OLE) ;

Compteur : COUNTER (NuméroAuto).

On notera qu'il n'est pas possible de créer un champ de type hypertexte via une commande SQL dans Access. Même remarque en ce qui concerne les listes de choix.

Pour supprimer une table, on utilise la clause DROP TABLE, comme le montre l'exemple suivant :

DROP TABLE Personnes;

Attention ! Si l'objet "Table" est sélectionné dans la fenêtre "Base de données" quand vous lancez l'exécution de la commande SQL de suppression, la table "Personnes" ne disparaîtra pas (c'est l'éternel problème de la synchronisation dans Access). Il suffit de cliquer, dans le menu, sur "Affichage", puis sur "Actualiser", pour que le nom de la table disparaisse.

Si nous enregistrons la commande de suppression de table, Access place devant son nom l'icône  spécifique des requêtes SQL liées à la création de tables.

17.5 La modification d'une table

Il est possible de modifier une table existante. Les exemples les plus classiques concernent l'addition d'une nouvelle colonne et la suppression d'une colonne existante. La commande :

```
ALTER TABLE Personnes
ADD Naissance DATE;
```

permet, lorsqu'on l'exécute, d'ajouter le champ intitulé "Naissance", de type Date/Heure, à la table "Personnes". La variante suivante fonctionne également :

```
ALTER TABLE Personnes
ADD COLUMN Naissance DATE;
```

La clause INIT, qui permet de donner une valeur initiale aux champs ainsi créés, ne fonctionne pas dans Access. Par défaut, cette valeur initiale est Null. Pour la modifier, il faut utiliser une commande UPDATE (dont nous parlerons au paragraphe 9 ci-dessous).

Pour supprimer la colonne que nous venons de créer, nous utilisons la commande suivante :

```
ALTER TABLE Personnes
DROP Naissance;
```

ou sa variante :

```
ALTER TABLE Personnes
DROP COLUMN Naissance;
```

En SQL standard, la commande ALTER TABLE peut aussi être utilisée pour modifier les propriétés d'une colonne existante. Exemple :

```
ALTER TABLE Personnes
MODIFY Nom CHAR(40);
```

mais la clause MODIFY n'est pas reconnue par Access, et l'exécution de la commande ci-dessus entraîne un message d'erreur. L'ignorance de la clause MODIFY enlève à la commande ALTER TABLE une bonne partie de son intérêt dans Access, et l'on se demande pourquoi l'éditeur a fait les choses à moitié.

Nous verrons cependant au paragraphe 8 que la commande ALTER TABLE admet la clause ADD CONSTRAINT, ce qui permet de rajouter une clé ou de créer une relation.

17.6 Les propriétés des champs

Le langage SQL est doté de clauses permettant de définir les propriétés des champs lors de la création d'une table. Mais le moteur d'Access ne les reconnaît pas toutes, loin de là.

Pour empêcher un champ de rester vide, nous utilisons la clause NOT NULL, comme le montre l'exemple suivant :

```
CREATE TABLE Personnes
(Nom CHAR(20) NOT NULL,
Prénom CHAR(20)) ;
```

Après avoir exécuté la commande nous vérifions, dans le propriétés de la table "Personnes" ainsi créée, que le Null est interdit dans le champ "Nom".

Pour qu'un champ soit indexé sans doublons, nous utilisons la clause UNIQUE, comme le montre l'exemple suivant :

```
CREATE TABLE Personnes
(Nom CHAR(20) UNIQUE,
Prénom CHAR(20)) ;
```

Après avoir exécuté la commande, nous ouvrons la table "Personnes" en mode modification, nous cliquons sur le champ "Nom", et nous vérifions que la propriété "Indexé :" vaut "Oui - Sans doublons". En fait, cette clause possède un intérêt limité pour deux raisons :

nous ne pouvons pas donner de nom à l'index. Ceci nous interdit de supprimer l'index via une commande SQL ;

nous ne pouvons pas créer d'index multi-champ de cette façon.

Il est donc souvent préférable d'utiliser la commande de création d'index que nous présenterons au paragraphe suivant.

