

1. Les systèmes de Gestion de Bases de Données	2
1.1 LA BASE DE DONNÉES	2
1.2 LE SYSTÈME DE GESTION DE BASES DE DONNÉES	2
2. L'organisation des données dans la base de données	3
2.1 PRÉSENTATION DE L'EXEMPLE	3
2.2 L'ORGANISATION DES DONNÉES DANS UN SGBD	3
2.3 DIVISER POUR MIEUX RÉGNER	4
2.4 UTILISER PLUSIEURS TABLES	5
2.5 IL EXISTE TROIS TYPES DE RELATIONS ENTRE LES TABLES :	7
2.6 LES INDEX	12
3. Application	13
CREATION D'UNE BASE DE DONNEES AVEC ACCESS	17
3.1 CRÉATION D'UNE TABLE	19
3.2 LES TYPES DE DONNÉES	20
3.3 LES PROPRIÉTÉS DES CHAMPS	22
3.4 LE FORMAT D'AFFICHAGE	24
Pour les champs texte et mémo	24
Pour les champs Numériques et Monétaires	24
Pour les champs de type Date / Heure	26
Format d'affichage des dates / heures dans Windows	28
Pour les champs de type OUI/NON	29
3.5 LE MASQUE DE SAISIE	29
3.6 LES LISTES DE CHOIX	30
3.7 TRI PARMIS LES ENREGISTREMENTS	42
3.8 FILTRER DES ENREGISTREMENTS	43
Le filtre sur un seul critère :	43
Filtre sur plusieurs critères	44
3.9 RECHERCHER DES ENREGISTREMENTS	45

Cours Microsoft ACCESS

1. Les systèmes de Gestion de Bases de Données

1.1 La base de données

Une base de données est un ensemble structuré de données enregistrées sur des supports accessibles par l'ordinateur pour satisfaire simultanément plusieurs utilisateurs de façon sélective et en un temps opportun. Elle doit avoir un certain nombre de caractéristiques :

- ?? **Données structurées** : les informations contenues dans une base de données sont réparties en enregistrements , chaque enregistrement ayant une structure bien définie
- ?? **Données non redondantes** : Une même information ne sera pas répétée plusieurs fois dans la base de données.
- ?? **Données cohérentes** : Il ne doit pas être permis d'enregistrer dans une base des informations incohérentes entre elles
- ?? **Données accessibles** directement selon de multiples critères
- ?? **Indépendance des programmes et des données** : La base de données doit être indépendante des programmes qui y ont accès, on doit pouvoir utiliser un autre programme pour traiter différemment ces données sans avoir à toucher à ces données
- ?? **Sécurité des données stockées** : la base de données doit permettre un système de sécurité permettant de gérer les droits d'accès aux informations par les utilisateurs.

1.2 Le Système de Gestion de Bases de Données

Un Système de Gestion de Bases de Données (S.G.B.D.) représente un ensemble coordonné de logiciels qui permet de décrire, manipuler, traiter les ensembles de données formant la base. Il doit également assurer la sécurité et la confidentialité des données dans un environnement où de nombreux utilisateurs ayant des besoins variés peuvent interagir simultanément sur ces données.

Il doit pouvoir être utilisé par des non-informaticiens. Il doit assurer la définition des structures de stockage et des structures de données et le suivi de leur évolutions ; c'est ce qu'on appelle l'administration des données. Il doit pouvoir au maximum vérifier la cohérence des données. Le SGBD sert donc d'interface entre les programmes d'application des utilisateurs d'une part, et la base de données d'autre part.

Microsoft Access est un SGBD dont nous allons étudier le fonctionnement.

2. L'organisation des données dans la base de données

2.1 Présentation de l'exemple

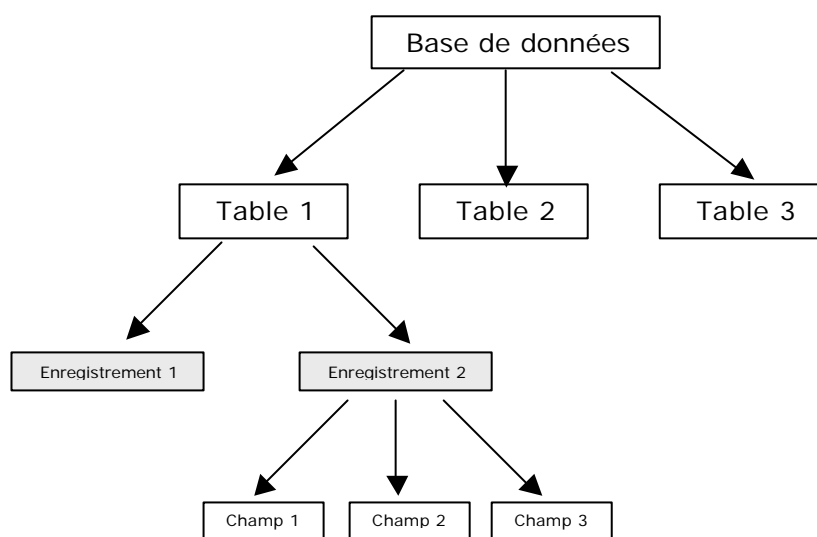
Avant de pouvoir exploiter les données contenues dans la base, il va falloir les modéliser, par modéliser, je veux dire trouver le meilleur moyen de représenter le monde réel en structurant la base de données de manière à pouvoir l'exploiter la plus simplement par la suite.

2.2 L'organisation des données dans un SGBD

Dans Access, comme dans les autres SGBD, les informations doivent être segmentées en données qui sont stockées dans des **tables**. Une **table** est donc un ensemble de données, organisées en lignes et en colonnes. On peut stocker dans une table n'importe quel type d'information (texte, chiffres, graphisme, son, etc...)

Chaque table est divisée en enregistrements, les enregistrements étant l'ensemble des données relatives à la même information. Prenons l'exemple d'un carnet d'adresses. Pour ce carnet d'adresses, nous allons utiliser une table «contacts» qui contiendra toutes les informations concernant chaque contact, chaque enregistrement contiendra les informations relatives à un (et un seul) contact (nom, adresse, téléphone, date de naissance, etc...).

Comme on vient de le voir, chaque enregistrement contient un ensemble d'informations (nom, adresse, téléphone) qui ont elles-même divisé l'enregistrement en plusieurs parties, chaque partie s'appelle un **champ**. Le champ contient une partie des informations de chaque enregistrement.



Organisation de la base de données

Si on reprend notre exemple de carnet d'adresse, nous allons avoir une table qui peut ressembler à ça :

Champ
↓

Nom	Adresse	Téléphone
Deuf John	Rue Barbe – 75001 Paris	01.02.03.04.05
Covert Harry	Rue Minant – 59000 Lille	02.03.04.05.06
Dalors Homère	Rue Tilant – 75003 Paris	03.04.05.06.07

← Enregistrement

Quelques définitions :

?? **La base de données** : C'est l'ensemble des tables utilisées pour gérer l'information

?? **La table** : c'est un ensemble de données organisées en rangées et en colonnes

?? **Le champ (ou colonne)** : C'est l'unité d'information dans une table. Une table est constituée de différents champs

?? **L'enregistrement (ou la ligne)** : C'est l'ensemble des données relatif à la même information (le 1^{er} enregistrement de la table contient les informations relatives au contact « John Deuf »)

2.3 Diviser pour mieux régner

Il faut toujours segmenter l'information en éléments les plus petits possibles. On aurait très bien pu créer une table contact avec un seul champ par enregistrement qui contiendrait toutes les informations en même temps. Mais une des règles d'or dans la création de la structure d'une base de données est de diviser le plus possible l'information pour pouvoir la traiter par la suite le plus simplement et le plus efficacement possible.

Le SGBD, pour retrouver les données qu'on lui demande, travaille avec le contenu des champs, il lui sera donc difficile, dans la table « Contacts » telle qu'elle est de retrouver les contacts habitant à Lille, la ville se trouvant mélangée au reste de l'adresse, de même, on ne pourra pas récupérer le nom seul de chaque contact, celui-ci se trouvant dans le même champ que le prénom.

Comme on ne peut pas toujours prévoir, lors de la création de la structure de la table, de l'usage qui en sera fait plus tard, il faut diviser le plus possible les données, sachant qu'il sera toujours compliqué (et risqué) de modifier la structure de la table par la suite lorsqu'elle sera remplie.

Une meilleure organisation de la table serait la suivante :

Nom	Prénom	Rue	Code Postal	Ville	Téléphone
Deuf	John	Rue Barbe	75001	Paris	01.02.03.04.05
Covert	Harry	Rue Minant	59000	Lille	02.03.04.05.06
Dalors	Homère	Rue Tilant	75003	Paris	03.04.05.06.07

L'organisation de la table est optimale, elle est décomposée en éléments aussi petits que possible, la ville est clairement séparée de l'adresse et le nom du prénom.

2.4 Utiliser plusieurs tables

Prenons l'exemple de la gestion d'une collection de disque. On pourrait être tenté de créer une seule table contenant les informations suivantes :

Collection
Nom Auteur
Prénom Auteur
Titre Album
Année
Genre
Support

Et nous n'aurions pas spécialement tort, toutes les informations relatives à un album s'y trouvent. Pourquoi faire plus compliqué ?

Néanmoins, nous allons voir qu'il est plus judicieux de créer ici deux tables :

Auteurs	et	Albums
Nom Auteur		Titre Album
Prénom Auteur		Année
Date de naissance		Genre
Adresse		Support

Pourquoi ?

On va regrouper ensemble dans une même table toutes les informations relatives au même « sujet ». On voit bien ici qu'on peut regrouper ensemble les données concernant uniquement l'identité de l'auteur et, dans une autre table, les informations relatives uniquement à l'album (par exemple, l'année de sortie d'un album n'a pas de rapport direct avec l'identité de l'auteur, on la trouvera dans une autre table).

Un bon moyen pour trouver les tables utilisées est de partir de la situation existante :

Un *auteur* fait des *albums*, a un auteur peut correspondre plusieurs albums

On en déduit directement une table « auteurs » et une table « albums ».

Ce qui, au passage, nous permet d'enrichir les informations sur les auteurs, on peut en profiter pour par exemple ajouter la date de naissance ou son adresse, ce qui aurait été pénible dans la 1^{ère} version, en effet, il aurait fallu, pour chaque album d'un auteur, ressaisir à chaque coup sa date de naissance et son adresse (sans compter les erreurs de frappe d'un album à l'autre), là les informations ne sont saisies qu'une fois, ce qui est d'abord moins fatiguant (et ça, c'est le plus important) et ensuite, évite les fautes de frappes.

Bon, c'est bien, maintenant on a deux tables, mais du coup, on ne sait plus qui a fait quoi, les informations sont éparpillées. Il va donc falloir trouver un moyen pour savoir exactement à quel auteur correspond chaque album. Pour cela on va ajouter dans la table *auteur* un champ qui va identifier de façon unique chaque enregistrement, ce champ d'identification unique est appelé **clef primaire**.

A quoi sert cette clef? A pouvoir retrouver de façon non équivoque n'importe quel enregistrement dans la table « Auteurs », on ne peut en effet utiliser le nom ou le prénom comme clef primaire car deux auteurs peuvent avoir le même nom (rare mais pas impossible).

Le numéro de sécurité sociale est un bon exemple de champ clef primaire car il identifie de manière unique un individu. Même si deux individus ont le même nom et le même prénom, ils ne peuvent avoir le même numéro de sécurité sociale.

La table « Auteurs » ressemble alors à ceci (la clef primaire est en gras)

Auteurs
N° Auteur
Nom
Prénom
Date de naissance
Adresse

Et pour pouvoir associer un album à un auteur, il va falloir ajouter un champ dans la table « Albums », ce champ contient le n° de l'auteur de l'album

Albums
N° Auteur
Titre Album
Année
Genre
Support

On voit ici que cette organisation répond à une des exigences des bases de données, à savoir la non redondance des informations, dans la première organisation, le nom et le prénom de l'auteur étaient répétés autant de fois qu'ils avait fait d'albums, ici il ne sera présent qu'une seule fois. Lorsqu'on voudra afficher les caractéristiques d'un album, Access, grâce au numéro de l'auteur présent dans l'enregistrement de la table « Albums » ira chercher automatiquement les informations le concernant dans la table « Auteurs ». Cette organisation a un énorme avantage, le jour où on veut modifier le nom ou le prénom d'un auteur, cette modification sera répercutée automatiquement sur les albums qu'il aura fait.

Clef primaire réalisée à partir de plusieurs champs

Lorsqu'on est sûr que le contenu d'une combinaison de champs ne se répétera pas dans une table, on peut utiliser cette combinaison de champs comme clef primaire. On aurait pu, à la place d'un n° d'auteur, utiliser la combinaison « nom + prénom + date de naissance », parce qu'il est rare que deux auteurs différents aient le même nom et le même prénom et la même date de naissance. Mais ce type de combinaison complexe est difficile à maintenir (il est plus simple de taper un numéro qu'une combinaison complexe de noms et de date, sans parler des erreurs de frappes) et en plus prendrait plus de place dans la base de données (on ne voit pas trop l'intérêt de découper les informations en deux tables si c'est pour retrouver dans le n° de l'auteur utilisé dans la table « Albums » quasiment toutes les informations contenues dans la table « Auteurs » !). Et pour terminer, ce genre de clef primaire complexe fonctionne mal dans Access.

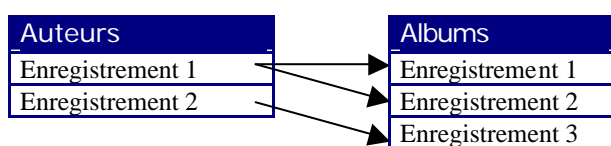
Grâce a cette clef, on va pouvoir mettre les deux tables en relation.

Établir une relation entre les tables de la base de données va permettre de réunir (pour les exploiter ensemble, les analyser, ...) les informations réparties dans différentes tables. La liaison se fait entre les tables à partir de la clef primaire, ici, on obtiendra la liste des albums d'un auteur grâce à la liaison entre la table *Auteur* et la table *Album* sur la clef primaire *n° auteur*.

2.5 Il existe trois types de relations entre les tables :

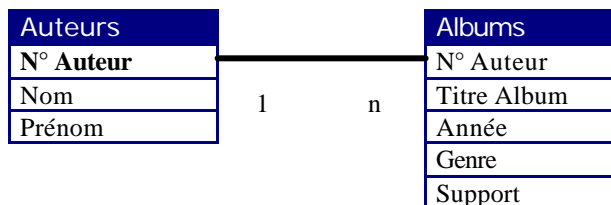
La relation de type « un à plusieurs » (1 – n)

Un enregistrement d'une table est en relation avec plusieurs enregistrements d'une autre table.



Exemple : a un auteur correspond un ou plusieurs albums (plusieurs flèches partent du même auteur), a chaque album ne correspond qu'un seul auteur.

La relation entre les deux tables se présente sous cette forme :



La relation « plusieurs à plusieurs » (n – m)

Un enregistrement de la table primaire peut être en relation avec plusieurs enregistrements de la table reliée et inversement, un enregistrement de la table reliée peut être en relation avec plusieurs enregistrements de la table primaire. Pour gérer ce type de relation, il faut la scinder en deux relations : une relation « un – plusieurs » et une relation « plusieurs – un ».

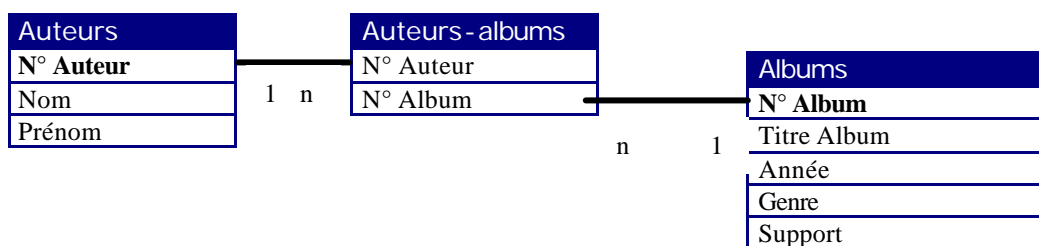
Exemple : On aurait pu avoir le cas où un album a plusieurs auteurs, là, ça ne marche plus, car telle que la table « Albums » est structurée, à un album ne peut correspondre qu'un auteur. Si on veut permettre qu'à un album correspondent plusieurs auteurs (on a alors une relation n – m), deux solutions s'offrent à nous :

1. La plus simple et la plus inélégante (les trucs les plus simples sont souvent inélégants) : ajouter quelques champs « auteurs » supplémentaires dans la table album pour arriver à quelque chose de ce genre, histoire de conserver une relation 1 – n :

Albums
N° Auteur
N° Auteur 2
N° Auteur 3
Titre Album
Année
Genre
Support

Ce genre de plaisanterie est à éviter à tout prix, en effet, si il n'y a qu'un seul auteur, les champs n° auteur 2 et n° auteur 3 ne vont rien contenir, et on va perdre de la place dans la base de données, et que va-t-il se passer si il y a quatre auteurs ?? vous répondrez qu'on peut prévoir quatre champs, ce à quoi je vous répondrez qu'il peut y avoir 5 auteurs, etc...

2. Il va donc falloir (on y coupe pas) créer une troisième table qui va servir d'intermédiaire entre la table « auteurs » et la table « albums » :



J'en entends déjà hurler !

Quelles sont les modifications ?

Nous sommes bien d'accord qu'avec un seul numéro d'auteur dans la table « albums » nous ne pouvons pas avoir plusieurs auteurs pour le même album. La solution 1 étant à proscrire (sauf dans le cas où on était sûr qu'il y aurait toujours 3 et seulement 3 auteurs), la troisième table est la seule solution.

Comment ? et bien, à chaque album correspond n enregistrements dans la nouvelle table « Auteurs-albums », chaque enregistrement renvoyant sur un seul enregistrement de la table « auteurs ». Votre esprit curieux aura noté qu'on a, pour les besoins de la création de cette nouvelle table, créé une clef primaire dans la table « albums ».

Pourquoi ? Pour faire correspondre un enregistrement de la table « Auteurs-albums » avec la table « Albums », il faut le faire sur la clef primaire, et comme on n'avait pas de clef primaire dans « Albums », il a bien fallu en créer une. Aucun des champs présents ne pouvait être utilisé comme clef primaire (la clef primaire, je le répète, doit identifier de façon UNIQUE un enregistrement) : deux albums peuvent avoir le même titre (rare mais pas impossible), peuvent être sortis la même année, être du même genre, etc..., on aurait pu utiliser une clef primaire complexe avec un couple (titre + année), plutôt rare que ce couple se retrouve deux fois, mais pour les raisons vues plus haut, on évitera d'utiliser ce type de clef.

Par exemple, on va avoir dans ces tables :

Auteurs

N° Auteur	Nom	Prénom
1	Crow	Sheryl
2	KS'Choice	
3	Sébastien	Patrick
4	Bezu	

Albums

N° Album	Titre Album	Année	Genre	Support
1	Cocoon Crash	1998	Good	CD
2	The Globe Sessions	1998	Good	CD
3	Tuesday Night Music Club	1993	Good	CD
4	Top NAZ Compil	1920	Debile-Blaireau	CD

Auteurs-Albums

N° Auteur	N° Album
1	2
1	3
2	1
3	4
4	4

Avec cette méthode, on peut affecter sans aucun problème n auteurs à m albums. (ici Sheryl Crow a deux albums, KS' Choice un seul, et l'album Top NAZ Compil a deux auteurs que je ne citerai pas)

Enfin, il existe un troisième type de relation, jamais utilisé: la relation de type «un – un»: un enregistrement d'une table est en relation avec un seul enregistrement d'une autre table et inversement. Ce type de relation ne doit pas se produire car les données de la table reliée n'ont aucune raison de se trouver là, comme elles ne correspondent qu'à un seul enregistrement de la 1^{ère} table, elles devraient être déplacées dans cette table.

Encore un peu de théorie avant de s'amuser avec Accès.

2.6 Améliorations

On peut encore améliorer le schéma de nos tables. En effet, pour la même raison qui nous avait fait séparer en deux tables « Albums » et « Auteurs » notre table de départ pour éviter à avoir à ressaisir les informations concernant un même auteur pour chacun de ses albums, il est ici possible d'effectuer à nouveau une division de la table « Albums ».

En effet, pour chaque album, on va indiquer le genre (rock, pop, reggae, rap etc...) et le support (CD, vinyl, K7, DAT, minidisc, etc...), ces informations vont être redondantes et pourraient être optimisées de la façon suivante :

- 1) On crée une table « genre » et une table « support »

Genre
N° Genre
Nom Genre

Support
N° Support
Nom support

- 2) On modifie la table « albums »

Albums
N° Auteur
N° Auteur 2
N° Auteur 3
Titre Album
Année
N° Genre
N° Support

Maintenant, au lieu d'indiquer explicitement pour chaque album son support et son genre, on indiquera un n° de genre et un n° de support. L'intérêt est triple : gain de place en mémoire (stocker un numéro prend moins de place que stocker une chaîne de caractères), gain de vitesse (entrer un numéro est plus rapide que taper le mot complet), optimisation de la structure (on pourra changer le nom d'un support, ce changement affectera instantanément l'ensemble des albums).

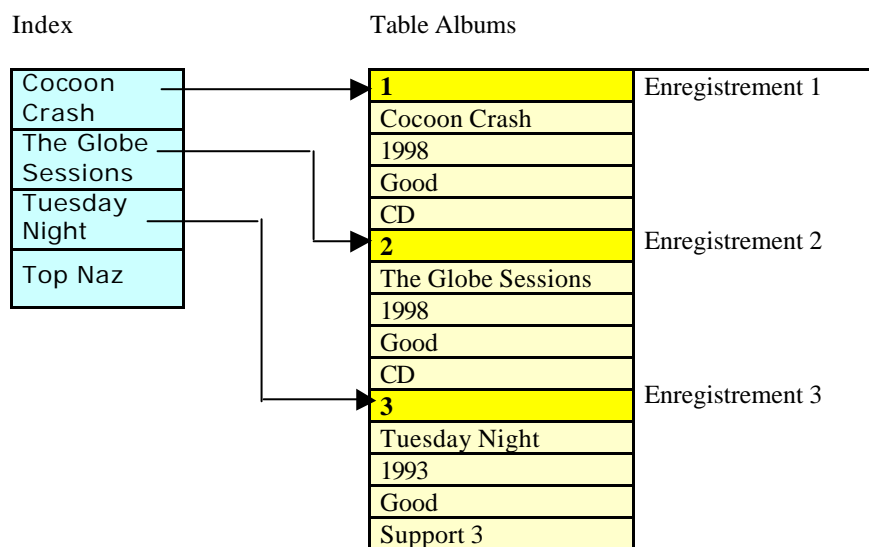
2.7 Les index

Pour optimiser les temps de recherche dans les tables, Access (et d'ailleurs tous les SGBD) ont besoin d'index. L'index permet à Access de retrouver plus rapidement les enregistrements concernés ; il accélère aussi les opérations de tri.

Comment ? Les index fonctionnent avec une table comme la table des matières avec un livre. Lorsqu'on indexe un champ, Access stocke hors de la table les valeurs de ce champ. Lorsqu'on effectue une recherche sur ce champ, plutôt que de le chercher dans la table, Access va faire sa recherche dans l'index, une fois qu'il aura trouvé, il affichera le contenu de l'enregistrement associé à cet index.