Pour poser une clé primaire sur un champ, nous utilisons la clause PRIMARY KEY, comme le montre l'exemple suivant :

```
CREATE TABLE Personnes
(Nom CHAR(20) PRIMARY KEY,
Prénom CHAR(20));
```

Après avoir exécuté la commande, nous ouvrons la table « Personnes » en mode modification, nous cliquons sur le champ « Nom », et nous vérifions qu'il est effectivement doté de la clé. Cette commande possède les deux mêmes défauts que la précédente : nous ne sommes pas maîtres du nom de la clé (le système l'appellera Index_976A9AC0_494C_41C1, par exemple…), et nous ne pouvons pas appliquer la clé à plusieurs champs simultanément. Le premier défaut peut être corrigé grâce à la commande suivante :

```
CREATE TABLE Personnes
(Nom CHAR(20) CONSTRAINT clé_primaire PRIMARY KEY,
Prénom CHAR(20));
```

qui permet d'attribuer le nom "clé_primaire" à la clé ainsi créée. Nous verrons au paragraphe suivant comment placer une clé sur plusieurs champs.

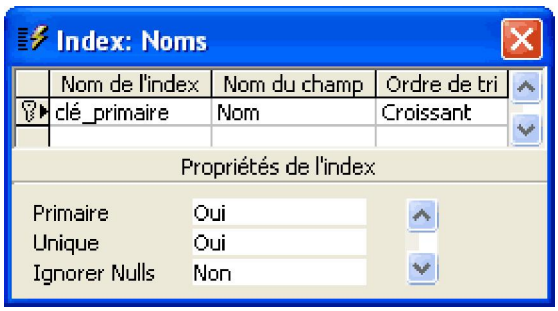
Les autres clauses permettant de définir les propriétés des champs ne fonctionnent pas dans Access. Il en est ainsi de DEFAULT, qui permet de fixer la valeur par défaut d'un champ, ainsi que de CHECK, qui permet de fixer des contraintes sur le contenu d'un champ (propriété "Valide si").

17.7 La clé primaire et l'index

Pour placer une clé primaire sur un champ, nous pouvons utiliser la clause CONSTRAINT, qui est obligatoirement suivie d'un nom d'index, et que nous avons déjà rencontrée au paragraphe précédent. Créons, par exemple, la table "Personnes" avec une clé primaire (intitulée clé_primaire) sur le champ "Nom". La commande s'écrit :

```
CREATE TABLE Personnes
(Nom CHAR(20) NOT NULL,
Prénom CHAR(20),
CONSTRAINT clé_primaire PRIMARY KEY(Nom)) ;
```

L'objet "Table" étant sélectionné, nous cliquons sur l'icône "Index", et la fenêtre du même nom s'ouvre. Nous vérifions que la clé est bien nommée "clé_primaire", comme le montre la figure suivante :



Pour appliquer la clé à deux champs, nous utilisons la syntaxe suivante :

```
CREATE TABLE Personnes
(Nom CHAR(20),
Prénom CHAR(20),
CONSTRAINT essai_index PRIMARY KEY(Nom, Prénom)) ;
```

La création d'un index peut s'effectuer alors que la table existe déjà, mais cela requiert l'usage d'une syntaxe différente. Créons par exemple un index sur le champ "Nom" de la table "Personnes" :

```
CREATE UNIQUE INDEX essai_index
ON Personnes (Nom) ;
```

Dans le cas d'un index sur deux champs, cette syntaxe devient :

```
CREATE UNIQUE INDEX essai_index
ON Personnes (Nom, Prénom);
```

Pour supprimer un index, la syntaxe SQL standard s'écrit :

```
DROP INDEX Personnes.essai_index;
```

Mais cette syntaxe standard ne fonctionne pas dans Access. Il faut utiliser la variante suivante :

```
DROP INDEX essai_index ON Personnes;
```

17.8 La création et la suppression d'une relation

Pour montrer comment on crée une relation 1-n entre deux tables, nous avons décomposé les opérations en quatre étapes. Dans un premier temps, nous créons la table "Personnes", avec un champ "Nom", un champ "Prénom", et un champ "Code_Ville" (entier long), en exécutant la commande suivante :

```
CREATE TABLE Personnes
(Nom              TEXT(30),
Prénom            TEXT(30),
Code_Ville LONG) ;
```

Dans un deuxième temps, nous créons la table "Villes", avec un champ "Code_Ville" (NuméroAuto) et un champ "Ville". Cette table servira de liste externe pour la table "Personnes". Nous exécutons la commande suivante :

```
CREATE TABLE Villes
(Code_Ville COUNTER,
Ville TEXT(30));
```

Dans un troisième temps, nous modifions la table "Villes" en plaçant une clé primaire sur le champ "Code_Ville", qui servira de côté 1 à la future relation. Pour ce faire, nous exécutons la commande suivante :

```
ALTER TABLE Villes
ADD CONSTRAINT clé_primaire PRIMARY KEY(Code_Ville);
```

Notons que, si nous ne désirons pas donner un nom à la clé (ici "clé_primaire"), nous pouvons simplifier la commande précédente en l'écrivant ainsi :

```
ALTER TABLE Villes
ADD PRIMARY KEY(Code_Ville);
```

Dans un quatrième temps, nous modifions la table "Personnes" en plaçant une clé étrangère, nommée "relation_ville", sur le champ "Code_Ville" de la table "Personnes" (on parle de **clé**

étrangère pour le côté n de la relation), en précisant que le côté 1 de la relation est le champ "Code_Ville" de la table "Villes". Nous exécutons donc la commande suivante :

```
ALTER TABLE Personnes
ADD CONSTRAINT relation_ville FOREIGN KEY (Code_Ville) REFERENCES Villes (Code_Ville);
```

Notons que :

si nous ne désirons pas donner un nom à la relation (ici "relation_ville"), et

s'il n'y a pas d'ambiguïté sur le champ qui est du côté 1 de la relation (ici "Code_Ville de la table "Villes")

nous pouvons simplifier la commande précédente en l'écrivant ainsi :

```
ALTER TABLE Personnes
ADD FOREIGN KEY (Code_Ville) REFERENCES Villes;
```

Nous vérifions dans la fenêtre "Relations" qu'une relation 1-n a bien été créée entre les deux tables, avec application de l'intégrité référentielle.

Nous aurions pu être plus directs, en installant les clés (clé primaire et clé étrangère) dès la création des tables. Pour la table ville, la commande s'écrit :

```
CREATE TABLE Villes
(Code_Ville COUNTER PRIMARY KEY,
Ville TEXT(30));
```

Pour la table "Personnes", la commande s'écrit :

```
CREATE TABLE Personnes
(Nom TEXT(30),
Prénom TEXT(30),
Code_Ville LONG,
CONSTRAINT relation_ville FOREIGN KEY (Code_Ville) REFERENCES Villes);
```

Ces deux commandes étant exécutées, nous vérifions dans la fenêtre "Relations" que les deux tables sont présentes avec tous leurs champs, et liées par une relation 1-n.

La suppression de cette relation s'obtient à l'aide de la commande ALTER TABLE. Dans le cas de la table "Personnes", côté n de la relation "relation_ville" créée précédemment, la commande s'écrit :

```
ALTER TABLE Personnes
DROP CONSTRAINT relation_ville;
```

On notera que les options "Mettre à jour en cascade les champs correspondants" et "Supprimer en cascade les enregistrements correspondants" ne sont pas disponibles.

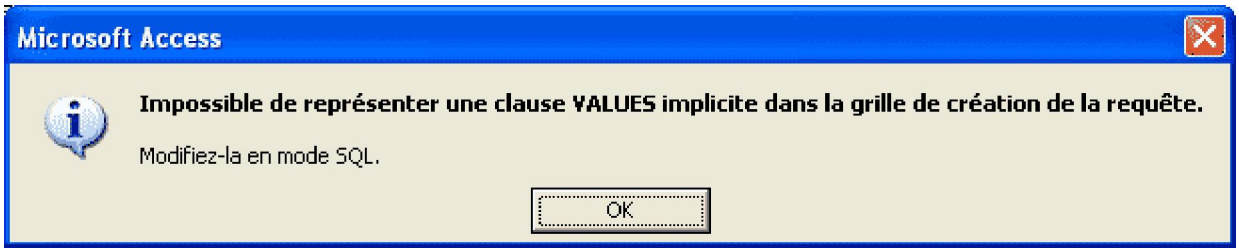
En conclusion, la création et la suppression d'une relation sont réalisées grâce à la clause CONSTRAINT appliquée à la table située du côté n de la relation.