Petit crobard :

Supposons que la table « Albums » soit indexée sur le titre de l'album



Lorsqu'on cherchera un enregistrement de la table albums à partir du titre de l'album, Access parcourra l'index et affichera l'enregistrement correspondant à l'index qu'il aura trouvé.

Un index peut être composé d'un ou de plusieurs champs, on aurait pu indexer la table albums sur un index « titre + année » par exemple (si ça peut servir à quelque chose), de même, on peut utiliser plusieurs index pour une table, par exemple, on aurait pu indexer la table album sur les champs Titre, Genre et Support.

Pourquoi alors ne pas tout indexer ? L'idée peut sembler bonne, cela accélérerait les recherches et les tris mais ralentirait toutes les mises à jour de la table, en effet, à chaque fois qu'un champ index est modifié dans la table, il faut le modifier également dans l'index, ce qui peut coûter très cher en temps. On utilisera donc les index avec parcimonie et uniquement si ils s'avèrent utiles.

3. Application

Maintenant que vous avez tout compris au film, nous allons nous pencher sur un autre cas de figure. Supposons qu'on veuille gérer un magasin avec Access. Créez donc les tables et les relations pour notre magasin, au cas où certains ignoreraient le fonctionnement d'un magasin, voici comment ça marche :

- ?? Un client commande des produits
- ?? Un fournisseur livre des produits
- ?? Le stock contient des produits

Et voilà une solution qui semble honnête :

(Les clefs primaires sont en gras, les indexes sont soulignés)

Le choix des index est totalement subjectif, il sera surtout fait en fonction des besoins ultérieurs. Ce n'est pas grave si on n'a pas choisi les bons index, il est toujours possible de les modifier ultérieurement.

Une table « clients »

Clients
N° Client
Titre
<u>Nom</u>
Prénom
Adresse
Code Postal
Ville
Observations

Une table « produits »

Produits
Code produit
<u>Désignation</u>
Prix unitaire
Taux TVA
Stock

Une table « Commandes » qui va indiquer ce qu'a commandé le client

Commandes
N° Commande
<u>N° Client</u>
Date de la commande
Commande réglée

On est d'accord, une commande peut avoir plusieurs lignes (plus d'un produit commandé par commande, ça vaut mieux), pour cela, on va avoir une table « lignes de commande » qui va « relier » la commande aux produits.

Pourquoi n'avoir pas mis ça directement dans la table commande ? on retombe sur notre problème de tout à l'heure : parce qu'on ne sait pas par avance le nombre de lignes qu'on peut avoir dans une commande.

On peut voir qu'un produit peut se retrouver dans n commandes, et que une commande peut faire référence à m articles, on a donc entre les tables « commandes » et « articles », une relation n – m (plusieurs à plusieurs) qu'Access ne peut gérer (ni d'ailleurs aucun autre SGBD), il faut donc transformer cette relation en deux autres relations 1-n et n-1 en utilisant une table supplémentaire, d'où l'apparition de cette table.

Lignes commande
N° Commande
Code produit
Quantité

Pour les fournisseurs, on a quelque chose de très ressemblant

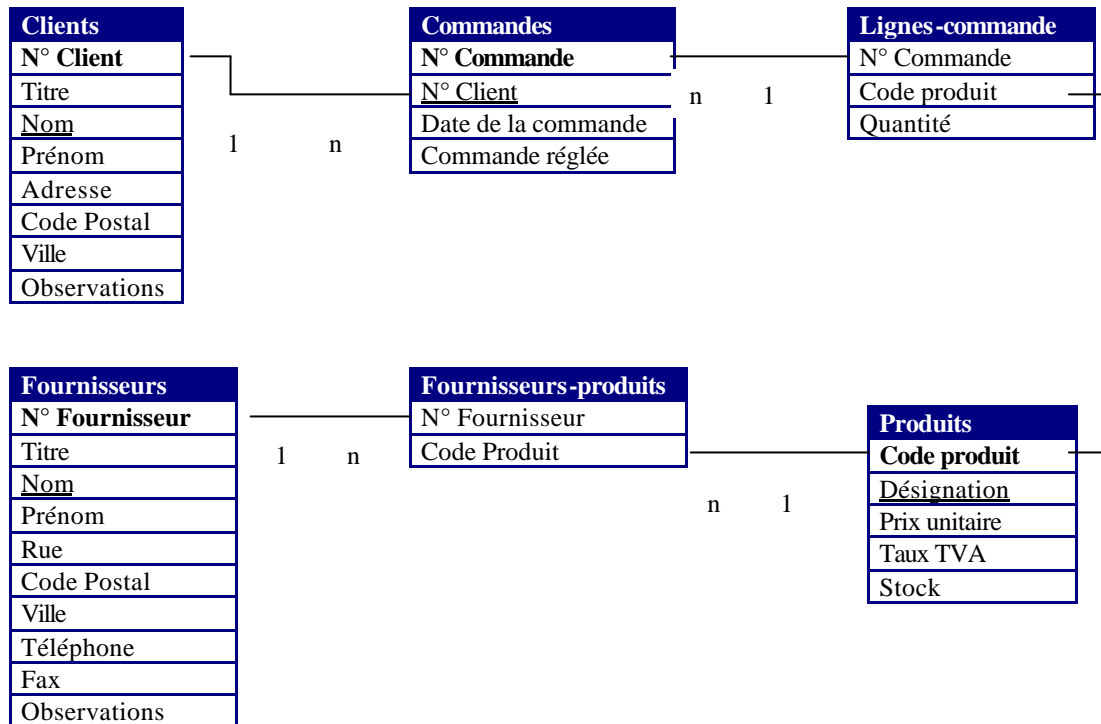
Une table « fournisseurs »

Fournisseurs
N° Fournisseur
Titre
<u>Nom</u>
Prénom
Rue
Code Postal
Ville
Téléphone
Fax
Observations

Même problème que précédemment, un fournisseur peut livrer plusieurs produits et le même produit peut être livré par plusieurs fournisseur, donc relation n-m, donc création d'une table intermédiaire

Fournisseurs-produits
N° Fournisseur
Code Produit

Et voilà le travail d'artiste, les relations entre tout ce beau monde donnent ceci



On aurait pu dire qu'un produit pouvait être livré par un seul fournisseur, dans ce cas, la table intermédiaire « Fournisseurs-Produits » aurait été inutile, pour savoir quel était le fournisseur d'un produit, il suffisait d'ajouter un champ « n° fournisseur » dans la table produit. Il aurait contenu, pour chaque produit, le n° de fournisseur le livrant.

Dernier exemple pour la route

On veut modéliser le problème d'une partie de la gestion de la scolarité d'un établissement.
On connaît les règles suivantes :

- ?? Les enseignements sont dispensés sous forme de cours
- ?? Un étudiant peut s'inscrire à plusieurs cours
- ?? Un enseignant est rattaché à un ou plusieurs établissements et peut enseigner plusieurs cours
- ?? Un cours peut être enseigné par plusieurs enseignants
- ?? Certains enseignants peuvent être responsables des autres

A vous de jouer... Donnez les différentes tables qui vont être utilisées et les relations entre elles (on s'intéresse pas spécialement au contenu exact des tables)

Les tables qui vont être utilisées sont :

Cours
N° cours
Nom
...

Étudiants
N° étudiant
Nom
Prénom
...

Inscriptions
N° étudiant
N° cours
Note semestre
...

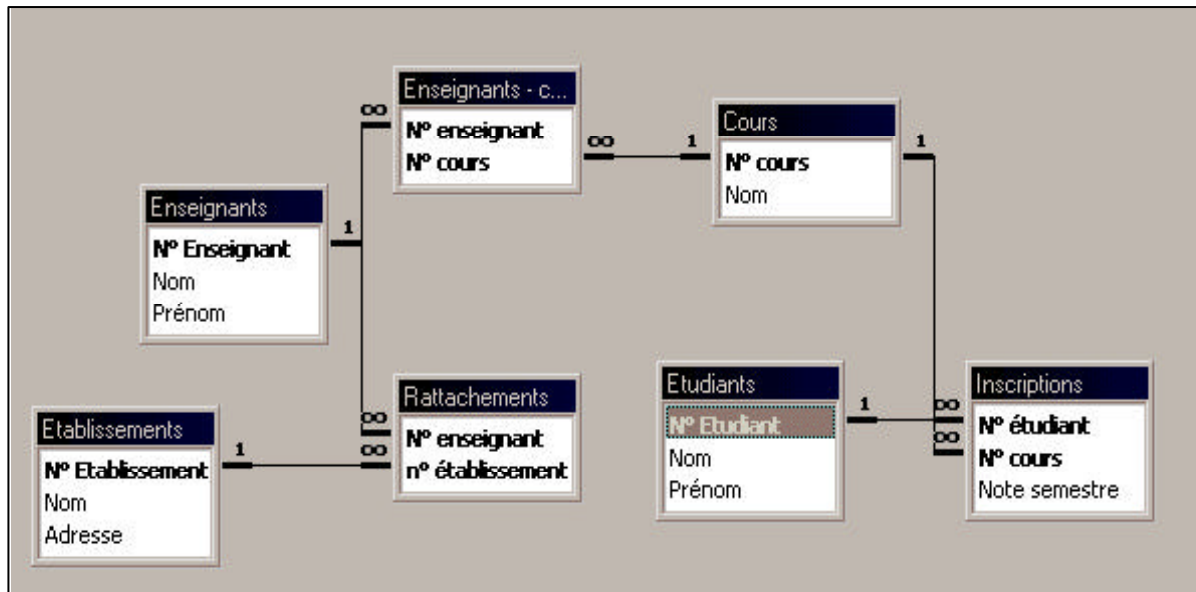
Enseignants
N° enseignant
Responsable
Nom
Prénom
...

Enseignants - cours
N° enseignant
N° cours

Établissements
N° établissement
Nom
Adresse
...

Rattachements
N° enseignant
N° établissement

Les relations vont être :

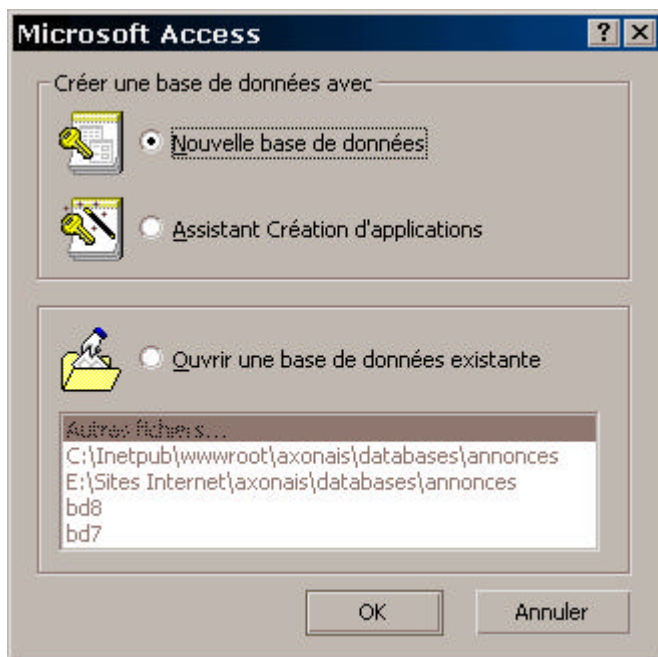


(fait avec Access !)

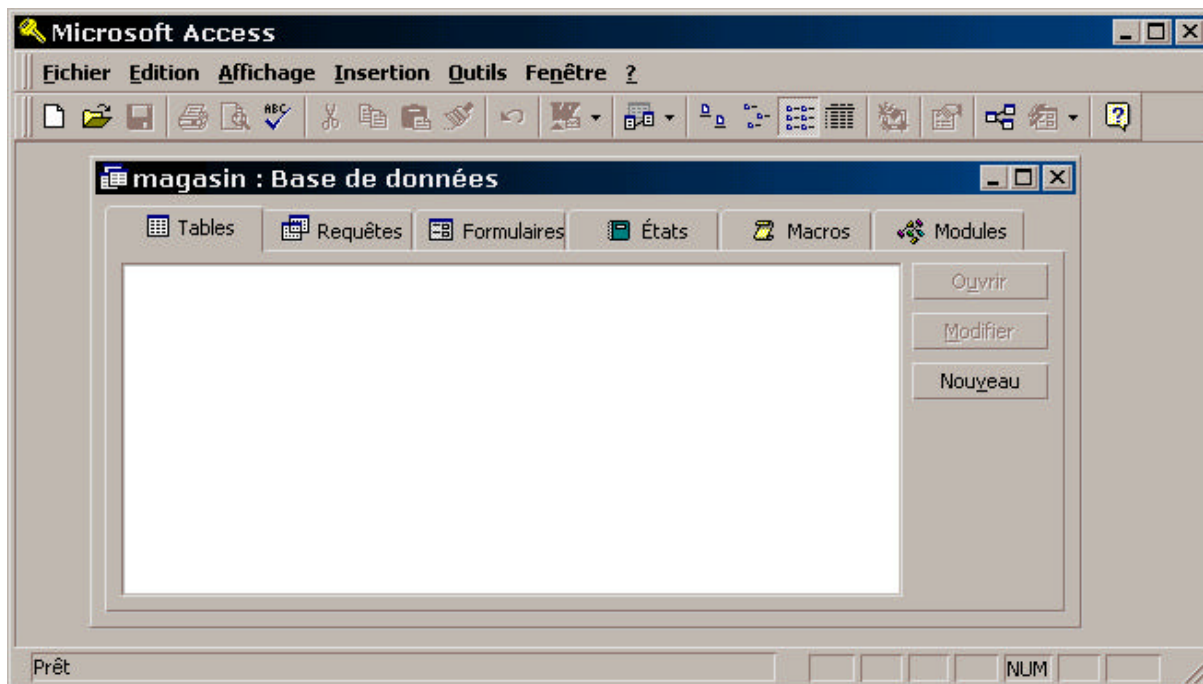
CREATION D'UNE BASE DE DONNEES AVEC ACCESS

Nous allons créer la base de données « Magasin » qui va contenir les tables dont nous avons vu la structure plus haut.

3.1 Création de la base de données



Cliquer sur "Nouvelle base de données" et entrer « magasin.mdb » comme nom de la base.



La fenêtre Base de Données s'affiche, elle contient tous les « objets » qui peuvent composer une base Access :

Les Tables : Chaque table de la base va contenir les données se rapportant à un sujet particulier, la table «clients» va contenir les informations sur les clients. C'est l'objet fondamental de la base de données, toute requête, formulaire ou état va être basé sur une ou plusieurs tables.

Les Requêtes : Les requêtes vont être utilisées pour obtenir des données contenues dans une ou plusieurs tables à partir d'une question. Par exemple, une requête va pouvoir nous afficher la liste des clients habitant à Paris.

Les Formulaires : Le formulaire est utilisé pour faciliter la saisie et la modification des données d'une table, par exemple, le formulaire "client" va permettre d'entrer de façon conviviale les informations concernant un client. On peut les saisir sans formulaire, ce que nous verrons, mais cette méthode est moins conviviale.

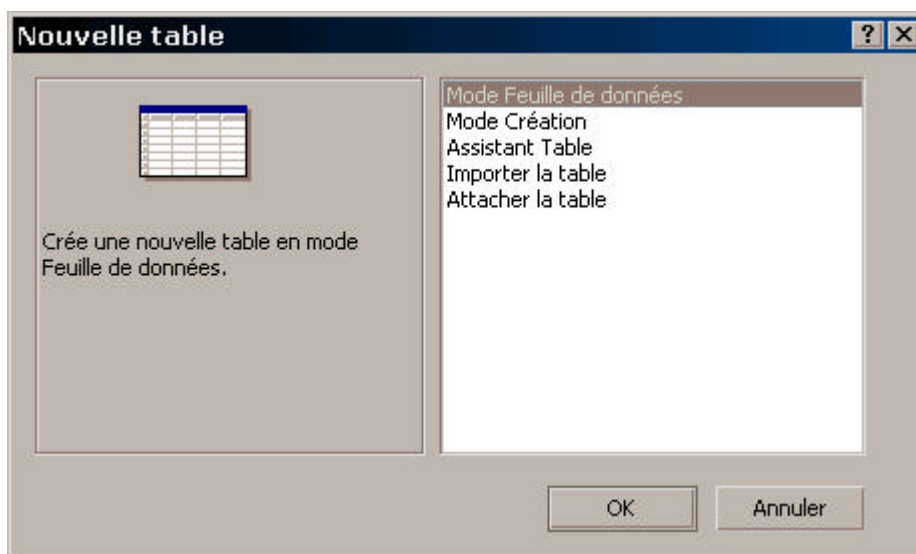
Les États : Les états permettent d'imprimer des données contenues dans des tables selon une présentation définie en y intégrant éventuellement des calculs.

Les Macros : Les macro commandes permettent d'automatiser une suite d'opérations répétitives.

Les Modules : Les modules sont des programmes écrits en Visual Basic for Application (VBA) pour réaliser des opérations qui seraient trop complexes en utilisant les seules fonctionnalités d'Access.

3.1 Création d'une table

Cliquer sur **Nouveau**



Access propose 5 méthodes pour créer une table :

- ?? **Mode feuilles de données** : ce mode permet de saisir directement des informations dans la table sans se préoccuper du nom ou du format des champs, la définition du nom et du format sera faite plus tard.
- ?? **Mode création** : On utilisera cette méthode le plus souvent, elle permet de créer les noms et la structure de chaque champ de la table.
- ?? **Assistant Table** : Access propose différents modèles de tables prédéfinis (carnet d'adresses, factures, etc...) et crée la table, vous devrez la personnaliser ensuite pour qu'elle réponde à vos besoins.
- ?? **Importer la table** : Si les données de votre table sont déjà contenues dans une autre table, dans un fichier ou dans une autre base de données, vous pouvez utiliser cette méthode pour importer vos données dans une nouvelle table.
- ?? **Attacher une table** : Cette option sert à attacher une table avec une autre table contenue dans une autre base de données, nous ne l'utiliserons pas.

Nous utiliserons ici le mode création :

La création de la table s'effectue en remplissant les trois colonnes « **Nom du champ** », « **Type de données** » et « **Description** »

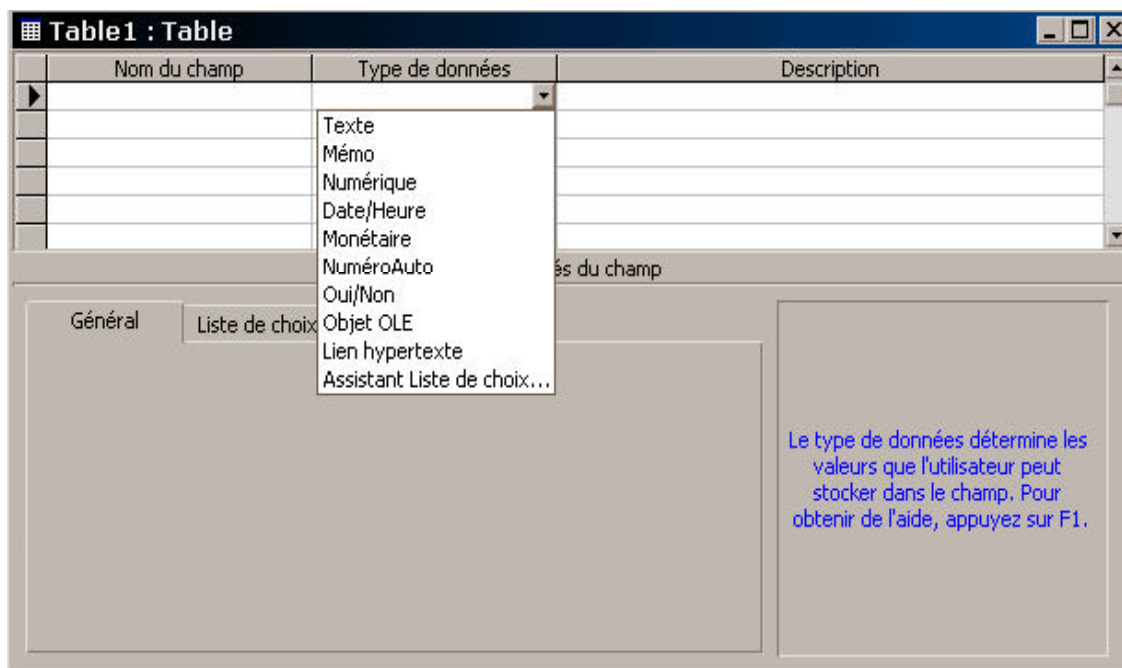
Chaque ligne de ce tableau contient les informations de chaque champ :

- ?? Le nom de chaque champ de la table (de 1 à 64 caractères)
- ?? Le type de données (voir plus bas) à choisir parmi 9 types proposés par Access
- ?? La description du champ (255 caractères maximum)

3.2 Les types de données

Chaque champ peut contenir des données de différents types : le nom du client va contenir des caractères alphabétiques, le code postal va contenir des chiffres, la date de livraison une date, un champ prix va contenir des valeurs monétaires.

Il faut choisir, pour chaque champ de la table, le type de données le plus approprié, en effet, rien n'interdit de choisir comme type de données pour un champ contenant une date ou un code postal un type « texte » ou pour champ contenant un prix un type « numérique », mais autant utiliser les types les plus appropriés !



Les différents types sont :

- ??**Texte** : Pour un champ destiné à contenir des caractères alphanumériques (lettres et/ou chiffres), attention, le champ de type texte ne peut contenir plus de 255 caractères, on l'utilise pour un nom, une adresse, etc...
- ??**Mémo** : Même utilisation que le type texte, mais le champ de type mémo peut contenir jusque 65535 caractères (64 Ko). Utilisé pour une description longue par exemple.
- ??**Numérique** : Le champ ne pourra contenir que des nombres avec ou sans décimales. (un code postal, un n° d'identification par exemple)
- ??**Date/Heure** : Le champ ne pourra contenir que des dates ou des heures (Access vérifie la validité des dates ou des heures saisies)
- ??**Monétaire** : Valeurs présentées sous format monétaire (exemple : 1 23,45F)

- ??**NuméroAuto** Valeur numérique incrémentée automatiquement lors de la saisie de chaque enregistrement. Si vous choisissez ce type de champ, vous ne pourrez rien saisir dedans, c'est Access qui y placera un nombre automatiquement à chaque fois que vous créez un nouvel enregistrement. On l'utilisera lorsqu'on veut être sûr que le contenu de ce champ ne se retrouvera pas dans un autre enregistrement de la table, par exemple pour le numéro de client : chaque nouveau client aura un numéro unique automatiquement affecté par Access, c'est un champ de ce type qui sera le plus souvent utilisé pour être une clef primaire.