17.9 La saisie et la correction des données

Pour saisir des données dans la table "Personnes", la commande SQL utilise la clause INSERT INTO. Les données en mode texte doivent être placées entre guillemets. Exemple :

```
INSERT INTO Personnes
VALUES
("Machin","Pierre");
```

Nous vérifions, après exécution de la commande, que Pierre Machin a bien été introduit dans la table "Personnes". Si nous enregistrons la commande, le SGBD la fait précéder de l'icône , qui symbolise les requêtes de type "Ajout". Cependant, si nous essayons de basculer en "Mode création" (le mode graphique), le SGBD Access nous oppose le message suivant :



En bon français, notre commande ressemble à une requête ajout, elle possède l'icône d'une requête ajout, elle utilise la clause INSERT comme une requête ajout, mais ce n'est pas une requête ajout. Qu'on se le dise !

Si le Null n'est pas interdit dans les champs de la table "Personnes", nous pouvons introduire un nom sans le prénom correspondant, en opérant de la manière suivante :

```
INSERT INTO Personnes (Nom)
VALUES
("Truc");
```

Pour modifier un enregistrement existant, nous faisons appel à la clause UPDATE (qui signifie "mise à jour" en anglais). Si, par exemple, nous voulons doter M. Truc de son prénom, nous écrirons :

```
UPDATE  Personnes
SET  Prénom="Henri"
WHERE Nom="Truc";
```

Si nous enregistrons cette commande, le SGBD Access lui attribue l'icône caractéristique des requêtes de mise à jour. C'en est effectivement une, comme nous pouvons le constater en basculant en mode graphique. Nous reparlerons de ce type de requête au chapitre 22.

Pour supprimer une ligne, nous utilisons la commande basée sur la clause DELETE :

```
DELETE FROM Personnes
WHERE Nom="Truc";
```

Lorsque la clause WHERE est absente, le SGBD supprime tous les enregistrements, laissant la table vide (mais ne la supprimant pas) :

```
DELETE FROM Personnes;
```

Si nous enregistrons ces deux commandes, le SGBD Access fait précéder leur nom de l'icône caractéristique des requêtes suppression. Cependant, si nous créons la requête suppression correspondante en mode graphique, et si nous basculons en mode SQL, nous obtenons une syntaxe légèrement différente. Nous reviendrons sur ce point au chapitre 22.

Dans certaines implémentations du langage SQL (mais pas dans Access), on peut omettre la clause FROM qui suit la clause DELETE.

17.10 Conclusion

La création d'une table en SQL n'est pas une travail bien ardu, même s'il est certain qu'une bonne interface graphique simplifie fortement l'opération. La saisie des informations en SQL, par contre, est une tâche quasi désespérante. A moins que les données ne soient importées, l'usage d'une interface graphique s'impose.

En ce qui concerne les tables, l'interface graphique du SGBD Access est beaucoup plus développée que son interface SQL. Nous verrons dans les chapitres suivants que la situation est très différente, et nettement plus équilibrée, pour les requêtes.

18 La sélection simple en SQL

18.1 Introduction

Nous avons vu au chapitre précédent qu'il était possible, dans le SGBD Access, de manipuler les tables en langage SQL. Cependant, les commandes correspondantes sont considérées comme des requêtes, et il n'est pas possible de basculer entre le mode graphique et le mode SQL. En effet, le mode graphique s'obtient lorsque l'objet "Tables" est sélectionné, alors que le mode SQL requiert que l'objet "Requêtes" soit actif.

En ce qui concerne les requêtes, la situation est nettement plus satisfaisante. La plupart des commandes SQL relatives aux requêtes sont connues du moteur d'Access, et on bascule sans problème du mode graphique au mode SQL (l'objet "Requêtes" étant sélectionné).

Il existe cependant quelques exceptions, que nous étudierons au chapitre 21. Il s'agit des opérations ensemblistes, pour lesquelles il n'existe pas d'interface graphique. Ces trois opérations sont :

l'union de deux tables, pour laquelle l'opérateur UNION fonctionne ;

l'intersection de deux tables, pour laquelle l'opérateur INTERSECT ne fonctionne pas ;

la différence de deux tables, pour laquelle les opérateurs EXCEPT et MINUS ne fonctionnent pas.