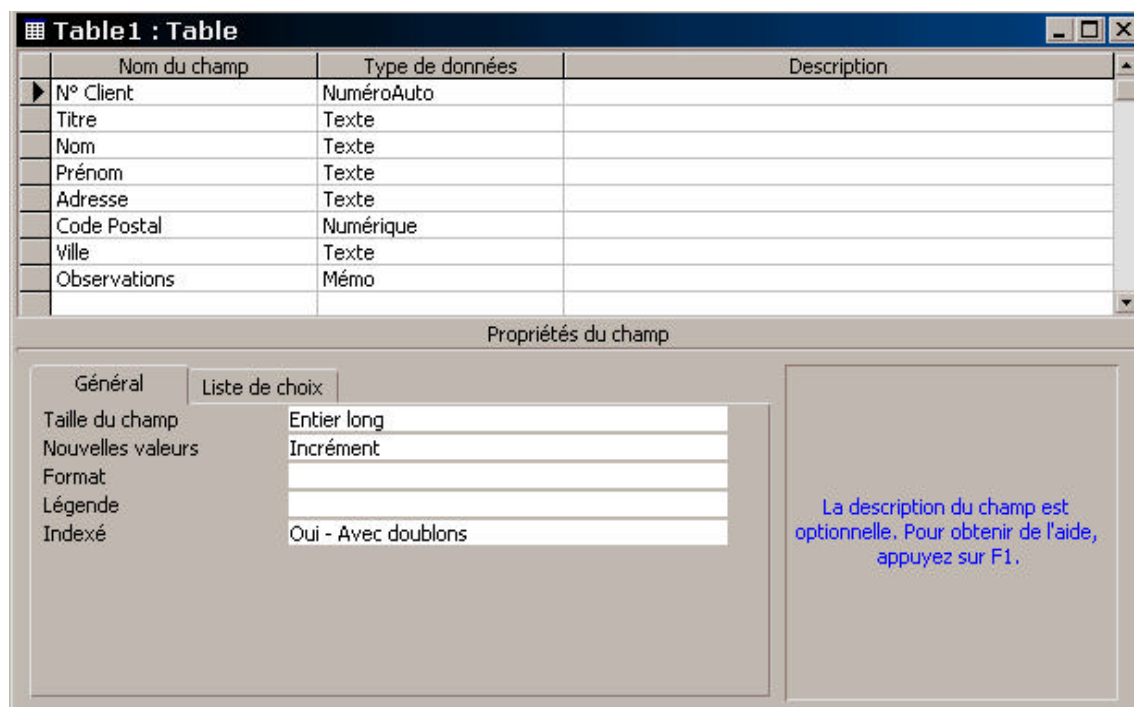
- ??**Oui/Non** : Seules deux données sont autorisées dans ce champ : Oui et Non (on utilisera ce type de données par exemple avec un champ « réglé » qui indiquera si une facture a été réglée ou non)

- ??**Objet OLE** : OLE (Object Linked and Embedded) est une technologie utilisée par Windows, elle permet d'insérer dans une application des objets provenant d'autres applications, on utilisera un champ de ce type pour insérer dans la table une image, un son, un fichier Word, etc...

- ??**Lien Hypertexte** Un champ de ce type contiendra une adresse Internet sur laquelle on pourra cliquer directement.

La liste de choix n'est pas vraiment un type de données, nous l'étudierons plus tard.

Une fois la table « **Clients** » correctement renseignée, on devrait avoir quelque chose qui ressemble à ça :



3.3 Les propriétés des champs

Comme vous pouvez le constater, chaque champ a des propriétés qui dépendent de son type de données, ces propriétés vont déterminer la façon dont les données qu'il contient vont être affichées ou comment les données vont être saisies dans ce champ. Ces propriétés sont affichées dans l'onglet « Général » en bas de la fenêtre. L'onglet « Général » affiche chaque propriété du champ et, à côté, dans des champs modifiables, la valeur de cette propriété.

Ces propriétés varient selon le type de données (par exemple, un champ numérique aura une propriété « décimales » qui indiquent le nombre de chiffres après la virgule que ne peut avoir évidemment un champ de type texte)

Propriétés des champs de type Texte et Mémo	
Taille du champ	Nombre de caractères maximum qui peuvent être saisis
Valeur par défaut	Texte qui sera contenu par défaut dans ce champ
Valide si	Expression pour valider la saisie, exemple : <> « machin », interdira la saisie du mot « machin » dans ce champ
Message si erreur	Message d'erreur qui va être affiché si l'expression saisie n'est pas valide, dans notre exemple, on pourrait mettre ici «saisie de machin interdite »
Null Interdit	Si oui, il faudra obligatoirement saisir une valeur dans ce champ
Chaîne vide autorisée	La chaîne vide est "", on peut l'utiliser lorsqu'on n'a rien à entrer dans un champ obligatoire, si cette propriété est non, il faudra obligatoirement saisir du texte dans ce champ
Indexée	Le champ est un index avec ou sans doublons, si l'index n'a pas de doublons, il ne sera pas possible de saisir deux fois le même contenu pour ce champ dans la table
Format	Va définir comment le contenu du champ va être affiché (voir plus loin)
Masque de saisie	Oblige à saisir le contenu du champ selon un format précis (par exemple un numéro de téléphone), on verra ça plus loin

Tous les autres types de données ont des propriétés similaires, certains ont des propriétés supplémentaires :

Propriétés des champs de type numérique	
Taille du champ	<p>Définit la valeur maximale qui va pouvoir être contenue dans ce champ. Il faudra faire attention et bien penser à l'avance quelle sera la valeur maximale permise dans ce champ pour pouvoir ajuster au mieux cette propriété :</p> <p><i>Octet</i> : de 0 à 255 <i>Entier</i> : de -32767 à +32768 <i>Long</i> : de -2147483648 à +2147483649 <i>Réel Simple</i> : de $-3.48^{E}38$ à $+3.40^{E}38$ <i>Réel Double</i> : de $-1.79^{E}308$ à $+4.96^{E}324$</p> <p>Plus on descend, plus le champ occupera de la place en mémoire et sur le disque dur (le Réel Double prend 8 fois plus de place que l'octet), pensez-y si la table contient des milliers de lignes...</p>
Décimales	Nombres de chiffres après la virgule

Propriétés des champs de type NuméroAuto	
Nouvelles valeurs	Indique quelle sera la prochaine valeur contenue dans un champ de type NuméroAuto : avec Incrément , la valeur sera égale à la valeur créée dans le précédent enregistrement+1, avec Aléatoire , Accès remplira ce champ avec une valeur tirée au hasard (mais qui ne se retrouvera pas dans un autre enregistrement).

3.4 Le format d'affichage

Nous avons vu plus haut qu'on peut modifier la façon dont les données contenues dans les champs peuvent être affichées ou forcer leur saisie selon un format précis.

La façon dont les données sont affichées se fait par l'intermédiaire de la propriété "Format d'affichage".

Attention, la façon dont les données sont affichées dans un champ ne modifie en rien le contenu de ce champ dans la table. Par exemple, si vous forcez à afficher un '1' avant le contenu d'un champ, si ce champ contient '234', la valeur affichée sera '1234' mais le contenu du champ dans la table sera toujours '234'. La propriété 'format d'affichage' n'influe que sur la façon dont les données d'un champ vont être affichées.

Pour modifier le format d'affichage, on utilise les codes suivants (quand ils marchent...) :

Codes du format d'affichage	
"texte"	Affiche le texte "texte" dans le champ
!	Justifie à droite le contenu du champ
*c	Remplace les espaces par le caractère c
[couleur]	Affiche le contenu du champ dans la couleur "couleur"

Il y a des formats d'affichages plus spécifiques à certains types de données :

Pour les champs texte et mémo

Formats d'affichage spécifiques aux champs Texte et Mémo	
@	Va afficher un espace si on a rien saisi
<	Va afficher le texte en minuscules
>	Va afficher le texte en majuscules

Pour les champs Numériques et Monétaires

Formats d'affichage spécifiques aux champs Numériques et Monétaires	
Nombre général	Affiche le nombre tel qu'il a été saisi
Monétaire	Nombre avec séparateur de milliers + 2 chiffres après la virgule + symbole monétaire
Fixe	Affiche au moins un chiffre + 2 chiffres après la virgule
Standard	Nombre avec séparateur de milliers + 2 chiffres après la virgule
Pourcentage	Multiplie le nombre par 100 et ajoute le symbole %
Scientifique	Nombre au format décimal avec exposant

On peut définir son propre format d'affichage pour les champs numériques ou monétaires si le format que l'on désire ne fait pas partie des formats proposés ci-dessus.

Pour cela, on utilise un format d'affichage spécial, ce format est défini par une chaîne de caractères qui a le format suivant (vous tapez cette chaîne de caractères dans le champ 'format' de la propriété du champ) :

; ; ;
 >0 <0 =0 NULL

La première partie (avant le premier symbole ";") va définir comment va être l'affichage si le contenu du champ est supérieur à 0, la deuxième partie, si le contenu du champ est inférieur à 0, la troisième partie si le contenu du champ = 0 et la dernière partie si le contenu du champ est vide.

Chaque partie est une combinaison des symboles suivants :

Symbole	Signification
,	Séparateur de milliers
.	Séparateur décimal
0	Un chiffre ou zéro
#	Un chiffre ou rien
%	Pourcentage (* 100) + symbole %
E-	Affichage scientifique, exposant positif non signé
E+	Affichage scientifique, exposant positif signé

Par exemple :

\$#,##0.00[vert];(\$#,##0.00)[rouge];"Zéro";"Vide"

Que va-t-il se passer avec ce format d'affichage ?

- ?? Si le nombre est supérieur à 0, il sera affiché en vert avec au moins un chiffre avant la virgule et deux après la virgule.
- ?? Si il est inférieur à zéro, il sera affiché en rouge, entre parenthèses avec au moins un chiffre avant la virgule et deux après la virgule.
- ?? Si il est égal à zéro, il sera affiché "Zéro".
- ?? Si le champ est vide, il sera affiché "Vide".

Notez que les symboles # et 0 symbolisent chaque chiffre du nombre qui doit être affiché.

Pour les champs de type Date / Heure

Formats d'affichage spécifiques aux champs Date / Heure	
Date, général	Par défaut, affiche la date sous le format JJ/MM/AA HH:MM:SS (si il n'y a pas d'heure, affiche seulement la date)
Date, Complet	Affiche la date complète comme prévue dans le panneau de configuration de Windows Par exemple : lundi 1 janvier 1999
Date, Réduit	Affiche : 1-jan-99
Date, Abrégé	Affiche la date abrégée comme prévue dans le panneau de configuration de Windows Par exemple : 1/1/99
Heure, Complet	Affiche l'heure comme prévue dans le panneau de configuration de Windows Par exemple : 17:34:22
Heure, Réduit	Exemple : 5:34 PM
Heure, Abrégé	Exemple : 17:34

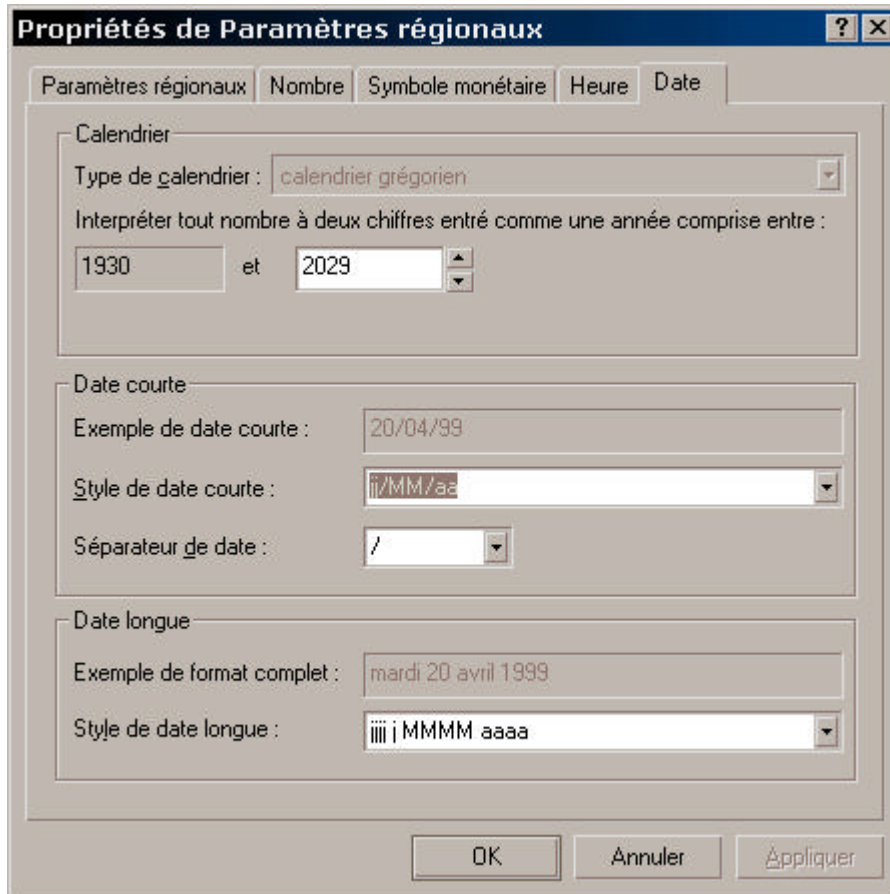
Comme pour les champs de type numériques ou date, on peut définir son propre format d'affichage en utilisant une combinaison des symboles suivants :

Symbole	Signification
:	Séparateur d'heure
/	Séparateur de date
c	Identique au format "Date, Général"
j	Jour du mois (1 à 31)
jj	Jour du mois sur deux chiffres (01 à 31)
jjj	Trois premières lettres du jour (dim à sam)
jjjj	Nom entier du jour de la semaine (dimanche à samedi)
jjjjj	Identique au format "Date, Abrégé"
jjjjjj	Identique au format "Date, Complet"
e	Jour de la semaine (1 à 7)
ee	Semaine de l'année (1 à 53)
m	Mois de l'année (1 à 12)

mm	Mois de l'année sur 2 chiffres (01 à 12)
mmm	Trois premières lettres du mois (jan à déc)
mmmm	Nom entier du mois (janvier à décembre)
t	Trimestre (1 à 4)
A	Jour de l'année (1 à 366)
aa	Deniers chiffres de l'année (00 à 99)
aaaa	Année complète (0100 à 9999)
h	Heure (0 à 23)
hh	Heure sur 2 chiffres (00 à 23)
n	Minutes (0 à 59)
nn	Minutes sur 2 chiffres (00 à 59)
s	Secondes (1 à 59)
ss	Secondes sur 2 chiffres (00 à 59)
Tttt	Identique au format "Heure, Complet"
AM/PM	Heure sur 12 heures + AM ou PM
am/pm	Heure sur 12 heures + am ou pm
A/P	Heure sur 12 heures + A ou P
a/m	Heure sur 12 heures + a ou p
AMPM	Heure sur 12 heures + indicateur matin/après-midi défini dans le panneau de configuration

Format d'affichage des dates / heures dans Windows

Pour définir le format d'affichage par défaut dans Windows, allez dans le panneau de configuration et double cliquez sur l'icône "**paramètres régionaux**" (une planète habituellement) :



Il y a deux onglets : Date et Heure qui définissent comment vont s'afficher la date et l'heure dans tous les logiciels Windows.

Pour les champs de type OUI/NON

Access propose pour ce type de champ, trois types d'affichage par défaut : à la place de OUI ou NON, on peut afficher "Vrai" ou "Faux" ou alors "Actif" ou "Inactif", si on veut afficher un autre message, il faut dans le champ format de la propriété du champ taper la commande suivante :

```
; "texte 1" ; "texte 2"
```

"texte 1" sera affiché à la place de Oui et "texte 2" à la place de Non.

On peut, bien sur, utiliser des codes d'affichages supplémentaires, par exemple :

```
; "OK" [vert] ; "NON" [Rouge]
```

affichera OK en vert à la place de Oui et NON en rouge à la place de Non.

N'oubliez pas que la ligne commence par le symbole ";"

3.5 Le Masque de saisie

Le masque de saisie permet de faciliter la saisie dans un champ en forçant l'utilisateur à entrer les données selon un format déterminé, les informations stockées dans la table auront le format défini dans le masque de saisie contrairement au format d'affichage qui affectait seulement la façon dont les données contenues dans un champ allaient être affichées.

Le masque de saisie, comme le format d'affichage est constitué d'une combinaison de caractères :

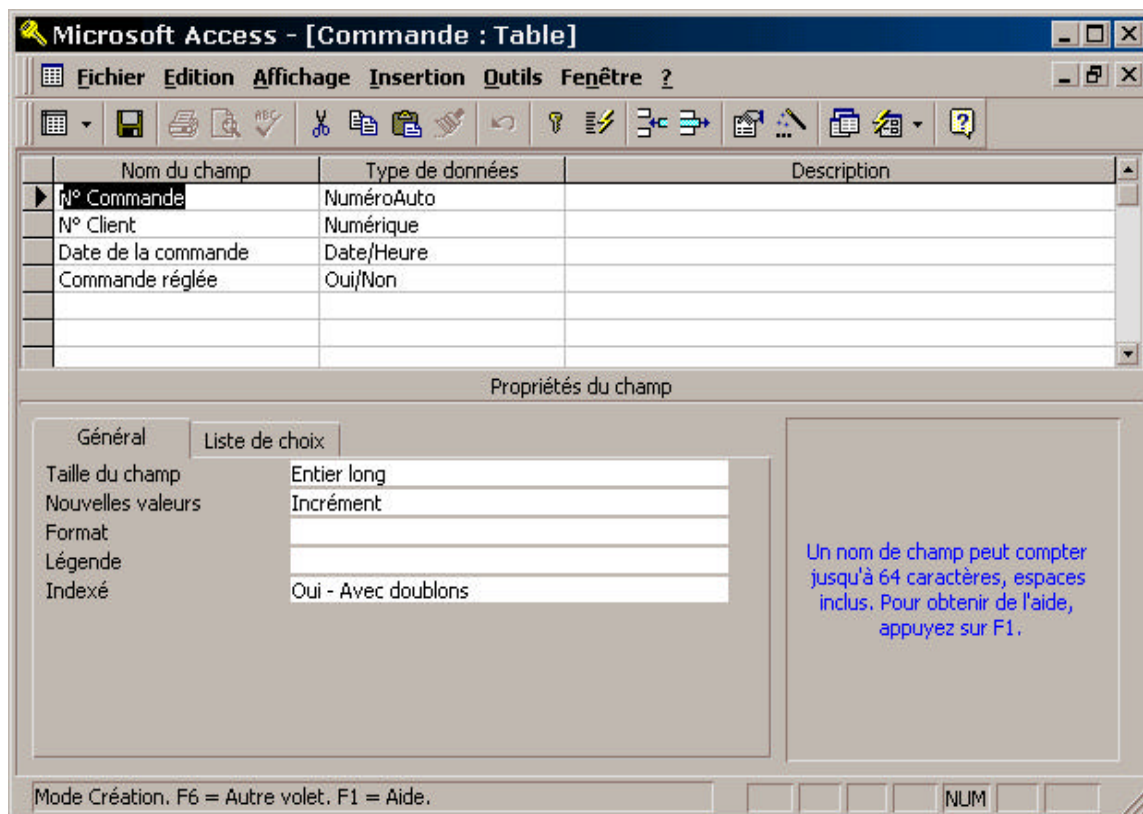
Symbole	Signification
0	Chiffre de 0 à 9 obligatoire
9	Chiffre ou espace facultatif
#	Chiffre ou espace ou + ou -
L	Lettre de A à Z obligatoire
?	Lettre de A à Z non obligatoire
A	Lettre ou chiffre obligatoire
a	Lettre ou chiffre facultatif
&	Caractère quelconque obligatoire
C	Caractère quelconque facultatif
<	Passe en minuscules
>	Passe en majuscules
!	Saisie à partir de la droite

Par exemple, si on veut saisir 5 chiffres obligatoirement pour un code postal, on utilisera : 00000, si on veut saisir une suite de 3 chiffres et de 3 lettres : 000LLL, si on veut saisir un numéro de téléphone : 00-00-00-00-00, ou un nom de famille dont la première lettre est toujours en majuscules : >L<???????????? (prévoir autant de ? que le nom peut comporter de lettres)

3.6 Les Listes de choix

Nous avons vu précédemment que parmi les types de données est proposé un type de données particulier : la liste de choix, la liste de choix n'est pas à proprement parler un type de données particulier, c'est un moyen de simplifier la saisie de données dans une table en proposant à l'utilisateur de cliquer sur un élément proposé dans une liste, le champ de la table sera rempli avec l'élément sélectionné dans la liste.

Pour pouvoir tester cela, nous allons créer la table "Commandes"



Elle doit ressembler à ça.

Nous allons maintenant retourner dans la table client et modifier le type de données du champ "Titre", Le champ "Titre" va contenir le titre du client : Monsieur, Madame, Mademoiselle. Au lieu de saisir à chaque fois le mot en entier, nous allons créer une liste de choix qui proposera ces trois titres, il suffira de cliquer sur l'un d'entre eux pour remplir automatiquement le champ.

Après avoir choisi le type "Liste de Choix" pour le champ "Titre", la fenêtre suivante s'affiche

Assistant Liste de choix

Cet Assistant crée une liste de choix, qui affiche une série de valeurs que vous pouvez sélectionner.

Comment souhaitez-vous que votre liste de choix obtienne ces valeurs ?

Je veux que la liste de choix recherche les valeurs dans une table ou requête.

Je taperai les valeurs souhaitées.

Buttons: Annuler, < Précédent, Suivant >, Terminer

Il y a deux façons de créer une liste de choix : soit à partir du contenu d'une autre table (ce que nous verrons après), soit à partir d'une liste que l'on va taper soi-même une fois pour toutes, c'est ce que nous allons choisir. (Cliquez ensuite sur Suivant)

Assistant Liste de choix

Quelles valeurs souhaitez-vous dans votre liste de choix? Entrez le nombre de colonnes souhaité dans la liste, et tapez les valeurs souhaitées dans chaque cellule.

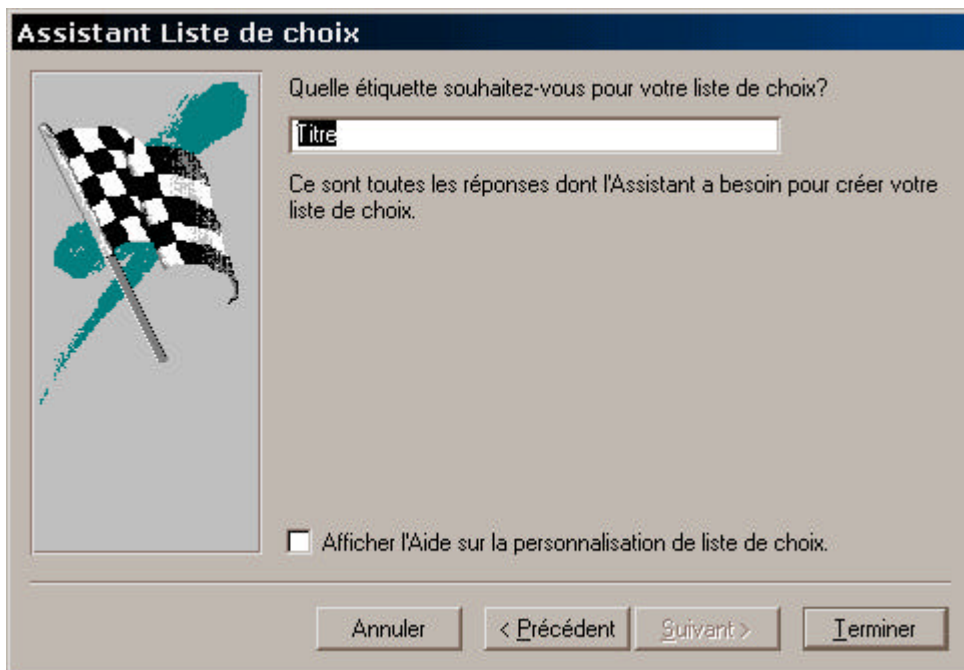
Pour ajuster la largeur d'une colonne, déplacez le bord droit jusqu'à la largeur souhaitée, ou cliquez deux fois sur ce bord droit de la colonne afin d'obtenir la meilleure largeur.

Nombre de colonnes:

	Col1
	Madame
	Mademoiselle
	Monsieur

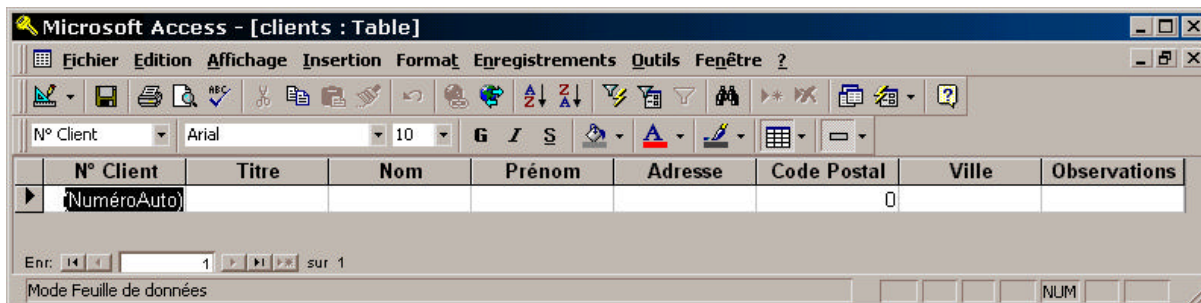
Buttons: Annuler, < Précédent, Suivant >, Terminer

On va taper dans la colonne Col1 la liste des titres possibles et cliquer sur Suivant une fois terminé.

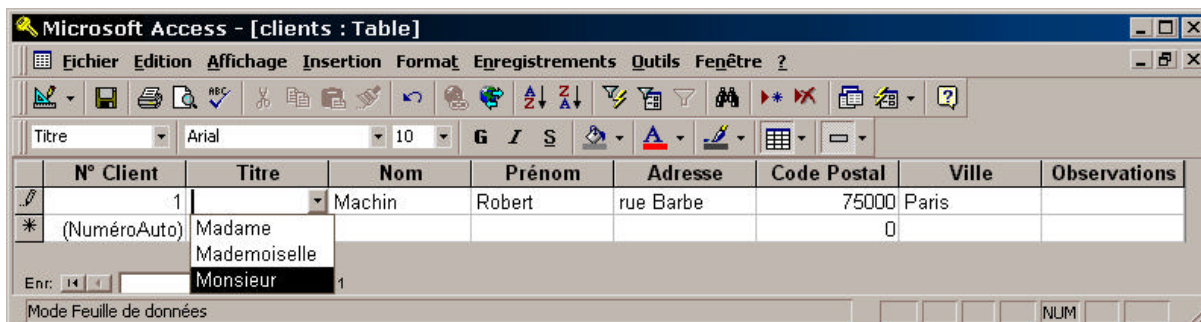


On nomme ici la liste de choix (peu importe le nom choisi) et on clique sur "Terminer"

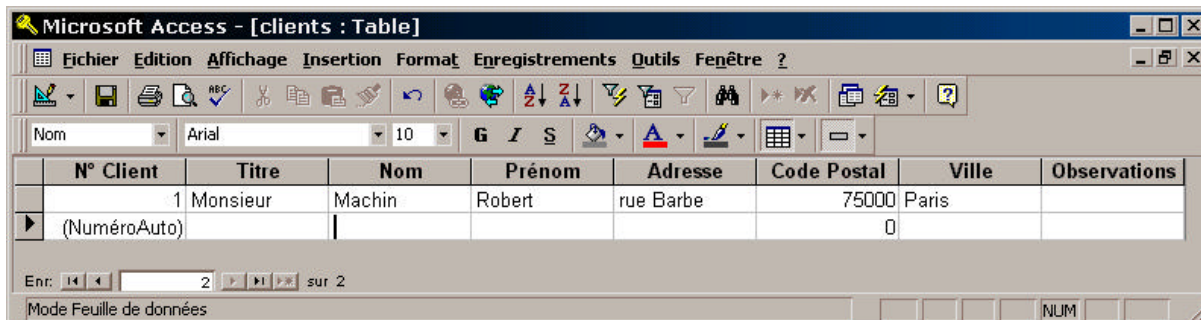
Maintenant, pour voir le résultat, nous allons saisir des données dans la table "Clients", dans la fenêtre où se trouve la liste des tables, on clique sur "Ouvrir", on arrive alors sur un tableau qui ressemble à ça :



Il y a une ligne par enregistrement, et une colonne par champ, on se positionne sur le champ désiré en cliquant dessus et on saisi directement les données.

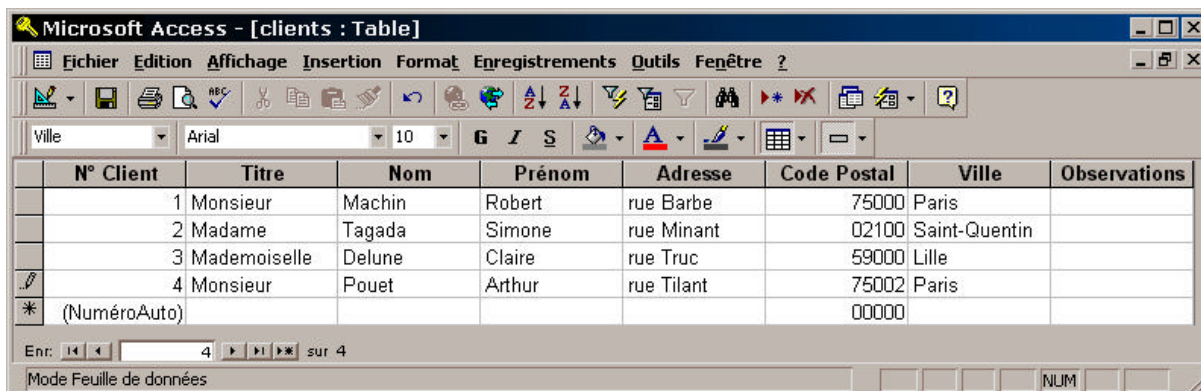


Et que se passe-t-il lorsqu'on clique sur "Titre" ? La liste de choix apparaît, on peut choisir ici un des titres proposés en cliquant dessus directement.



On peut voir que "Monsieur" a été entré directement dans le champ "Titre".

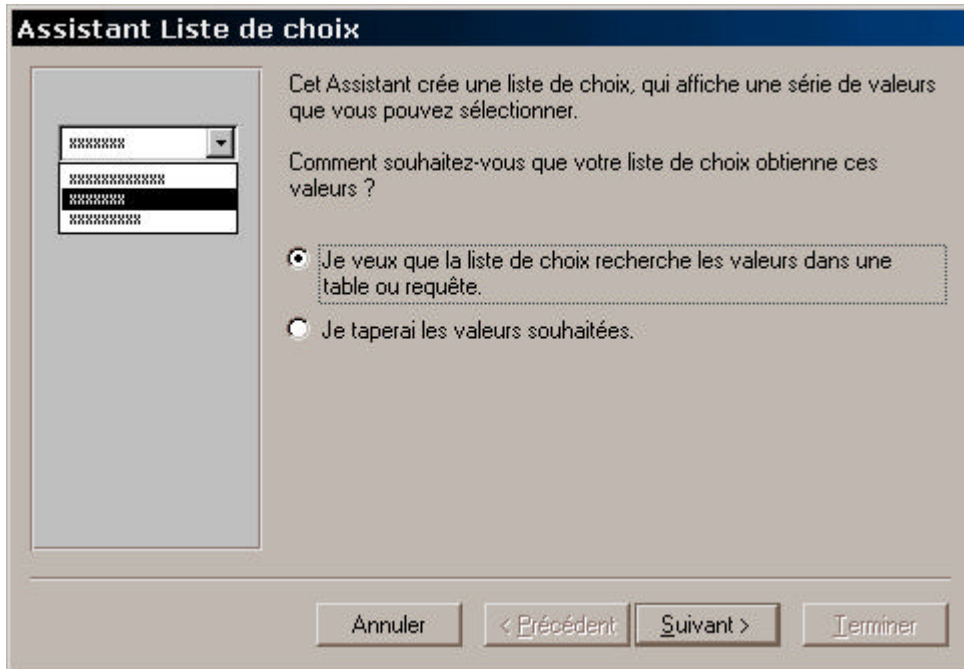
On va ainsi créer plusieurs clients :



Une fois ces clients saisis, nous allons aller modifier le type de données du champ "N° Client" de la table commande pour y mettre une liste de choix.

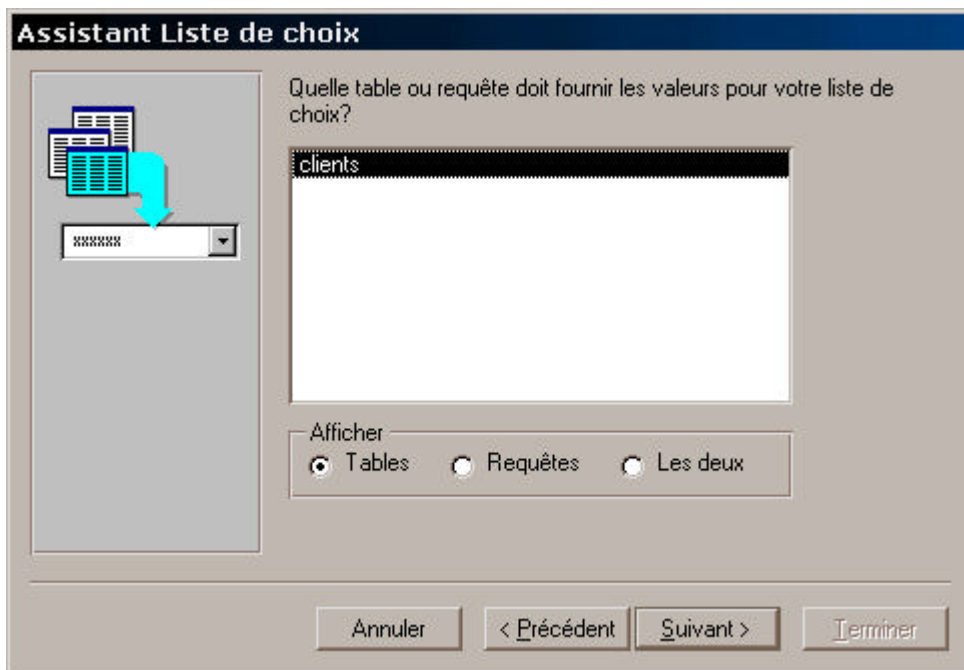
Pourquoi ? Une commande contient le numéro du client qui l'a passé, lorsque l'on saisi une commande et qu'on a que le nom du client, il faut se souvenir de son numéro pour le taper, la liste de choix va ici proposer la liste de tous les clients se trouvant dans la table "Clients", il suffira de cliquer sur le nom de l'un d'entre eux pour que son numéro soit automatiquement entré dans le champ "N° Client" de la table "Commande".

Voici comment faire :

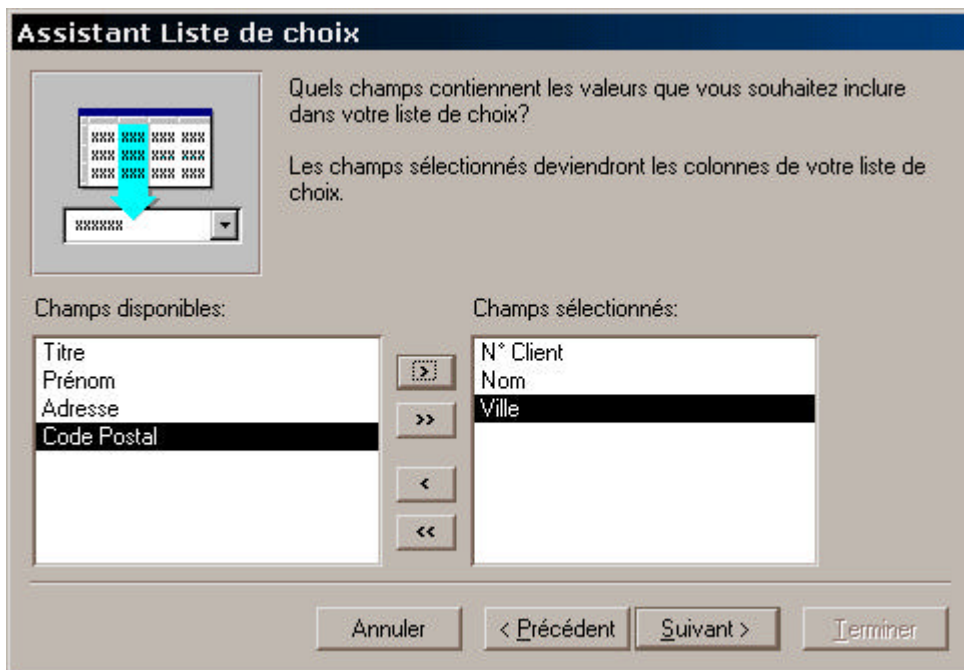


On change le type de données du champ "N° Client" dans la table "Commande" et on choisit "**Liste de Choix**". Cette fois-ci, nous n'allons pas saisir tous les éléments de la liste, ça n'aurait pas de sens, il faudrait remodifier cette liste à chaque fois qu'on ajoute un nouveau client. Nous allons dire à Access de construire sa liste de choix en allant chercher les éléments de cette liste dans une autre table, pour ça on clique juste sur "**Suivant**".

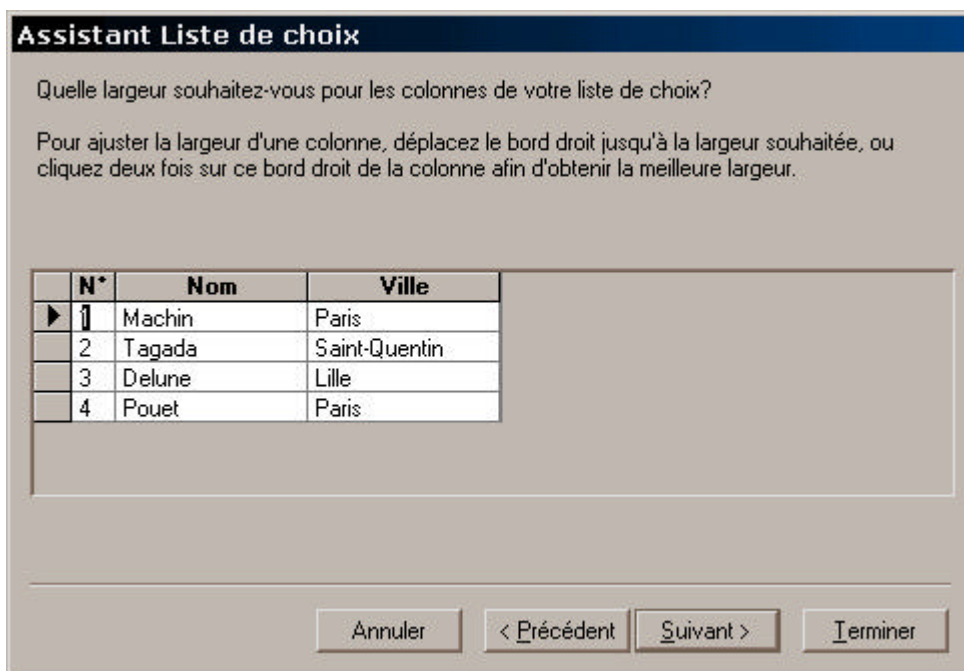
Access nous affiche les autres tables de la base :



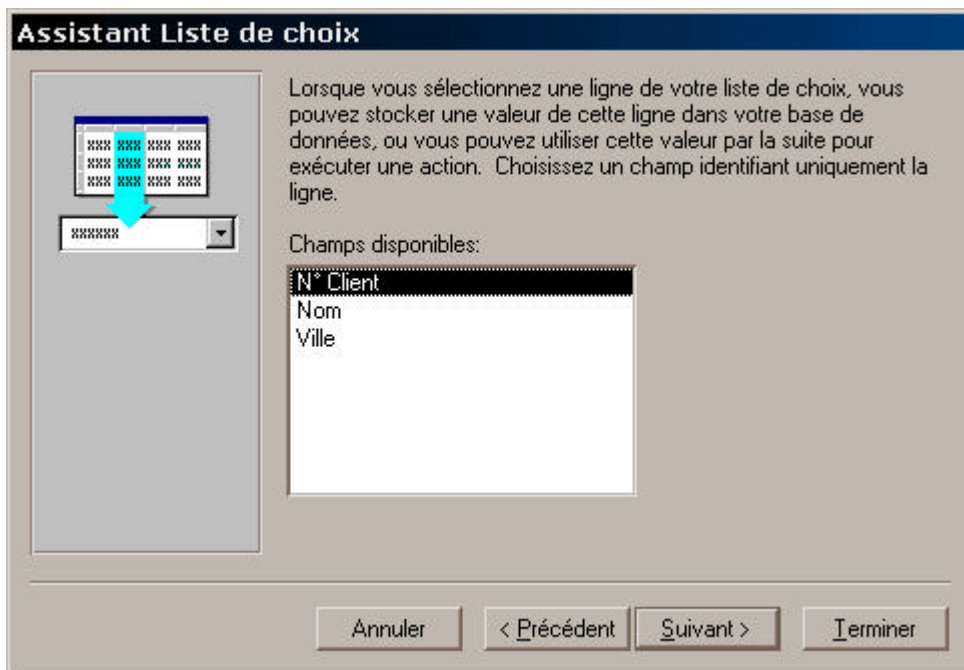
On choisit "Clients" (!) et on clique sur **Suivant**



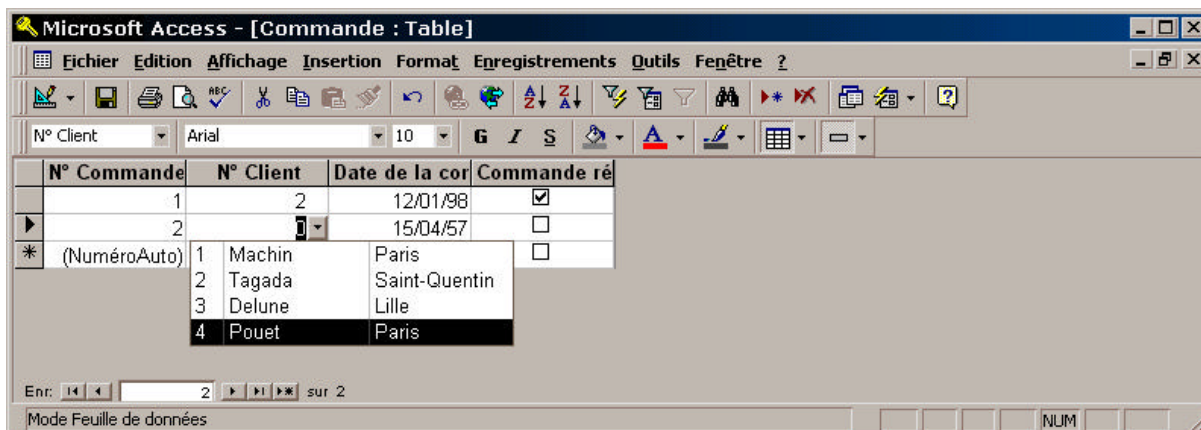
Access nous demande ici quels vont être les champs de cette table qui vont être affichés dans la liste, nous n'avons besoin que du numéro du client, de son nom et de sa ville (au cas où il y aurait deux clients homonymes). On sélectionne à gauche les champs que l'on désire voir apparaître dans la liste, et on clique sur ">" pour les ajouter, sur "<" pour les retirer, ">>" et "<<" ajoutent et retirent tous les champs.



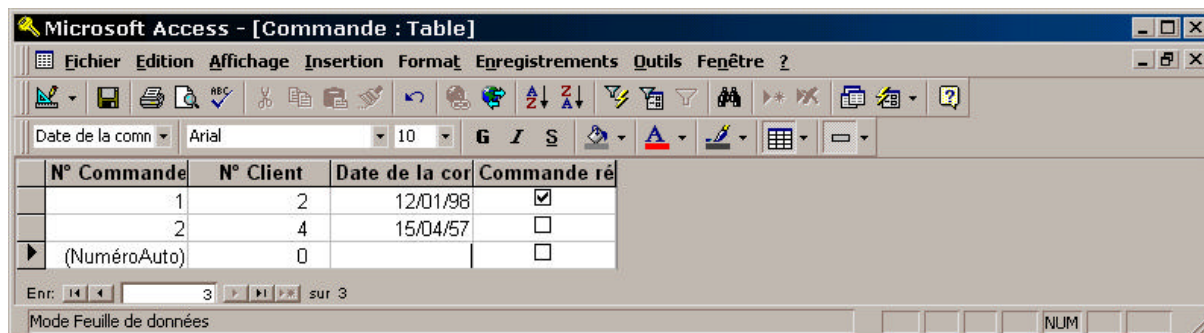
Access affiche un exemple du contenu de la liste. Vous pouvez ici dimensionner la taille des colonnes.



Nous avons choisi d'afficher trois champs de la table "Clients" dans notre liste de choix, Access nous demande lequel de ces trois champ va servir à initialiser le champ "N° Client" de la table "Commande", il est clair que nous voulons que le champ "N° Client" de la table "Commande" soit initialisé avec le n° de client du client que nous aurons sélectionné dans la liste, on choisit donc "n° Client" et on clique sur Suivant, on nomme ensuite la liste, et on peut commencer à saisir des commandes :



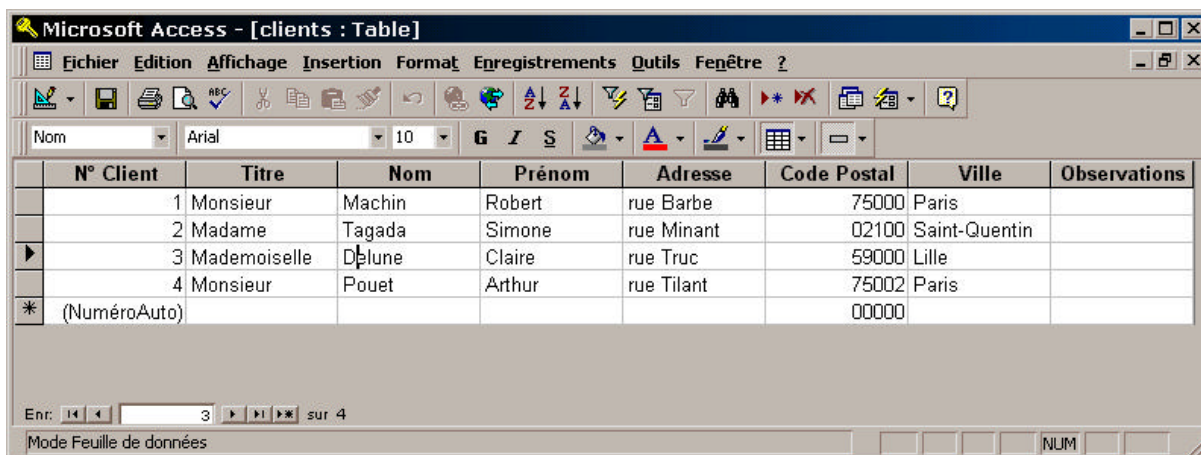
Que se passe-t-il lorsqu'on se positionne sur le n° de client ? Access va lire toute la table "Clients", et créer une liste de choix à partir de cette table. Cette liste contient les champs que nous lui avons indiqué. Il suffit de sélectionner un client dans la liste, et comme nous lui avons dit d'initialiser le champ N° Client de la table "Commande" avec la première colonne de la liste de choix, son numéro sera automatiquement copié.