18.2 La sélection simple

Créons, dans l'interface graphique, la requête qui extrait de la table "Personnes" (contenant une liste de personnes) les deux champs "Nom" et "Prénom". Cliquons sur la petite flèche située à droite de l'outil  "Affichage", et dans la liste déroulante, choisissons "Mode SQL". La **commande** (ou instruction) suivante s'affiche :

```
SELECT Personnes.Nom, Personnes.Prénom
FROM Personnes;
```

La requête simple commence par la **clause** "SELECT", suivie du nom des champs, puis continue avec la clause "FROM", suivie du nom de la table à laquelle appartiennent les champs. Le point-virgule marque la fin de la commande.

La syntaxe relative aux noms des champs consiste à écrire le nom de la table, suivi d'un point et du nom du champ. Cette façon de procéder s'appelle la **qualification**.

Dans le cas présent, cette qualification est redondante, et nous pouvons très bien écrire :

```
SELECT Nom, Prénom FROM Personnes;
```

La politique la plus raisonnable consiste à qualifier les champs chaque fois qu'une ambiguïté existe (même nom de champ dans deux tables différentes, lors d'une requête multi-table), et de ne pas les qualifier dans le cas contraire.

Nous avons vu au chapitre précédent qu'il existait des restrictions sévères sur les noms des tables et des champs en SQL. Pour s'en affranchir, il faut mettre les noms des champs, et celui de la table, entre crochets pour éviter les ennuis. Les expressions :

```
SELECT [Personnes].[Nom], [Personnes].[Prénom] FROM [Personnes];
```

```
SELECT [Nom], [Prénom] FROM [Personnes];
```

sont parfaitement valables. Par prudence, certains professionnels utilisant les SGBD préfèrent s'abstenir de tout caractère accentué, remplacent systématiquement l'espace par le caractère de soulignement, et évitent d'utiliser les termes réservés. Rappelons que l'implémentation de SQL par Access accepte les caractères accentués pour les noms des champs et des tables.

Attention aux détails de syntaxe ! Comme tous les langages informatiques, SQL a ses petites manies qui empoisonnent les utilisateurs. L'interface graphique a ceci de bon qu'elle nous débarrasse de ces problèmes stupides -- en plus du fait qu'elle nous permet de créer des requêtes plus simplement et plus rapidement. On notera que, dans Access, le point-virgule qui marque la fin d'une commande n'est pas indispensable.

18.3 La requête avec création de table

Récupérons la requête précédente dans l'interface graphique, faisons en sorte qu'elle crée une table appelée "Essai", puis basculons en mode SQL. Nous obtenons :

```
SELECT Personnes.Nom, Personnes.Prénom INTO Essai
FROM Personnes;
```

Nous voyons que la création de la table est effectuée grâce à la clause INTO, suivi du nom de la table. En SQL version Oracle, on écrirait plutôt :

```
INSERT INTO Essai
SELECT Personnes.Nom, Personnes.Prénom
FROM Personnes;
```

Dans Access, cette syntaxe fonctionne à condition que la table "Essai" préexiste, et contienne au moins les champs "Nom" et "Prénom" avec les mêmes propriétés que dans la table "Personnes". Access effectue alors une requête ajout des deux premières colonnes de la table "Personnes" à la table "Essai".

18.4 Le tri simple ou multiple

Nous pouvons demander que le résultat de la requête soit trié sur un ou plusieurs champs. Récupérons la requête précédente dans l'interface graphique, faisons en sorte que le résultat soit trié sur les noms d'abord, sur les prénoms ensuite, et basculons en mode SQL. Nous obtenons :

```
SELECT Personnes.Nom, Personnes.Prénom
FROM Personnes
ORDER BY Personnes.Nom, Personnes.Prénom;
```

Nous voyons que le tri (dans l'ordre croissant) s'obtient grâce à la clause ORDER BY, suivi des noms des champs. Le tri multiple est effectué dans l'ordre d'énumération des champs.