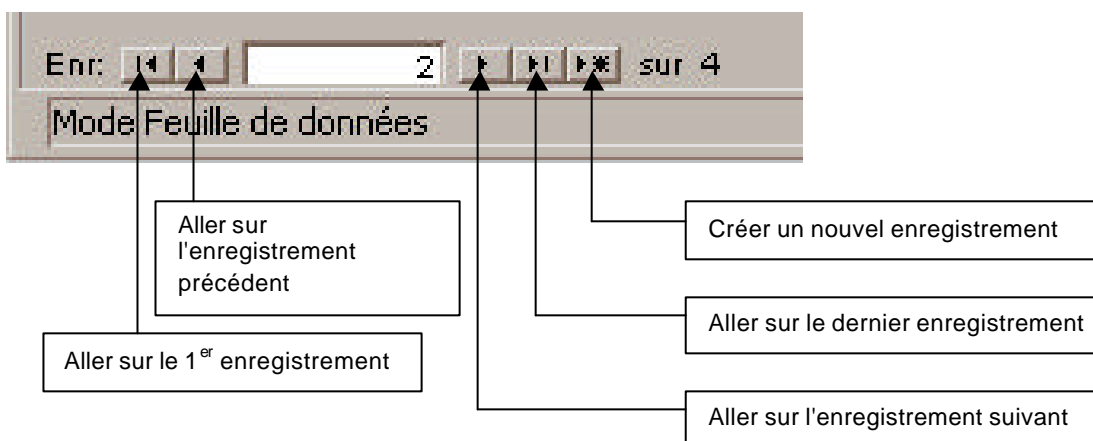
C'est beau non ?

3.6 La saisie des enregistrements

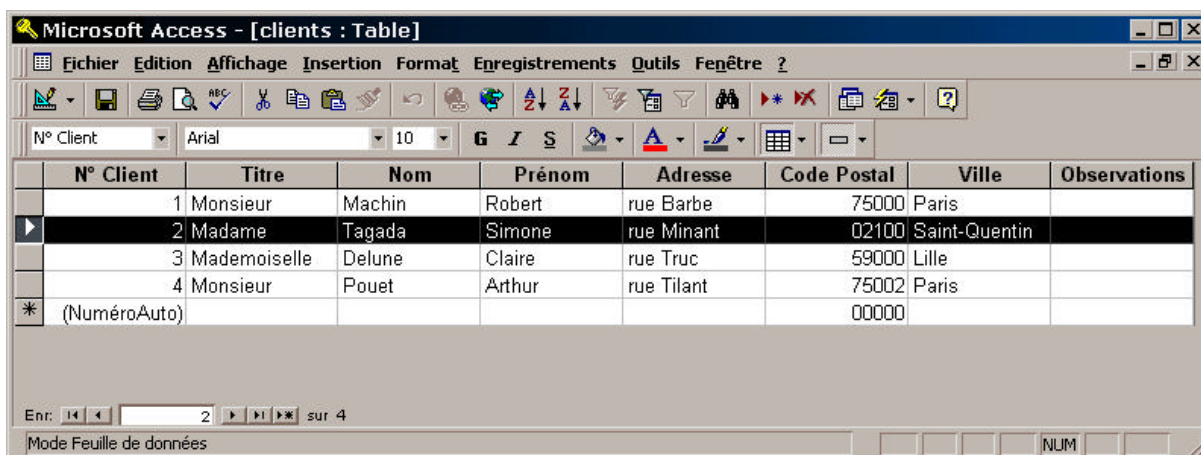
Jusqu'à présent, nous avons saisi rapidement des enregistrements dans les tables, attardons nous sur la saisie et les recherches / filtres .



Pour se déplacer parmi les enregistrements de la table, on utilise les icônes fléchés en bas à gauche de la fenêtre de saisie:



Pour supprimer un enregistrement, on clique à gauche de la ligne



et on appuie sur la touche "Suppr", Access vous demande alors une confirmation.

Deux petits trucs avant d'en finir avec la saisie de données dans les tables :

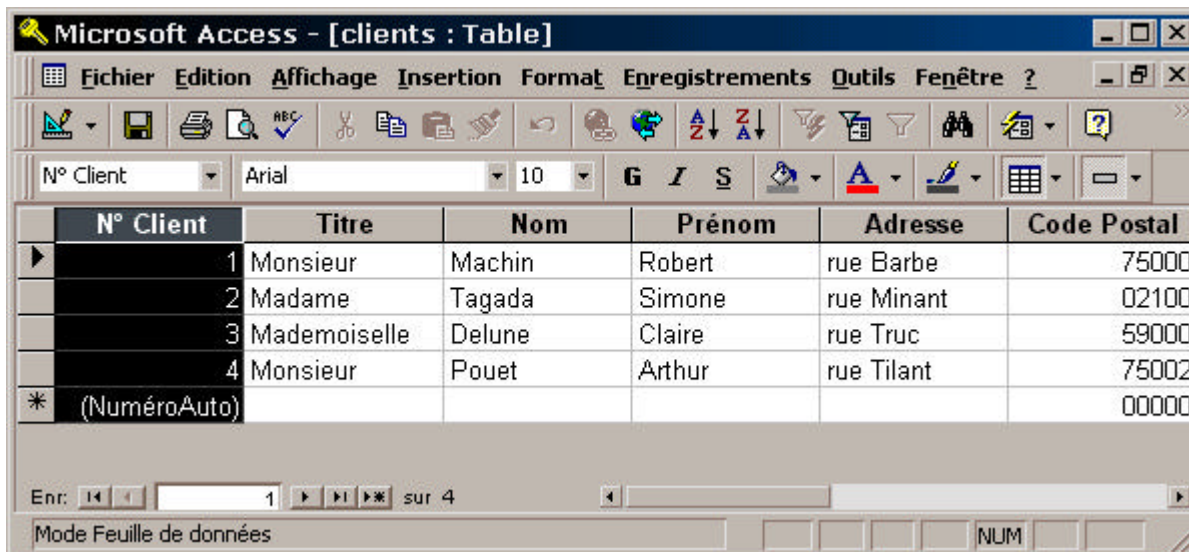
On peut cacher une colonne, pour ça, on va dans le menu "Format", et on clique sur l'option "Afficher les colonnes..."



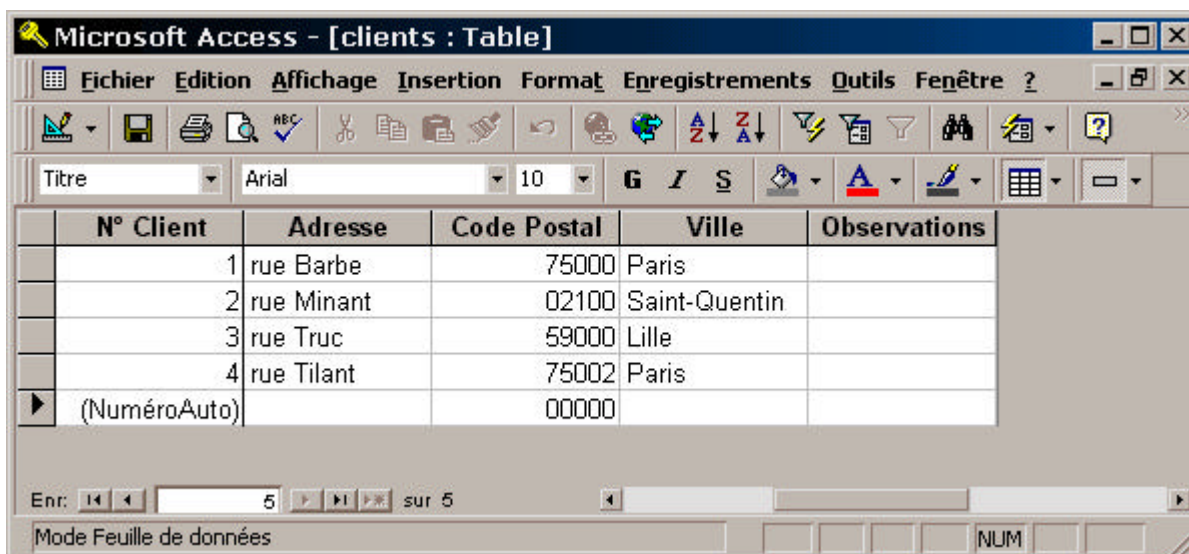
La liste des champs (colonnes) de la table s'affiche, vous sélectionnez celles que vous voulez voir apparaître, celles que vous n'avez pas sélectionné disparaîtront de l'affichage quand vous cliquerez sur "Fermer".

Attention, ca les fait juste disparaître de l'affichage pour la saisie, en aucun cas, le champ en question va être supprimé de la structure de la table.

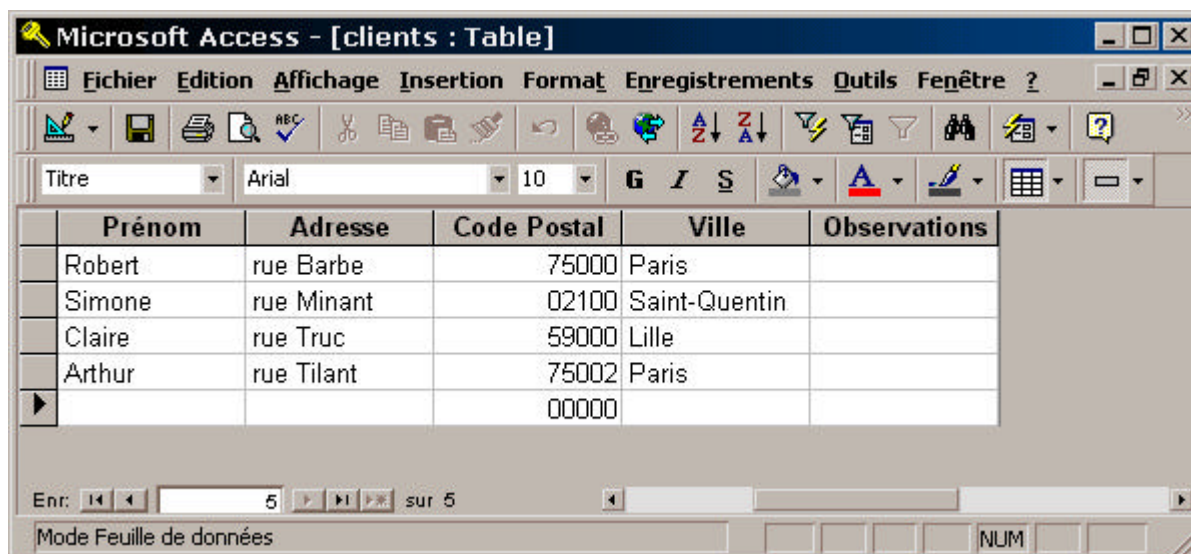
Autre truc intéressant, on peut "figer" une colonne, à quoi ça sert ? Si pendant la saisie nous voulons avoir en permanence le numéro client, nous allons sélectionner la colonne en question en cliquant sur son titre :



Une fois la colonne sélectionnée, on va dans le menu "Format" et on clique sur "Figer les colonnes". Que se passe-t-il maintenant ? Quand on va déplacer l'ascenseur en bas pour modifier un des champs de l'enregistrement, la colonne "N° Client" sera toujours visible :



Si la colonne n'était pas figée, on aurait eu :



	Prénom	Adresse	Code Postal	Ville	Observations
	Robert	rue Barbe	75000	Paris	
	Simone	rue Minant	02100	Saint-Quentin	
	Claire	rue Truc	59000	Lille	
	Arthur	rue Tilant	75002	Paris	
			00000		

Enr: 5 sur 5
Mode Feuille de données

La colonne "N° Client" a bougé quand on a déplacé l'ascenseur, lorsqu'elle était figée, elle était bloquée.
Pour autoriser le déplacement de toutes les colonnes et annuler l'option "**Figer les colonnes**", dans le menu "**Format**", cliquez sur l'option "**Libérer les colonnes**".

3.7 Tri parmi les enregistrements

Pour trier la table sur un champ, on se positionne sur le champ en question (peu importe l'enregistrement) et on utilise un des deux icônes de la barre d'outils :



Celui de gauche permet le tri alphabétique croissant et celui de droite le tri alphabétique décroissant.

Exemple :

N° Client	Titre	Nom	Prénom	Adresse	Code Postal
3	Mademoiselle	Delune	Claire	rue Truc	59000
1	Monsieur	Machin	Robert	rue Barbe	75000
4	Monsieur	Pouet	Arthur	rue Tilant	75002
2	Madame	Tagada	Simone	rue Minant	02100
(NuméroAuto)					00000

Ceci est l'état de notre table après l'avoir trié de façon croissante sur le champ "Nom"

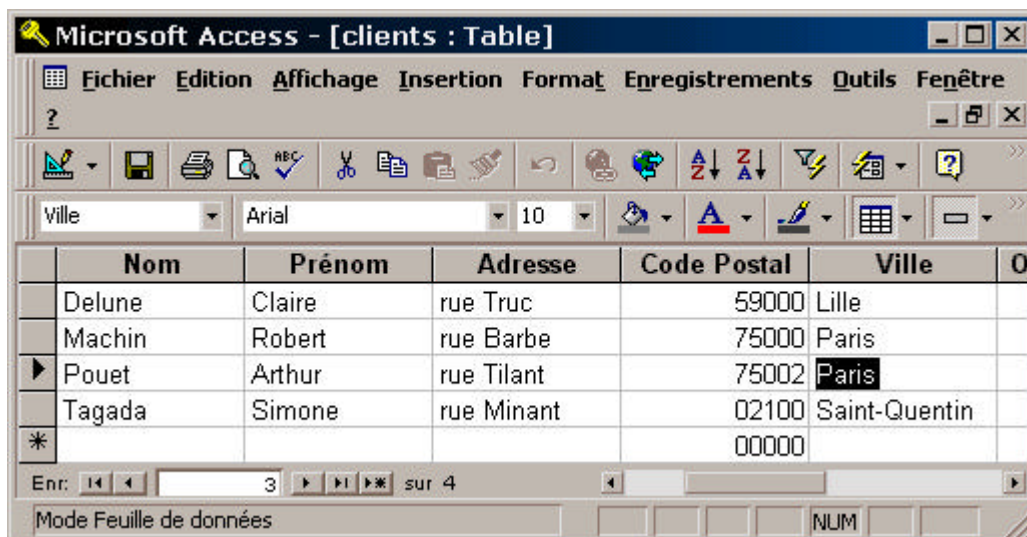
3.8 Filtrer des enregistrements

Les filtres permettent de limiter, de façon temporaire, les enregistrements affichés dans la table. On peut filtrer selon deux méthodes :

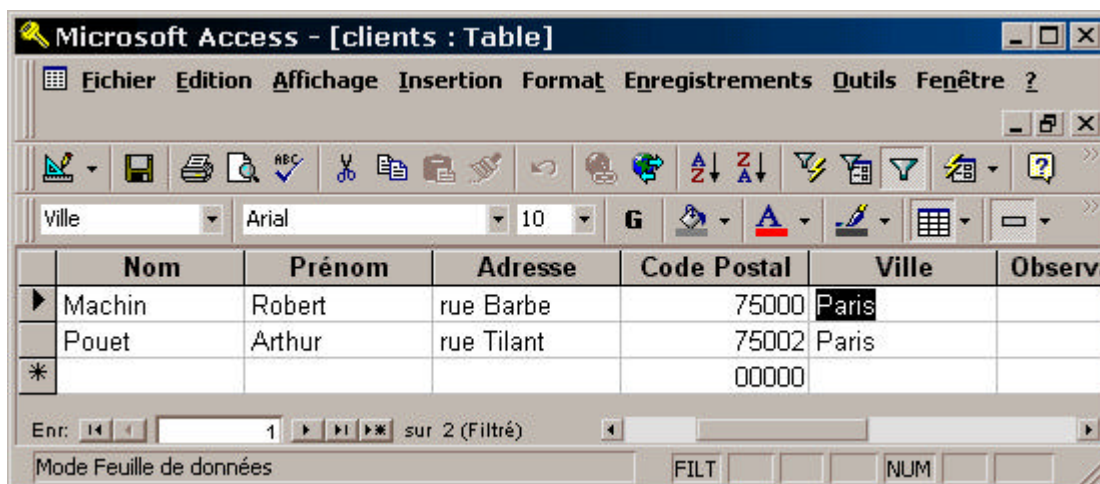
Le filtre sur un seul critère :

Par exemple, on ne veut afficher que les clients qui habitent Paris.

Pour cela, on se positionne sur l'enregistrement d'un client habitant Paris, et on sélectionne le mot "Paris" dans le champ "Ville":



Puis on clique sur l'icône du filtre dans la boîte à outils :



Ne s'affichent à présent que les client habitant à Paris. On peut aussi limiter le filtre à une partie d'un mot, par exemple, on aurait pu sélectionner seulement la lettre "P" du mot "Paris", ne se seraient alors affichés que les clients dont le nom de la ville commence par "P".

Pour faire réapparaître tous les enregistrements, on clique sur l'icône



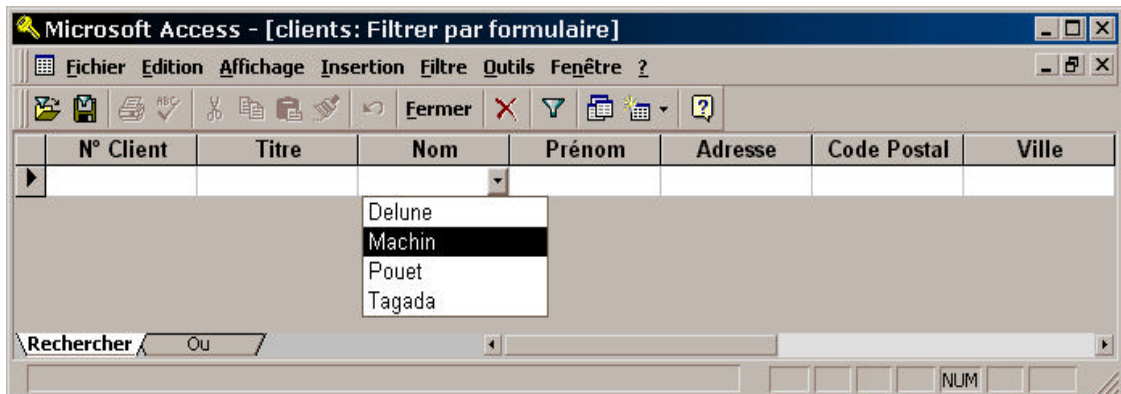
Filtre sur plusieurs critères

On clique sur l'icône :



Un enregistrement vide s'affiche.

Supposons que l'on veuille afficher les clients qui s'appellent "Machin" ou qui habitent à Saint-Quentin.

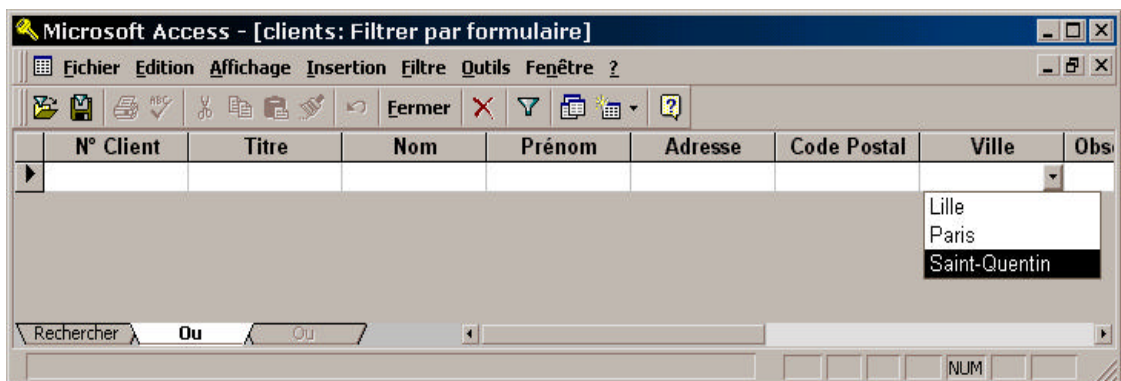


On va cliquer sur "Machin", si on voulait afficher les clients qui s'appellent "Machin" et dont le prénom est "Robert", on cliquerait sur "Robert" dans la colonne "Prénom".

Vous avez remarqué que, en bas, se trouvent deux onglets :



En cliquant sur l'onglet "Ou", vous obtenez un nouvel enregistrement vide dans lequel on va pouvoir saisir les critères de notre deuxième condition, le but étant d'avoir les clients qui s'appellent "Machin" OU qui habitent à Saint-Quentin, dans le deuxième onglet, on va sélectionner "Saint-Quentin" dans le champ ville :

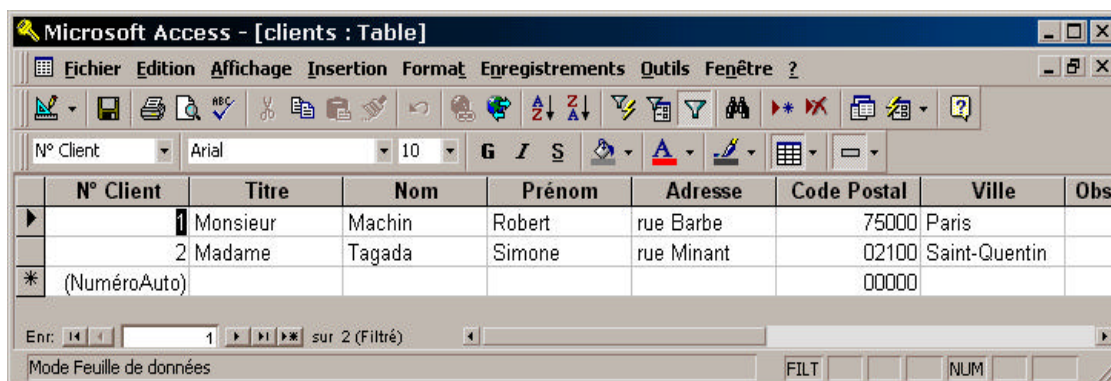


Au passage, notez qu'un troisième onglet est apparu au cas où on voudrait une troisième condition.

Une fois entrée la deuxième condition, on clique sur :



Et on obtient :

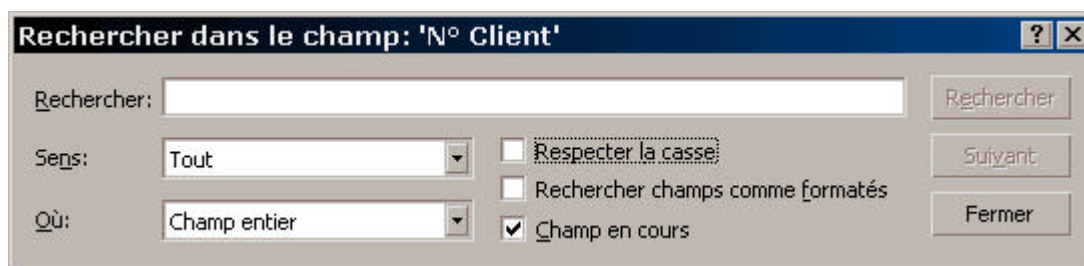


Soit la liste des clients dont le nom est soit "Machin" soit la ville est "Saint-Quentin".

Vous noterez au passage qu'on peut sauver son filtre multi critère sur disque en cliquant sur l'icône disquette dans la fenêtre de filtre.

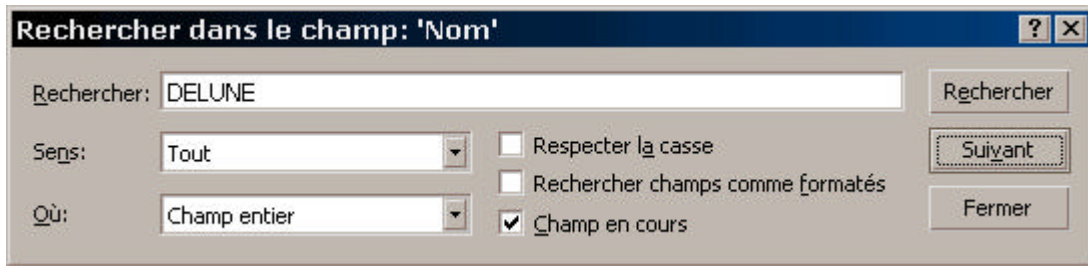
3.9 Rechercher des enregistrements

Pour faire une recherche, cliquez sur l'icône :

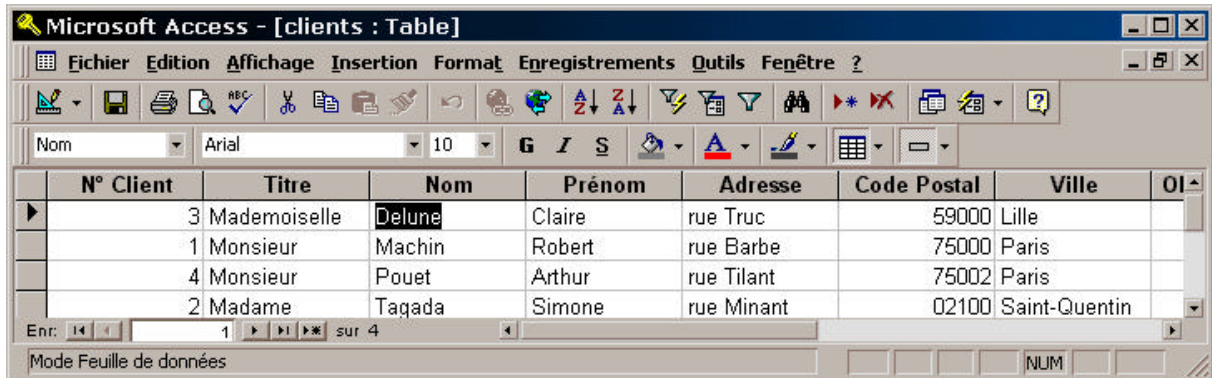


Rechercher	on indique la valeur à chercher
Sens	Indique dans quel sens va se faire la recherche : ?? <u>Tout</u> : dans toute la table ?? <u>Bas</u> : à partir de l'enregistrement courant jusque la fin de la table ?? <u>Haut</u> : à partir du début de la table jusque l'enregistrement courant.
Où	Sur quoi va se faire la recherche ?? <u>Champ entier</u> : La valeur cherchée doit être le contenu exact du champ ?? <u>Début de champ</u> : La valeur cherchée est le début du champ ?? <u>N'importe où</u> : La valeur cherchée peut se trouver n'importe où dans le champ
Respecter la casse	Doit-on différencier les majuscules des minuscules ?
Rechercher les champs comme Formatés :	Activez cette option si vous souhaitez qu'Access retrouve les données telles qu'elles sont affichées dans la table (en utilisant la propriété "Format" du champ) ou telles qu'elles sont stockées (c'est-à-dire telles qu'elles ont été saisies). Par exemple : pour le code postal, on a défini un format d'affichage : 00000 pour que si le code postal commence par 0 (comme 02100), il soit affiché correctement et pas sous la forme 2100. Si on active l'option "Rechercher champs comme formatés", on pourra rechercher le code postal 02100, sinon, il faudra chercher le code postal 2100.
Champ en cours	Indique si la recherche doit s'effectuer uniquement dans le champ en cours ou dans tous les champs de la table

Par exemple, pour rechercher le client "Delune" :

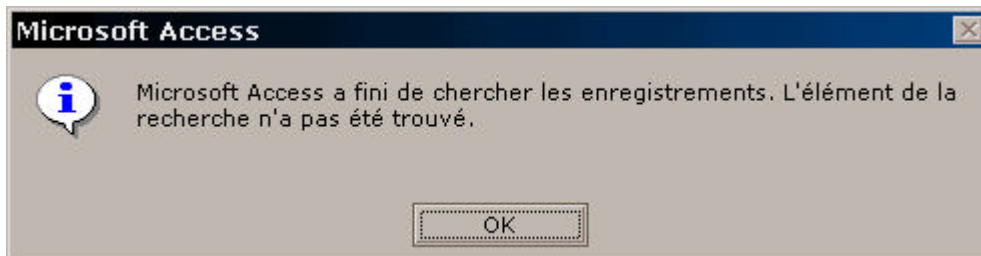


On clique sur "Rechercher"



Access se positionne sur le premier enregistrement dont le contenu du champ "Nom" est "Delune"

Pour trouver le suivant, on clique sur "Suivant"



En l'occurrence ici, il n'y en a pas d'autres, Access renvoie ce message.

On peut faire des recherches approximatives, pour cela on utilise des caractères génériques :

Caractère	Signification
*	Remplace une chaîne de valeur quelconque
?	Remplace un seul caractère
#	Remplace un chiffre unique
[]	Recherche des caractères parmi plusieurs
!	Exclu certains caractères de la recherche

Exemple :

La chaîne	Permet de retrouver	Mais ne retrouve pas
*rue	112 rue Boulevard Machin	
??rue	5 rue 8 rue	112 rue
##rue	10 rue	La rue
DUPON[TD]	DUPONT DUPOND	DUPONS
DUPON![TD]	DUPONS	DUPOND DUPONT

Prochain chapitre : les formulaires ...

1. Introduction.....	2
2. Création d'une requête.....	2
3. Définition des critères de sélection.....	5
3.1 Opérateurs	5
3.2 Les Fonctions.....	6
3.3 Plusieurs critères portant sur des champs différents.....	7
3.4 Requête paramétrée	8
4. Les requêtes multitables	9
5. Les fonctions de regroupement.....	10
5.1 Les opérations.....	11
5.2 Quelques exemples.....	12
6. Les requêtes d'analyse croisée	13
7. Les requêtes ACTION.....	14
7.1 Les requêtes Création.....	14
7.2 Les requêtes Ajout	15
7.3 Les requêtes Mise à Jour.....	16
7.4 Les requêtes Suppression.....	17

LES REQUETES

1. Introduction

Les requêtes vont servir à afficher uniquement certaines données contenues dans les tables selon certains critères. Elles peuvent aussi faire des calculs sur vos données, ainsi que créer des sources de données pour les formulaires, les états ou même d'autres requêtes (on peut faire une requête sur le résultat d'une autre requête). Elles servent encore à modifier des tables existantes ou à en créer des nouvelles.

Il existe différents types de requêtes que nous allons détailler après :

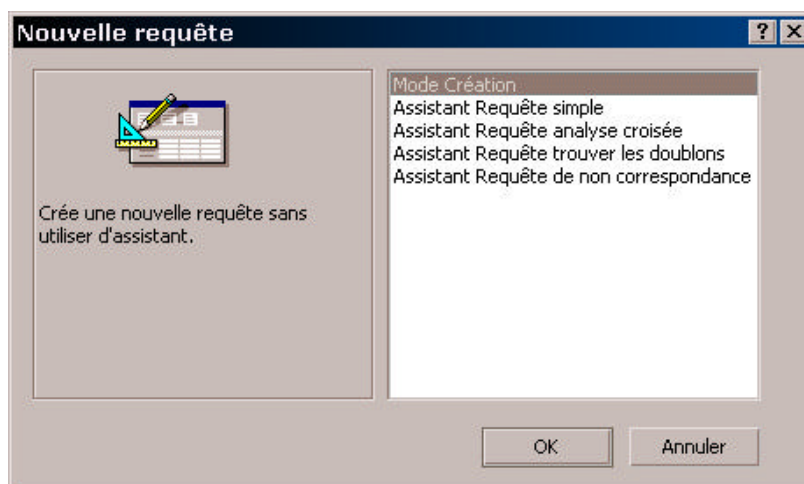
- ?? **La requête sélection** : C'est celle qu'on utilisera le plus souvent. Elle permet de sélectionner des enregistrements, de faire des calculs et des regroupements. Elles ressemblent beaucoup aux filtres, mais permettent, en plus, de travailler sur plusieurs tables simultanément.
- ?? **La requête d'Analyse croisée** : Cette requête présente ses résultats sous forme de tableau (de type Excel). On l'utilisera pour comparer des valeurs, dégager des tendances.
- ?? **La requête de Création de table** : Cette requête crée une table à partir des données qu'elle a extraites dans une ou plusieurs autres tables.
- ?? **La requête Mise à Jour** : Cette requête modifie le contenu d'un ou plusieurs champs d'une ou plusieurs tables. C'est le moyen le plus efficace pour mettre à jour un grand nombre d'enregistrements en une seule opération.
- ?? **La requête Ajout** : Cette requête ajoute les données qu'elle a extraites à la fin d'une table déjà existante.
- ?? **La requête Suppression** : Cette requête supprime un ou plusieurs enregistrements dans une ou plusieurs tables.

Les trois derniers types de requêtes ne seront pas étudiés :

- ?? La requête SQL direct
- ?? La requête de définition de données
- ?? La requête UNION

2. Création d'une requête

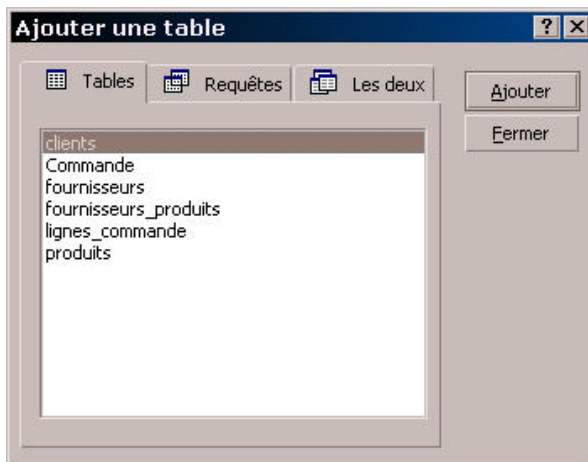
Dans la fenêtre principale d'Access, cliquez sur l'onglet "Requêtes", puis sur le bouton "Nouveau".



Access nous propose 5 façons de créer une requête :

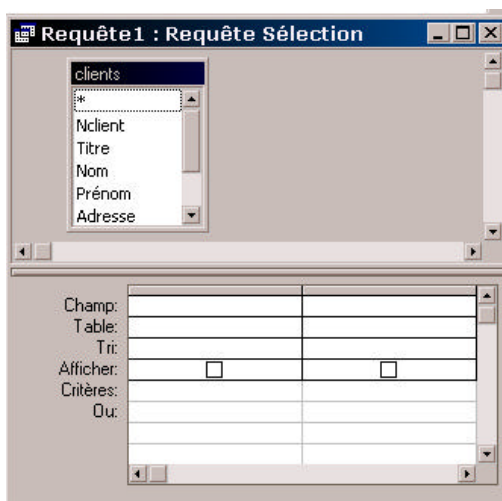
- ?? **Mode création** : nous allons créer la requête de A à Z par nous-mêmes
- ?? **Assistant de Requête simple** : Crée une requête simple sur une ou plusieurs tables, on pourra, dans cette requête simple, faire des calculs, des regroupements.
- ?? **Assistant de Requête d'analyse croisée** : Aide à la création d'une requête d'analyse croisée
- ?? **Assistant de Requête trouver les doublons** : Localise les enregistrements doublons dans une table ou une requête.
- ?? **Assistant de Requête de non-correspondance** : Localise les enregistrements d'une table auxquels ne correspond aucun autre enregistrement d'une autre table. On peut, par exemple, utiliser un tel type de requête pour localiser les clients qui n'ont pas passé de commande.

Comme nous ne reculons pas devant l'effort, nous allons créer les requêtes par nous même, nous cliquons donc sur **Mode Création**, puis, sur OK :

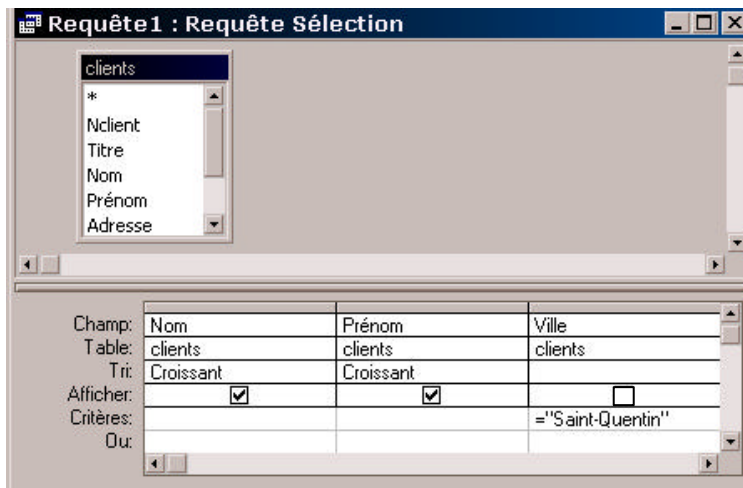


Access nous affiche la liste des tables de la base. Nous choisissons ici les tables sur lesquelles vont porter la requête. Nous allons prendre pour exemple la requête "liste des clients habitant la ville de Paris". La requête va donc porter sur la table "**Clients**", on la sélectionne, et on clique sur "**Ajouter**", comme il n'y a aucune autre table à ajouter, on clique sur "**Fermer**".

Notez qu'on peut aussi faire porter la requête sur le résultat d'une autre requête, dans ce cas, on choisira la requête dans l'onglet "Requête".



Dans la partie supérieure de la fenêtre se trouvent la ou les tables avec la listes de leur champs, c'est sur ces tables que vont porter les requêtes. Dans la partie inférieure, se trouve la description de la requête.



Voici notre requête :

Nous avons un tableau composé de colonnes et de lignes, dans chaque colonne, on indique les champs qui vont apparaître dans le résultat de la requête ou qui vont servir dans la requête, ici on veut la liste des clients habitant Saint-Quentin, on veut donc voir apparaître le champ **Nom** et le champ **Prénom**, le critère de la requête va se faire sur le champ **ville**, on ajoute donc aussi ce champ.

Pour ajouter un champ, on peut, soit le sélectionner dans la table et l'amener avec la souris sur une colonne, soit on clique sur la ligne "**Champ**", la liste de tous les champs s'affiche alors et on en sélectionne un dans la liste. Un des champs proposés s'appelle '*', ce champ signifie "tous les champs de la table", si on choisit ce champ, tous les champs de la table apparaîtront dans le résultat de la requête, en plus des autres champs que vous aurez choisis.

La ligne "**Table**" sert à sélectionner la table à laquelle appartient le champ sélectionné, dans notre cas, il n'y a qu'une table, le choix est vite fait.

La colonne "**Tri**" indique de quelle façon vont être triés les champs dans le résultat de la requête : il y a trois sortes de tri : Croissant (de A à Z), Décroissant (de Z à A) et non trié. On a choisi ici de trier le résultat de la requête par nom et par prénom de façon croissante. Le tri se fait toujours de gauche à droite : le résultat de la requête sera d'abord trié par nom, puis par prénom.

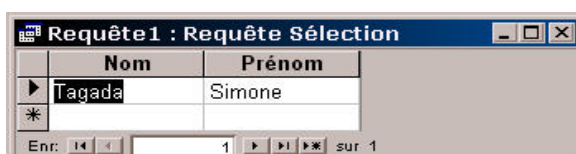
La colonne "**Afficher**" indique si le champ doit être affiché ou non, ici on veut afficher le nom et le prénom de chaque client habitant à Saint-Quentin, mais on a pas besoin d'afficher, pour chaque client, la ville dans laquelle il habite, puisqu'il s'agira toujours de Saint-Quentin, on a donc désélectionné l'affichage du champ "**Ville**".

Enfin la colonne "**Critères**" va indiquer le critère de la requête, on veut la liste des clients habitant Saint-Quentin, le critère est donc : le champ ville doit être égal à Saint-Quentin, d'où le critère = "Saint-Quentin".

Pour exécuter la requête, on clique sur l'icône :



Access affiche le résultat :



Le résultat de la requête s'affiche sous la forme d'une table que l'on peut modifier comme si il s'agissait d'une table normale, on peut ajouter des enregistrements, les modifier, faire des filtres, des tris, etc...

Attention : Les résultats des requêtes, même si ils sont présentés sous forme de tables ne sont pas de véritables tables qu'aurait généré la requête. Ils ne sont qu'une "vue" faite à partir des tables qui ont servi à faire la requête. Autrement dit, si vous modifiez quelque chose dans le résultat de la requête, la modification se repercutera dans la table qui a servi à faire la requête, si on change ici le nom du client, la modification sera reportée dans la table Clients, si on ajoute un client au résultat de la requête, un nouvel enregistrement va être créé dans la table Clients, et en plus, cet enregistrement sera incomplète car seuls deux champs auront pu être saisis dans le résultat de la requête (les champs nom & prénom).

3. Définition des critères de sélection

3.1 Opérateurs

On peut utiliser dans les requêtes les opérateurs suivants :

Opérateur	Signification
=	Egal
<>	Différent
<	Inférieur
>	Supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal

Access met à notre disposition d'autres opérateurs :

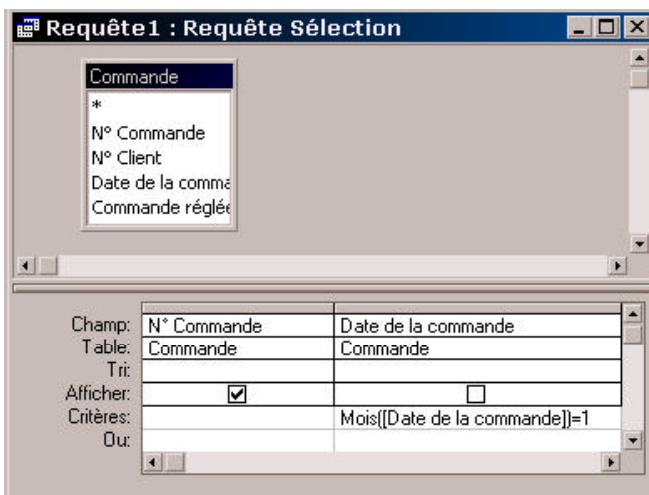
Opérateur	Signification	Exemple
Entre	Sélectionne les enregistrements pour lesquels la valeur d'un champ est comprise dans un intervalle de valeurs.	Entre "A" et "C" Entre 10 et 20 Entre #01/01/99# et #31/12/99#
Dans	Sélectionne les enregistrements pour lesquels la valeur d'un champ est comprise dans une liste.	Dans ("Paris";"Saint-Quentin")
Est	Sélectionne les enregistrements pour lesquels un champ est vide ou non	Est NULL Est pas NULL
Comme	Sélectionne les enregistrements contenant une donnée approximative.	Comme "rue*"
Pas	Sélectionne les enregistrements ne correspondant pas au critère	Pas Entre "A" et "C"

3.2 Les Fonctions

On peut intégrer des fonctions dans les critères de sélection. Access met à notre disposition un très grand nombre de fonctions (pour en avoir la liste complète, consultez l'aide intégrée à Access). Ce sont les mêmes fonctions que celles qui sont utilisés dans les contrôles des formulaires (et d'ailleurs dans tous les logiciels de la gamme Microsoft Office).

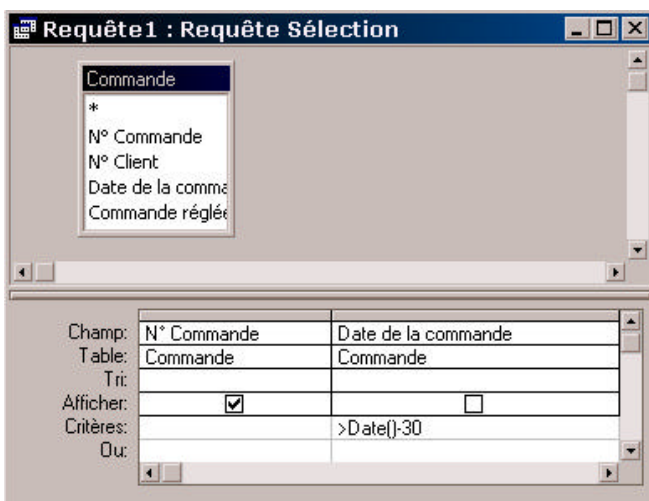
Par exemple, parmi ces fonctions, on trouve la fonction Mois (), à qui on passe une date en paramètre, cette fonction renvoie le mois de cette date, par exemple Mois (#10/2/98#) renvoie 2. Une autre fonction, Date () renvoie la date du jour. On peut utiliser ces deux fonctions dans des critères de sélection :

Par exemple, pour obtenir la liste des commandes du mois d'octobre :



Notez que pour faire référence au champ "Date de la commande" dans la fonction Mois (), on a écrit le champ entre crochets []. On utilise les crochets lorsqu'on travaille avec un champ dont le nom comporte des espaces. Par extension et par mesure de précaution, on les utilisera avec tout type de champ.

Ou pour obtenir la liste des commandes passées il y a moins d'un mois :



Il existe un très grand nombre de fonctions, pour en avoir la liste et la syntaxe, consultez l'aide intégrée à Access.

3.3 Plusieurs critères portant sur des champs différents

On peut avoir plusieurs critères de sélection, ces critères étant séparés entre eux par des OU ou des ET, par exemple : "liste des clients habitant à Saint-Quentin OU à Paris", "Liste des clients s'appelant Dupont ET vivant à Paris".

1- Liste des clients s'appelant Dupont et vivant à Saint-Quentin

Champ:	Nom	Prénom	Ville
Table:	clients	clients	clients
Tri:			
Afficher:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Critères:	= "Dupont"		= "Paris"
Ou:			

La requête ressemble à la précédente, sauf que là, il y a deux critères : le premier Nom = "Dupont" ET le deuxième Ville = "Paris". Vous noterez au passage qu'il n'est pas nécessaire d'afficher le contenu du champ "ville" puisqu'on sait que ce sera toujours Paris. Vous me direz que c'est la même chose avec "Nom", certes, mais c'est mieux d'avoir une liste de nom + prénom qu'une seule liste de prénom.

2- Liste des clients habitant Saint-Quentin OU Paris

Champ:	Nom	Prénom	Ville	Ville
Table:	clients	clients	clients	clients
Tri:				
Afficher:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Critères:			= "Paris"	
Ou:				= "Saint-Quentin"

Quelle est la différence ?

Les deux critères ne sont plus sur la même ligne. Ils sont sur deux lignes séparées. Le premier est sur la ligne "Critères", le second sur la ligne "Ou". Si on avait voulu ajouter un troisième critère (Liste des clients habitant Saint-Quentin OU Paris OU Lille), on aurait ajouté le critère = "Lille" sur une troisième ligne et ainsi de suite.

En règle générale :

- ?? Si deux critères sont séparés par des ET, on les place sur la même ligne.
- ?? Si deux critères sont séparés par des OU, on les place sur des lignes différentes.