Le tri d'un champ dans l'ordre décroissant s'obtient en faisant suivre le nom de ce champ par **l'opérateur** DESC. L'exemple suivant effectue un tri croissant sur les noms, suivi d'un tri décroissant sur les prénoms :

```
SELECT Personnes.Nom, Personnes.Prénom
FROM Personnes
ORDER BY Personnes.Nom, Personnes.Prénom DESC;
```

18.5 L'élimination des doublons

Comme nous l'avons vu au chapitre 8, la requête simple peut créer des doublons, et il est possible de remédier de façon simple à cette situation en jouant sur les propriétés de la requête. Créons dans l'interface graphique une requête de sélection simple qui concerne le seul champ "Nom" de la table "Personnes". Modifions la propriété "Valeurs distinctes" de "Non" à "Oui", puis basculons en mode SQL. Nous obtenons :

```
SELECT DISTINCT Personnes.Nom
FROM Personnes;
```

Nous voyons que l'élimination des doublons s'obtient à l'aide de l'opérateur DISTINCT placé juste après la clause SELECT. Une syntaxe plus ancienne est également comprise par Access, mais elle ne semble plus guère utilisée :

```
SELECT DISTINCT(Nom)
FROM Personnes;
```

Pour éviter de créer des doublons sur deux champs, la commande SQL s'écrit :

```
SELECT DISTINCT Personnes.Nom, Personnes.Prénom
FROM Personnes;
```

18.6 La requête avec création de champ

Reprenons l'exemple déjà traité au chapitre 8, lequel consiste à concaténer le nom avec le prénom, en les séparant par un espace. Appelons "Nom_complet" le nouveau champ. En mode SQL, nous obtenons :

```
SELECT [Nom] & " " & [Prénom] AS Nom_complet
FROM Personnes;
```

La façon d'extraire le contenu des champs et d'exprimer la concaténation varient d'un SGBD à l'autre. Cependant, la possibilité de créer un nouveau champ (et d'en définir le contenu à partir de champs existants) se retrouve dans tous les SGBD dignes de ce nom.

18.7 La requête multi-fonctionnelle

En définitive, nous pouvons regrouper toutes les opérations précédentes (requête simple, création de table, création de champ, tri et élimination des doublons) en une seule requête, dont voici le code SQL (en version Access) :

```
SELECT DISTINCT [Nom] & " " & [Prénom] AS Nom_complet INTO Liste_de_noms
FROM                                                    Personnes
ORDER BY [Nom] & " " & [Prénom];
```

18.8 Les requêtes emboîtées

Nous avons vu au chapitre 8 qu'il est possible de créer dans Access une requête à partir du résultat d'une autre requête, à condition que cette dernière ne crée pas de table. En mode SQL, la commande s'écrit :

```
SELECT Requête1.Nom
FROM Requête1;
```

On ne peut pas rêver plus simple pour emboîter deux requêtes ! Cette belle simplicité ne se retrouve pas en SQL pur et dur, où l'emboîtement de deux requêtes est d'une écriture plutôt complexe. Que l'on en juge :

si la première requête (encore appelée sous-requête, ou sous-interrogation) ramène une valeur numérique unique (résultat d'une opération du type comptage, sommation, calcul de moyenne, etc.), on utilise les opérateurs arithmétiques usuels : =, <,

>, >=, <= et <> ;

si la première requête ramène une seule ligne, on utilise les opérateurs IN, ALL, ou ANY suivant les cas ;

si la première requête est susceptible de ramener plusieurs lignes, on utilise EXISTS ou NON EXISTS.

Bonjour les erreurs !

18.9 Conclusion

Le chapitre 11 est consacré à la sélection simple, mise en oeuvre à l'aide de l'interface graphique d'Access. Le présent chapitre 19 suit pratiquement le même plan, mais utilise le langage SQL. La comparaison entre ces deux chapitres nous amène à faire une double constatation :

- le langage SQL est d'un usage assez facile, sauf en ce qui concerne l'emboîtement des requêtes. Pour réaliser l'emboîtement, l'utilisation de l'interface graphique d'Access est beaucoup plus simple. Cette situation résulte du fait que le SQL mis en oeuvre par Access permet d'évoquer une requête par son nom, ce qui n'est pas le cas du SQL usuel ;

- Access fournit un moyen didactique commode pour aborder l'étude du langage SQL.

19