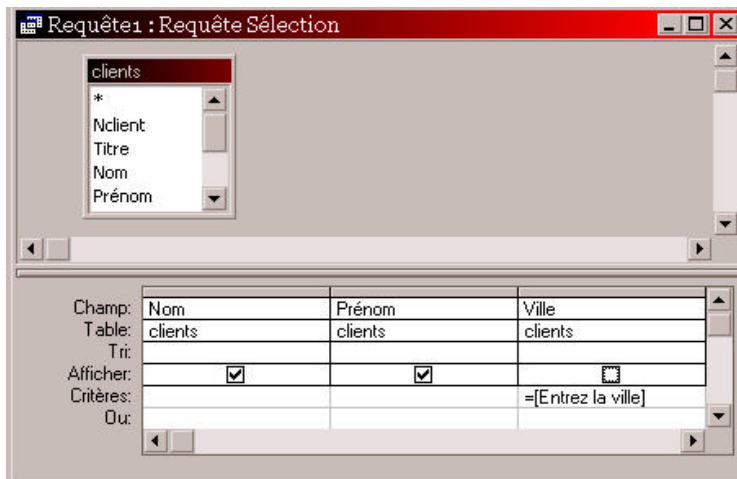
3.4 Requête paramétrée

Pour l'instant, les critères de chaque requête que nous avons fait étaient indiqués explicitement dans la requête. Supposons maintenant que nous voulions faire une requête du type :

« Liste de tous les clients qui habitent dans une ville » mais qu'on veuille entrer nous même la ville.

On ne connaît pas par avance la ville en question, et on ne va pas préparer autant de requêtes qu'il peut y avoir de villes (on n'est pas sorti de l'auberge), la solution la plus simple est alors de demander à l'utilisateur (l'utilisateur, c'est celui qui va utiliser votre requête plus tard) d'entrer la ville, et de faire la requête en fonction de ce qu'il a entré.

Pour faire ça, on procède ainsi :



Au lieu d'indiquer une ville, on a mis =[Entrez la ville]. Que va-t-il se passer lorsqu'on va exécuter la requête?

Habituellement, pour Access, tout ce qui est indiqué entre crochets est le nom d'un champ, si on avait mis par exemple = [Nom], il aurait cherché les clients qui habitent une ville qui s'appelle comme leur nom. Or ici, [Entrez la ville] n'est pas un nom de champ, Access ne sait donc pas ce que c'est, et il demande à l'utilisateur d'entrer la valeur de ce champ inconnu :

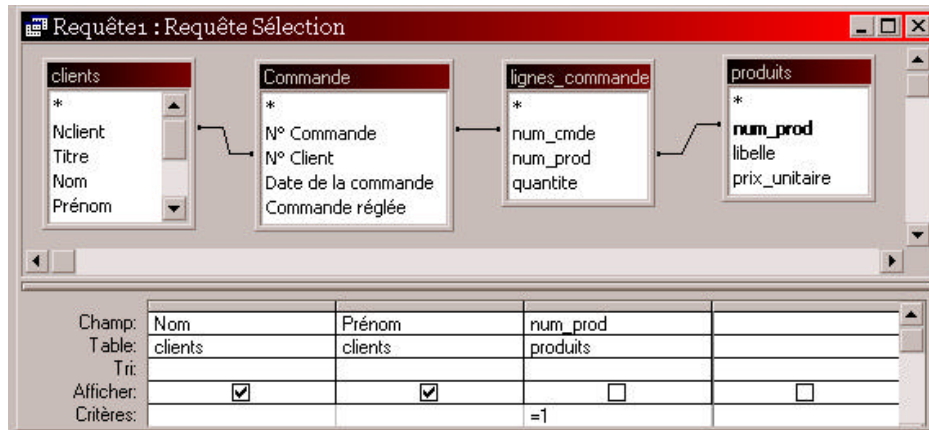


L'utilisateur va entrer ici la ville (notez que le message affiché est ce qui était indiqué entre crochets dans la requête). Maintenant, à chaque fois que Access va rencontrer dans la requête [Entrez la ville], il va le remplacer par ce qu'à saisi l'utilisateur. Si l'utilisateur a saisi Paris, Access va remplacer dans son critère =[Entrez la ville] par = "Paris".

4. Les requêtes multitable

Pour l'instant, nos requêtes ne portaient que sur une seule table, il est possible de faire des requêtes qui portent sur plusieurs tables simultanément. Dans ce cas, les requêtes peuvent être très complexes.

Par exemple, si nous voulons la liste des clients ayant commandé le produit n° 1 :



Avec le bouton : , on affiche la liste des tables de la base.

On va choisir ici toutes les tables entrant en jeu pour pouvoir faire la requête. Ici on veut la liste des clients, il faut donc la table **clients**. "ayant commandé", il faut donc la table **commandes**, "le produit n°1", il faut donc la table **produits**. Il faut aussi faire intervenir la table "**Lignes-commandes**".

Pourquoi ? parce que c'est elle qui fait la liaison entre la table commande et la table produits. De façon générale, lorsqu'on fait une requête portant sur plusieurs tables, il faut respecter deux règles :

- ?? Toutes les tables intervenant dans la requêtes doivent être reliées entre elles, il ne doit pas y avoir de tables isolées, sinon, Access va essayer de trouver lui-même les relations entre ces tables isolées et parfois le résultat peut être folklorique.
- ?? Il ne doit pas y avoir de tables n'ayant rien à faire dans la requête, sinon, Access va se baser sur les relations entre ces tables n'ayant rien à voir avec la choucroute et celles ayant à voir et va donner des résultats erronés.

En résumé : toutes les tables nécessaires, mais pas plus.

Vous pouvez noter que, une fois les bonnes tables installées avec les bonnes relations entre elles, la requête est fort simple, il suffit d'indiquer produits=1, et Access, grâce aux relations, va retrouver la liste des clients ayant commandé ce produit.

5. Les fonctions de regroupement

Jusqu'à présent, nos requêtes nous permettaient de répondre à des questions du type : "Liste des clients habitant Paris", "Liste des produits commandés par le client 1". Grâce aux fonctions de regroupement, nous allons pouvoir répondre à des questions du type : "Combien de clients habitent Paris ?" ou "Pour combien à commandé chaque client ?".

Pour cela, cliquez sur le bouton :



Une nouvelle ligne '**Opération**' apparaît dans la requête, c'est grâce à elle que nous allons faire nos opérations.

1- Combien de clients habitent Paris

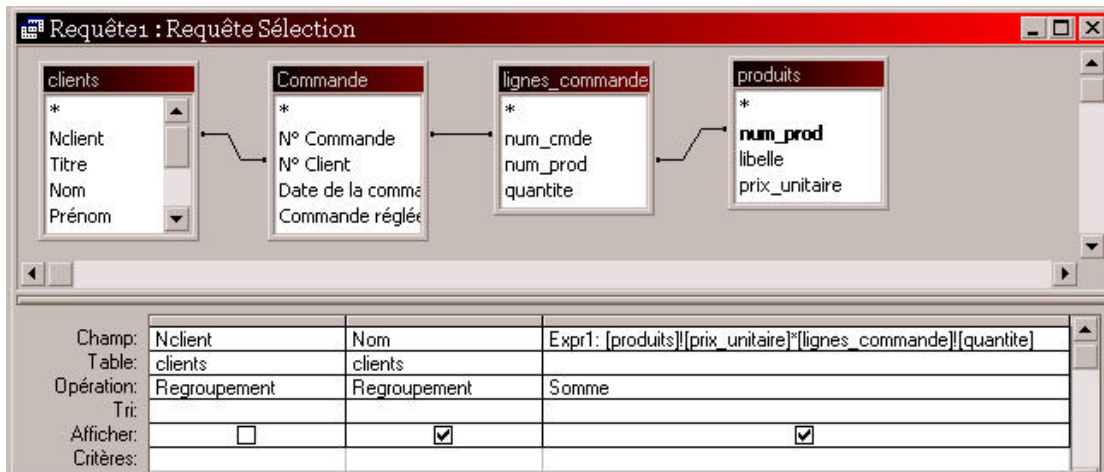


Qu'y-a-t-il de différent ?

Nous avons deux champs : nous allons compter le nombre de clients, chaque client a un numéro, on insère donc dans la requête le numéro de client, c'est le nombre de ces numéros que nous allons compter, il faut qu'on compte les clients habitant Paris, d'où le deuxième champ Ville.

La différence est sur la ligne Opération :

Elle indique comment va se faire l'opération, si nous voulons compter, on va choisir l'opération "**compte**". Dans tous les champs où il y a un critère de sélection (ici Ville = "Paris"), on choisira l'opération où. Le résultat de la requête sera une table avec un unique champ, non modifiable, qui indiquera le nombre de clients habitant Paris.

2- Montant commandé par chaque client

Ca se corse !

Ici, on veut, par client, la somme de ses commandes. A partir du moment où on veut un résultat par catégorie (une somme par client, un nombre de clients par ville, etc...), il y a regroupement, ici on veut la somme des commandes regroupées par clients, c'est pour ça qu'on a choisi comme opération pour numéro de client "Regroupement".

Pourquoi at-on ajouté le champ "nom" ? Simplement pour ne pas avoir une liste de numéros avec un montant correspondant. Pourquoi dans ce cas, n'a-t-on pas regroupé les clients par leur nom au lieu de le faire par numéro et nom ? parce que plusieurs clients peuvent avoir le même nom et qu'on ne veut pas cumuler le montant des commandes par clients homonymes, on regroupe donc les clients par numéro et nom.

Enfin, le troisième champ : on veut la somme du montant des commandes : qu'est-ce qu'une commande ? c'est une liste de prix unitaires * une quantité. A la place d'un champ, on indique donc qu'on veut les prix unitaires * les quantités. La syntaxe est [nom de la table].[champ de la table], une commande, c'est donc une liste de [produits].[prix unitaire] * [lignes_commandes].[quantité]. Et on veut la somme de toutes ces commandes, on choisit donc l'opération "Somme".

5.1 Les opérations

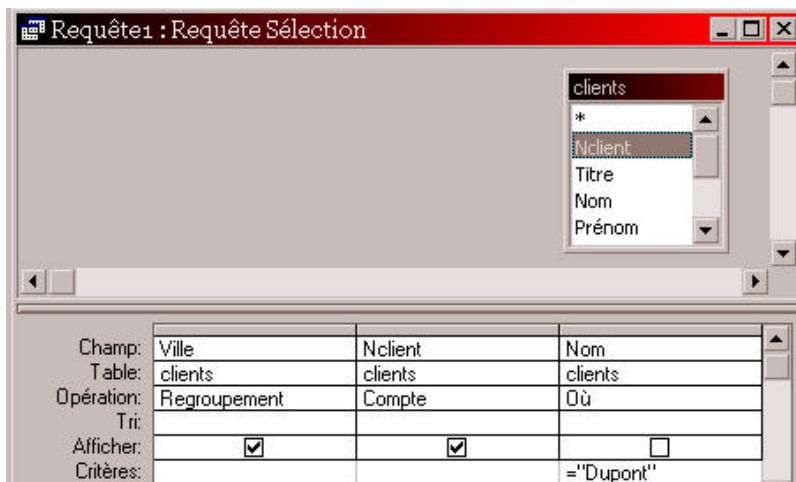
En plus de la somme et de compte, il existe d'autres opérations possibles :

Opération	Signification
Compte	Compte le nombre de valeurs
Dernier	Valeur du dernier enregistrement
Ecarttype	Ecart type
Max	Valeur la plus élevée
Min	Valeur la plus faible
Moyenne	Moyenne
Premier	Valeur du premier enregistrement
Somme	Total
Var	Variance

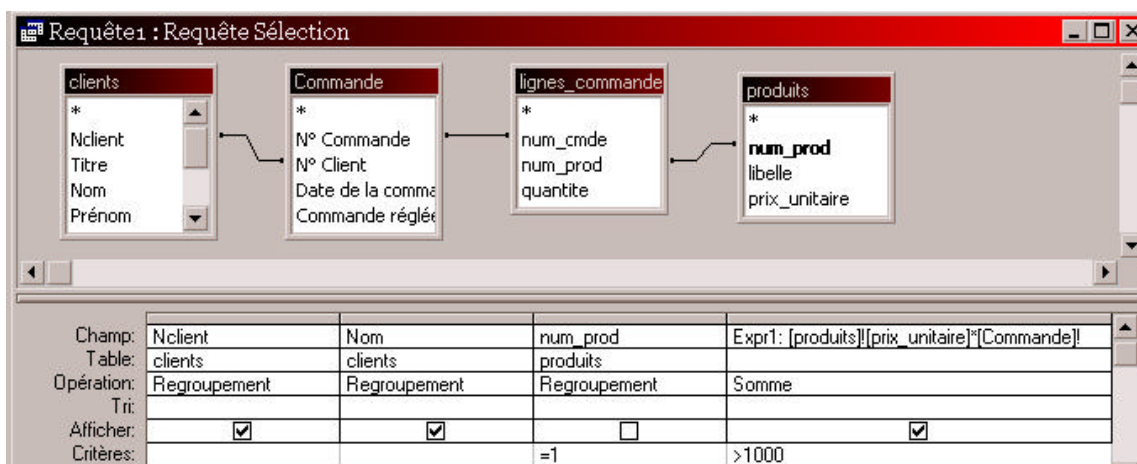
5.2 Quelques exemples

Voici quelques exemples de requêtes :

Nombre de clients nommés "Dupont" par ville



Liste des clients ayant commandé plus de 1000 F du produit 1



6. Les requêtes d'analyse croisée

Les requêtes d'analyse croisée permettent de répondre à des questions du type "qui a commandé combien de quoi ?". Elles retournent le résultat sous forme d'un tableau comportant des champs en abscisse et en ordonnée, avec, dans chaque case la réponse à notre question.

Exemple : Qui a commandé combien de quoi ?

	Qui	Qui
Quoi	Combien	Combien
Quoi	Combien	combien

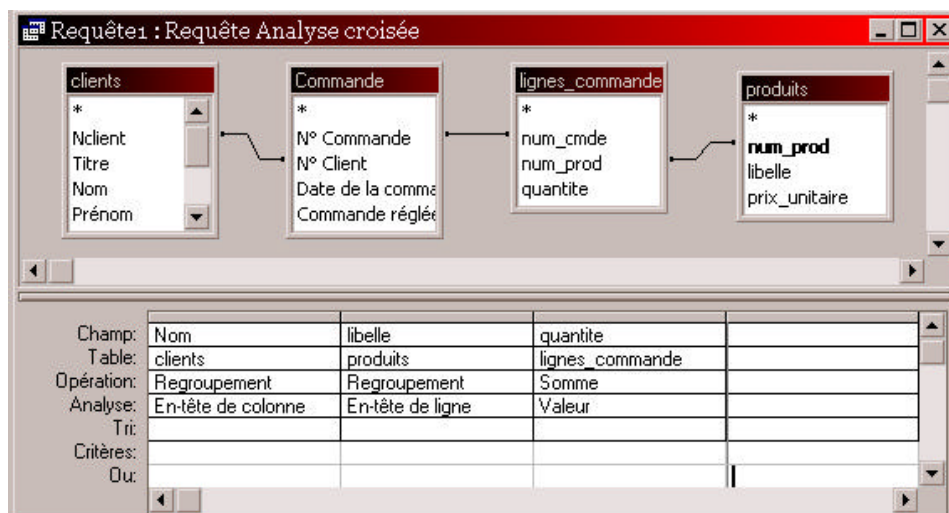
L'en-tête "Qui" va contenir le nom des clients

L'en-tête "Quoi" va contenir la liste des produits

Et combien va donner, pour chaque client, le nombre de produits qu'il a commandé.

Comment procède-t-on ?

1. D'abord on crée une requête standard : pour cette requête nous avons besoin du nom dans la table clients, du libellé du produit dans la table produit et de la quantité commandé dans la table lignes-commandes.
2. On transforme la requête en requête d'analyse croisée en allant dans le menu "Requête" et en choisissant "Analyse croisée".
3. Une nouvelle ligne apparaît dans la requête : la ligne "Analyse"
4. Dans cette ligne, on va indiquer si le champ qu'on a choisi va être l'en-tête des colonnes, l'en-tête des lignes ou la valeur contenue dans les cases du tableau.
5. L'Opération pour les en-têtes est toujours "Regroupement"
6. L'Opération pour les valeurs des cases dépend de ce qu'on cherche, ici on cherche le nombre de produits acheté, l'opération est donc "Somme"



7. Les requêtes ACTION

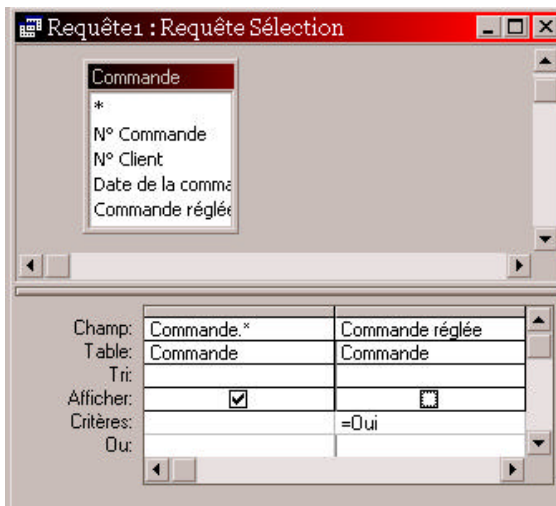
Les requêtes faites jusqu'à présent se contentent de retourner le résultat d'une sélection sous forme de table, cette table, étant, selon le type de requête, modifiable. Il existe cependant d'autres types de requêtes, les requêtes action, ces requêtes vont permettre de :

- ?? Créer une table à partir du résultat d'une requête
- ?? Ajout des enregistrements à une table à partir des résultats de la requête
- ?? Mettre à Jour une table en fonction de certains critères
- ?? Supprimer des enregistrements répondant aux critères de la requête

7.1 Les requêtes Création

Une requête création crée une table à partir des résultats qu'elle produit à partir d'une table existante. Supposons que nous voulions créer une table "Commandes réglées" qui contiendrait la liste des commandes déjà réglées.

1. Créer la requête normalement : nous voulons la liste des commandes réglées :



2. On la transforme en requête Création (Menu Requête / Requête Création de table)
3. Access nous demande le nom de la table à créer :



Attention, si vous sélectionnez une table existante, la table va être écrasée par cette opération

- Exécutez la requête avec l'icône point d'exclamation : la table va être créée avec le résultat de la requête. Si vous voulez vérifier avant de créer la table quel sera le résultat de la requête, cliquez sur l'icône:



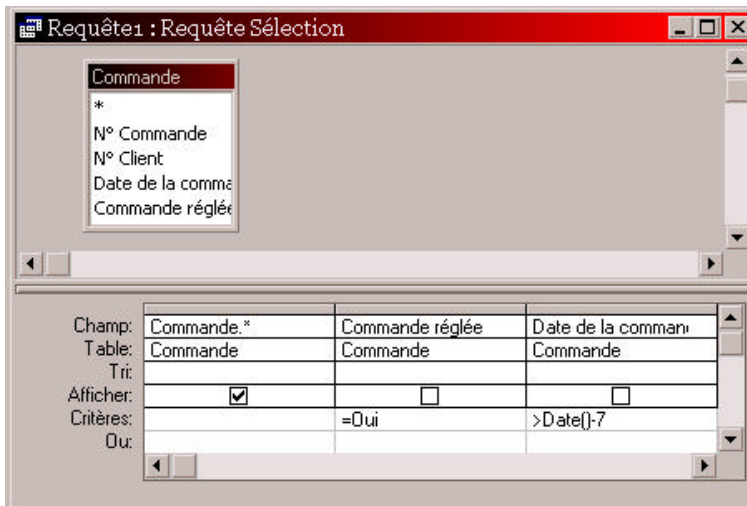
Une fois que le résultat sera conforme à vos attentes, vous pourrez cliquer sur l'icône point d'exclamation.

7.2 Les requêtes Ajout

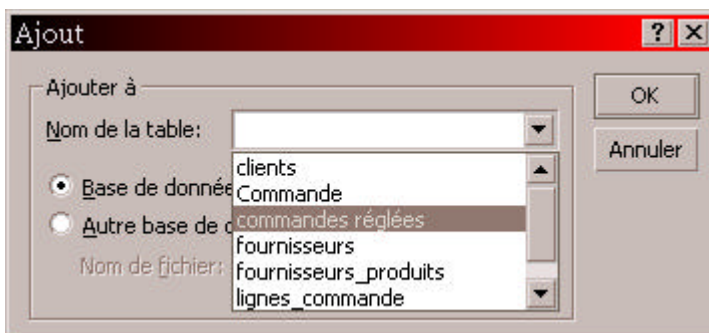
Une requête Ajout copie tout ou partie des enregistrements d'une table (la table source) à la fin d'une autre table (la table cible).

Toujours dans notre table "Commandes réglées", nous voulons ajouter les commandes qui ont été réglées depuis la semaine dernière :

- Création de la requête normale :



- On la transforme en requête Ajout (Menu Requête / Requête Ajout)
- Access nous demande le nom de la table à laquelle il faut ajouter le résultat de la requête :



- Comme précédemment, avec les icônes Affichage et point d'exclamation, vérifiez et validez votre requête.

Attention :

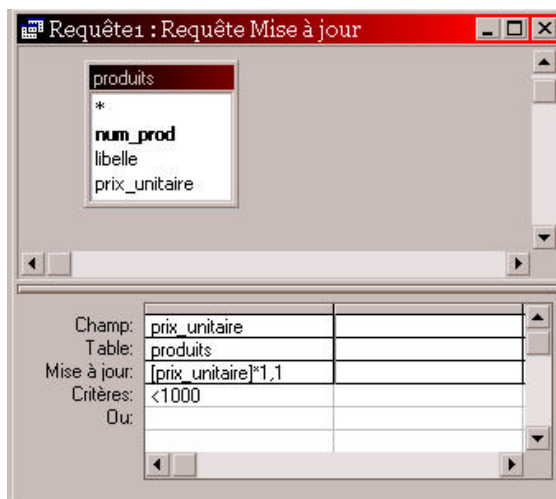
- ?? Les enregistrements sont *COPIES* de la table source vers la table cible et non pas déplacés.
- ?? Les deux tables doivent avoir des noms de champs identiques (et les mêmes types de données), les deux structures ne doivent pas nécessairement être identiques.
- ?? Si la table source comporte plus de champs que la table cible, les champs supplémentaires sont ignorés
- ?? Si la table source comporte moins de champs que la table cible, les champs dont les noms sont identiques sont copiés, les autres sont laissés vides.
- ?? Access ne copie *QUE* les champs que vous avez déclarés dans la requête (d'ou le champ *)

7.3 Les requêtes Mise à Jour

Les requêtes mise à jour permettent de modifier rapidement tous les enregistrements d'une table ou un groupe d'entre eux :

Supposons que l'on veuille augmenter de 10% le prix des produits dont le prix actuel est inférieur à 1000 F.

1. On crée une requête, et dans le menu Requête, on clique sur "Requête Mise à Jour"
2. Un champ "Mise à jour" apparaît dans la requête, c'est là qu'on va indiquer la modification qui va avoir lieu :

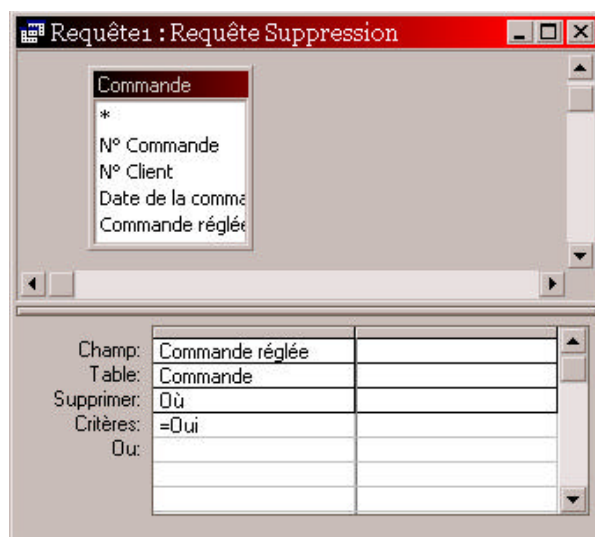


Dans la case Mise à Jour, on a indiqué, qu'à la place du prix unitaire, nous voulions [prix_unitaire]*1.1 (soit une augmentation de 10%), ceci pour les produits dont le prix est inférieur à 1000 (critère).

7.4 Les requêtes Suppression

Les requêtes suppression vous permettent de supprimer un groupe d'enregistrements qui répondent à un critère donné. Supposons que l'on veuille supprimer de la table commande toutes les commandes réglées :

1. On crée une requête standard, puis dans le menu Requête, on clique sur Requête Suppression :



2. On exécute la requête ou on vérifie avant que le résultat est conforme à ce que l'on attend.

Attention :

Vérifiez bien le résultat avant d'exécuter la requête, il n'est pas possible de revenir en arrière après avoir effacé des enregistrements.

1.	Qu'est-ce que SQL ?.....	2
2.	La maintenance des bases de données.....	2
2.1	La commande CREATE TABLE.....	3
2.2	La commande ALTER TABLE	4
2.3	La commande CREATE INDEX	4
3.	Les manipulations des bases de données	5
3.1	La commande INSERT	5
3.1.1	Présentation & syntaxe.....	5
3.1.2	Insérer tout ou une partie d'un enregistrement	6
3.1.3	Insérer plusieurs enregistrements ou plusieurs parties d'enregistrements.....	7
3.2	La commande UPDATE.....	7
3.3	La commande DELETE	8
3.4	La commande SELECT	9
3.4.1	Présentation & Syntaxe	9
3.4.2	Les opérateurs de condition	9
3.4.3	Opérateurs logiques.....	10
3.4.4	Clauses IN et BETWEEN	11
3.4.5	La clause LIKE	11
3.4.6	Les jointures	12
3.4.7	Supprimer les doubles avec DISTINCT	12
3.4.8	Les fonctions d'ensemble.....	13
3.4.9	La clause GROUP BY	13
3.4.10	Les sous-requêtes	14
3.4.11	Les UNIONS.....	15
4.	Les commandes de contrôle des bases de données	15

SQL

Jusqu'à présent, nous avons vu comment créer une requête en utilisant la fenêtre requête d'ACCESS. Il existe une autre façon de faire les requêtes : en passant directement par le langage SQL. D'ailleurs, ACCESS utilise aussi ce langage pour faire ses requêtes, il traduit ce que vous avez entré en SQL, puis exécute la requête SQL.

1. Qu'est-ce que SQL ?

SQL est un langage de manipulation de bases de données mis au point dans les années 70 par IBM. Il permet, pour résumer, trois types de manipulations sur les bases de données :

- ?? La maintenance des tables : création, suppression, modification de la structure des tables.
- ?? Les manipulations des bases de données : Sélection, modification, suppression d'enregistrements.
- ?? La gestion des droits d'accès aux tables : Contrôle des données : droits d'accès, validation des modifications.

L'intérêt de SQL est que c'est un langage de manipulation de bases de données standard, vous pourrez l'utiliser sur n'importe quelle base de données, même si, à priori, vous ne connaissez pas son utilisation. Ainsi, avec SQL, vous pouvez gérer une base de données Access, mais aussi Paradox, dBase, SQL Server, Oracle ou Informix par exemple (les bases de données les plus utilisées).

Attention : Le langage SQL a souvent été implémenté de façon différente. Les commandes de base sont toujours les mêmes mais ont parfois des variantes ou des extensions. La version de SQL implémentée dans Access peut parfois être un peu différente de la version ANSI qui est la version standardisée.

2. La maintenance des bases de données

La première série de commandes sert à la maintenance de la base de données : création des tables et des indexes, modification de la structure d'une table ou suppression d'une table ou d'un index.

Attention : La maintenance des bases de données dépend étroitement de la base de données utilisée, notamment en ce qui concerne les types de données, vérifiez donc toujours les types de données supportés par votre SGBD avant de créer une table avec SQL. Il se peut qu'il en supporte plus que ceux indiqués plus bas.

SQL dispose pour cela des instructions suivantes :

- ?? ALTER TABLE
- ?? CREATE TABLE
- ?? CREATE INDEX

2.1 La commande CREATE TABLE

La commande CREATE TABLE permet de créer une table dans la base de données courante. Sa syntaxe est la suivante :

```
CREATE TABLE    table
                  (champ type CONSTRAINT champ propriétés, ... );
```

Paramètre	Signification
Champ	Nom du champ
Type	Type de données, dans la plupart des versions de SQL, vous aurez droit aux types de données suivants : <ul style="list-style-type: none"> ?? Char (x) : chaîne de caractères, x est le nombre maximum de caractères autorisés dans le champ. ?? Integer : Nombre entier, positif ou négatif ?? Decimal (x , y) : Nombre décimal, x est le nombre maximum de chiffres et y le nombre maximum de chiffres après la virgule. <i>Decimal ne fonctionne pas avec Access, il ne supporte que le type 'float' (flottant), le type float ne permet pas d'indiquer le nombre de chiffres après ou avant la virgule</i> ?? Date : Une date et/ou heure ?? Logical – Deux valeurs possibles : oui / non
propriétés	Propriétés du champ : <ul style="list-style-type: none"> ?? NULL ou NOT NULL : autorise ou non que le champ puisse être vide. ?? UNIQUE : indique que deux enregistrements ne pourront avoir la même valeur dans ce champ. ?? PRIMARY KEY : indique que ce champ est la clef primaire ?? CHECK (condition) : équivaut à la propriété "ValideSi" d'Access, va forcer SQL a faire une vérification de la condition avant de valider la saisie, exemple : CHECK (prix > 100) interdira la saisie dans ce champ si la valeur contenue dans le champ prix est inférieure à 100. <i>CHECK ne fonctionne pas avec Access.</i> ?? DEFAULT = valeur, place une valeur par défaut dans le champ (ne fonctionne pas avec Access) <i>DEFAULT ne fonctionne pas avec Access</i>

Exemple : Créer la nouvelle table "table_test" contenant deux champs : un champ avec un entier qui doit toujours être saisi et un champ contenant une chaîne de 5 caractères :

```
CREATE TABLE table_test (champ1 integer CONSTRAINT champ1 NOT
NULL, champ2 char(5));
```

2.2 La commande ALTER TABLE

La commande ALTER TABLE permet de modifier la structure d'une table, sa syntaxe est la suivante :

2.2.1 ALTER TABLE table Action (spécifications du champ);

ALTER TABLE permet trois actions, ces actions sont :

- ?? **ADD** Ajoute un champ a une table
- ?? **DROP** Supprime un champ d'une table
- ?? **MODIFY** Modifie les caractéristiques d'un champ

Après l'action, on indique, entre parenthèses, les spécifications du champ de la même façon que pour la commande CREATE TABLE. On ne peut faire qu'une action à la fois (ajout, suppression ou modification dans la même commande).

Exemple :

Ajout d'un champ Date :

```
ALTER TABLE table_test ADD champ3 Date;
```

Suppression du champ2 :

```
ALTER TABLE table_test DROP champ2;
```

2.3 La commande CREATE INDEX

La commande CREATE INDEX permet de créer un index sur une table, sa syntaxe est :

```
CREATE UNIQUE INDEX nom_index  
ON table (liste de champs);
```

Pour vous souvenir ce qu'est un index, retournez voir le chapitre 1.

Nom_index est le nom de l'index, table est le nom de la table.

Nous avons vu dans le chapitre 1 qu'un index peut être composé d'un ou de plusieurs champs, la liste de ces champs est indiquée entre parenthèses après le nom de la table.

Enfin, la clause UNIQUE indique à SQL si l'index créé va être unique ou s'il peut contenir plusieurs fois la même valeur dans la table, s'il peut contenir plusieurs fois la même valeur, il ne faudra pas ajouter la clause UNIQUE.

Exemple : création d'un index sur le champ1

```
CREATE UNIQUE INDEX index1 ON table_test (champ1);
```

3. Les manipulations des bases de données

Une fois les tables créées, on peut commencer à y insérer des données, les mettre à jour, les supprimer ou y faire des requêtes. Toutes ces opérations sont des opérations de manipulation des bases de données.

Pour effectuer ces manipulations, SQL dispose de 5 instructions :

```
?? INSERT
?? UPDATE
?? DELETE
?? SELECT
?? CREATE VIEW (non utilisé dans Access)
```

3.1 La commande INSERT

3.1.1 Présentation & syntaxe

La commande INSERT est utilisée pour ajouter des enregistrements ou des parties d'enregistrements dans des tables. Elle est utilisée généralement sous deux formes :

1^{ère} forme

```
INSERT
INTO      table (champ1, champ2, ...)
VALUES    ('valeur1', 'valeur2', ...);
```

Cette forme est utilisée lorsqu'on veut insérer un seul enregistrement ou une partie d'un seul enregistrement. On créera un nouvel enregistrement dont le contenu du champ1 sera valeur1, le contenu du champ2 sera valeur2, etc...

2^{ème} forme

```
INSERT
INTO      table (champ1, champ2, ...)
              (requête);
```

Dans cette seconde forme, le résultat de la requête va être inséré dans les champs indiqués de la table. Cette méthode est utilisée lorsque plusieurs enregistrements sont ajoutés simultanément.

Dans les deux cas, les valeurs insérées doivent correspondre au type de données du champ dans lequel l'insertion va être faite, on ne peut pas, par exemple demander l'insertion d'une chaîne de caractères dans un champ de type numérique ou monétaire. Les chaînes de caractères doivent être placées entre apostrophes ('), les champs numériques ou vides (NULL) ne doivent pas être placés entre apostrophes.

3.1.2 Insérer tout ou une partie d'un enregistrement

Si des valeurs doivent être insérées dans tous les champs de l'enregistrement de la table, la liste des noms des champs n'a pas besoin d'être explicitement indiquée dans la commande. Les valeurs des champs à insérer doivent cependant apparaître dans le même ordre que les noms des champs lors de la création de la table, sans oublier un seul champ.

Faites attention en utilisant cette syntaxe, si la structure de la table change plus tard, la commande qui était bonne risque de ne plus fonctionner correctement. Il vaut mieux toujours indiquer explicitement le nom des champs sur lesquels on veut agir.

La syntaxe est alors :

```
INSERT  
INTO      table  
VALUES   ('valeur1','valeur2','valeur3',...);
```

Par exemple, dans notre table Client, si nous voulons insérer un nouveau client, nous allons entrer :

```
INSERT  
INTO Clients  
VALUES (100,'Mr','Dupond','Jean','rue de la paix','75000',  
. 'Paris'.NNTT.);
```

La commande ci-dessus va insérer un enregistrement dans la table 'clients' avec les informations suivantes : Le client 100, dont le titre est 'M.', dont le nom est 'Dupond', dont le prénom est 'Jean', l'adresse est 'Rue de la paix', le code postal est 75000 et la ville est 'Paris'.

Notez que le numéro de client, qui est un champ de type 'NuméroAuto' a été indiqué explicitement. Lors d'une création d'un enregistrement avec une commande SQL, le numéro automatique n'est pas toujours généré automatiquement, vérifiez les possibilités offertes par votre SGBD.

Notez aussi l'utilisation de NULL : Si on insère moins de valeurs de champ qu'il y a de champs dans la table, soit on écrit explicitement le nom des champs à insérer, soit on insère la valeur NULL dans le champ où il n'y a rien à mettre, à condition que la propriété du champ "NULL interdit" soit à NON, sinon, il y aura une erreur lors de la création de l'enregistrement.

La syntaxe est : *INSERT*
INTO table (champ1, champ3)
VALUES ('valeur1','valeur 3');

Ou *INSERT*
INTO table
VALUES ('valeur1',NULL,'valeur3');

Dans notre exemple, le champ 'Observations' à sa propriété "NULL interdit" à NON, on n'est donc pas obligé de saisir des observations, ce qui est le cas ici, on entre donc NULL, ce qui signifie à Access que le champ est vide.

3.1.3 Insérer plusieurs enregistrements ou plusieurs parties d'enregistrements

On peut insérer simultanément plusieurs enregistrements ou parties d'enregistrements dans une table. Dans ce cas, les données insérées vont être récupérées dans une ou plusieurs autres tables. Par exemple, si nous voulons placer dans une table nommée "Clients_stq" les clients habitant Saint-Quentin, nous allons procéder ainsi :

- 1) Créer une table "Clients_stq" avec Access, nous allons supposer que cette table comporte 3 champs: le nom, le prénom et l'adresse.
- 2) Taper la commande :

```
INSERT
INTO      Clients_stq(nom,prenom,adresse)
          (SELECT nom,prenom,adresse
          FROM Clients
          WHERE ville='Saint-Quentin')
```

La partie entre () est une requête SQL, nous verrons cela en détail plus bas, elle va sélectionner les champs nom, prénom et adresse de la table Client pour les enregistrements dont la ville est "Saint-Quentin". Le résultat de cette requête va être inséré dans la table Clients_stq.

La commande INSERT INTO avec requête ne déplace pas les enregistrements de la table "Clients" vers la table "Clients_stq", elle se contente de faire une copie. Pour effacer les enregistrements de la table Clients, nous utiliserons la commande DELETE.

3.2 La commande UPDATE

La commande UPDATE est utilisée pour changer des valeurs dans des champs d'une table. Sa syntaxe est :

```
UPDATE      table
SET        champ1 = nouvelle_valeur1,
           champ2 = nouvelle_valeur2,
           champ3 = nouvelle_valeur3
WHERE condition;
```

La clause SET indique quels champs de la table vont être mis à jour et avec quelles valeurs ils vont l'être. Les champs non spécifiés après la clause SET ne seront pas modifiés.

Par exemple, si nous voulons, dans la table produit, modifier le prix d'un produit dont le nom est "prod1", nous taperons :

```
UPDATE produits
SET prix_unitaire = 1000
WHERE libelle = 'prod1';
```

La commande UPDATE affecte tous les enregistrements qui répondent à la condition donnée dans la clause WHERE. Si la clause WHERE est absente, tous les enregistrements de la table seront affectés.

Par exemple, si nous tapons :

```
UPDATE produits
SET prix_unitaire = 1000;
```

Le prix unitaire de TOUS les produits de la table produit va être modifié.

Tout comme la commande INSERT, la commande UPDATE peut contenir une requête. Dans ce cas la syntaxe est la suivante :

```
UPDATE      table
SET        champ1 = nouvelle_valeur1,
             champ2 = nouvelle_valeur2,
             champ3 = nouvelle_valeur3
WHERE      condition =
             (requête);
```

La requête est une requête faite avec la commande SELECT.

3.3 La commande DELETE

Pour supprimer des enregistrements d'une table, utilisez la commande DELETE. La syntaxe est la suivante :

```
DELETE
FROM      table
WHERE      condition;
```

On ne peut pas supprimer seulement le contenu de quelques champs des enregistrements. La commande DELETE supprime des enregistrements entiers, c'est pour cela qu'il n'est pas nécessaire d'indiquer ici des noms de champs. La condition spécifiée après WHERE va déterminer quels sont les enregistrements à supprimer.

Par exemple, pour supprimer tous les clients dont la ville est Saint-Quentin :

```
DELETE
FROM Clients
WHERE ville='Saint-Quentin';
```

Pour supprimer tous les enregistrements d'une table, n'indiquez pas de clause WHERE :

```
DELETE FROM table;
```

Cette variante de la commande DELETE ne supprime pas la table, elle supprime seulement les enregistrements contenus dans cette table et laisse une table vide.

On peut aussi, comme précédemment utiliser une requête qui servira à déterminer la condition de la suppression. La syntaxe est la suivante :

```
DELETE
FROM      table
WHERE      condition =
             ( requête );
```


3.4 La commande SELECT

3.4.1 Présentation & Syntaxe

La commande SELECT est la commande la plus complexe de SQL. Cette commande va servir à faire des requêtes pour récupérer des données dans les tables. Elle peut être associée à une des commandes de manipulation de tables vues avant pour spécifier une condition.

Sa syntaxe est :

```
SELECT    champ1, champ2, champ3, ...
FROM      table;
```

S'il y a plus d'un champ spécifié après SELECT, les champs doivent être séparés par des virgules. Les champs sont retournés dans l'ordre spécifié après la clause SELECT, et non pas dans l'ordre qu'ils ont été créés dans la table.

Par exemple :

Pour sélectionner les champs "prénom" et "nom" de tous les enregistrements de la table Clients :

```
SELECT prénom,nom
FROM clients
```

Va renvoyer les prénom et nom de tous les clients de la table Clients.

Si on veut récupérer tous les champs des enregistrements sélectionnés, la syntaxe est la suivante :

```
SELECT    *
FROM      table;
```

Les clauses SELECT et FROM doivent obligatoirement apparaître au début de chaque requête, on peut, ensuite, indiquer des critères de sélection avec la clause WHERE :

```
SELECT    *
FROM      table
WHERE     condition;
```

Par exemple, pour sélectionner tous les Clients de la table "Clients" dont le code postal est 75000 :

```
SELECT *
FROM Clients
WHERE code_postal = 75000;
```

3.4.2 Les opérateurs de condition

On peut utiliser les opérateurs suivants dans les conditions :

Opérateur	Signification
=	Egal
<>	Différent (parfois noté aussi !=)
<	Inférieur
>	Supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal

Pour sélectionner tous les articles dont le prix est supérieur à 100 F :

```
SELECT *
FROM Clients
WHERE prix_unitaire > 100;
```

3.4.3 Opérateurs logiques

Il est possible de combiner plusieurs conditions avec des opérateurs logiques :

L'opérateur **AND** réunit deux ou plusieurs conditions et sélectionne un enregistrement seulement si cet enregistrement satisfait TOUTES les conditions listées. (C'est-à-dire que toutes les conditions séparées par AND sont vraies). Par exemple, pour sélectionner tous les clients nommés 'Dupond' qui habitent Saint-Quentin :

```
SELECT *
FROM Clients
WHERE nom = 'Dupond' AND ville = 'Saint-Quentin';
```

L'opérateur **OR** réunit deux conditions mais sélectionne un enregistrement si UNE des conditions listées est satisfaite. Par exemple, pour sélectionner tous les clients nommés 'Dupond' ou 'Durant' :

```
SELECT *
FROM Clients
WHERE nom = 'Dupond' OR nom = 'Durant';
```

AND et **OR** peuvent être combinés :

```
SELECT *
FROM Clients
WHERE nom = 'Dupond' AND (ville = 'Saint-Quentin' OR ville =
'Paris');
```

Nous sélectionnons ici les clients nommés "Dupond" qui habitent soit à Saint-Quentin, soit à Paris. Pourquoi avons-nous placé des parenthèses ? Pour résumer, on peut dire que l'opérateur AND a une plus grande priorité que l'opérateur OR. Ce qui signifie que SQL va d'abord sélectionner les conditions séparées par des AND, puis celles séparées par des OR, si on avait omis les parenthèses ici, SQL aurait cherché les clients nommés "Dupond" vivant à Saint-Quentin ou les clients habitant à Paris, ce qui n'est pas le but recherché. Pour généraliser, mettez toujours des parenthèses pour bien séparer vos conditions.

3.4.4 Clauses IN et BETWEEN

Pour sélectionner des enregistrements dont la valeur d'un champ peut être comprise dans une liste ou entre deux valeurs, on utilise les clauses IN et BETWEEN.

Par exemple : Pour sélectionner les clients vivant à Saint-Quentin ou Paris :

```
SELECT *
FROM Clients
WHERE ville IN ('Saint-Quentin', 'Paris');
```

Ou pour sélectionner les produits dont le prix est compris entre 100 et 1000 F :

```
SELECT *
FROM Produits
WHERE prix_unitaire BETWEEN 100 AND 1000;
```

Pour sélectionner les produits dont le prix n'est pas dans cet intervalle :

```
SELECT *
FROM Produits
WHERE prix_unitaire NOT BETWEEN 100 AND 1000;
```

De la même façon, NOT IN sélectionne les enregistrements exclus de la liste spécifiée après IN.

Notez que NOT a une priorité plus grande que AND :

NOT Condition1 AND Condition2 sélectionnera les enregistrements ne satisfaisant pas la condition1 et satisfaisant la condition2, alors que NOT (Condition1 AND Condition2) sélectionnera les enregistrements ne satisfaisant pas les deux conditions 1 et 2. Une fois de plus, n'hésitez pas à mettre des parenthèses !

3.4.5 La clause LIKE

La clause LIKE permet de faire des recherches approximatives sur le contenu d'un champ. Par exemple, pour sélectionner les clients dont le nom commence par la lettre D :

```
SELECT *
FROM Clients
WHERE nom LIKE 'S*';
```

Tout comme dans les requêtes Access, le symbole * remplace un ensemble de caractères, pour représenter tous les noms commençant par S, on utilisera 'S*', tous ceux se terminant par S, on utilisera '*S', et tous ceux comportant la lettre S : '*S*'. Le symbole ? ne remplace qu'un seul caractère. Si on a deux clients nommés Dupond et Dupont, on utilisera 'Dupon?'.

*Attention : certaines versions de SQL n'utilisent pas les caractères * et ? mais d'autres caractères spécifiques, certaines versions utilisent notamment le caractère % à la place de *. Consultez donc la documentation de votre SGBD.*

3.4.6 Les jointures

La jointure va nous permettre de sélectionner des informations dans plusieurs tables grâce aux relations existant entre ces tables. Il va néanmoins falloir indiquer comment se fait la relation entre ces tables.

Par exemple : récupérer le nom et le prénom du client ayant passé la commande n°1 :

```
SELECT nom, prénom
FROM Clients, Commande
WHERE Commande.num_client = Client.num_client AND num_commande = 1;
```

La clause WHERE indique que le numéro de commande doit être égal à 1 et que la jointure va se faire sur le numéro de client : une fois que SQL va trouver la commande n° 1 dans la table commande, il va prendre le numéro de client contenu dans l'enregistrement et avec ce numéro, aller chercher dans la table Clients le nom et le prénom correspondants à ce numéro.

Notez que lorsqu'on utilise plusieurs tables, il faut faire attention que deux tables n'aient pas de champs ayant le même nom, si c'est le cas, et pour les différencier, on utilise, comme dans l'exemple, la notation: table.nom_du_champ. Si on est sûr que le nom ne se retrouvera pas dans plusieurs tables, on peut l'utiliser sans le préfixer avec le nom de la table.

3.4.7 Supprimer les doubles avec DISTINCT

Supposons que nous voulions la liste des clients ayant acheté quelque chose. Nous voulons que chaque client ayant acheté quelque chose ne soit affiché qu'une seule fois – nous ne voulons pas savoir ce qu'a acheté chaque client – nous voulons juste connaître les clients qui ont acheté quelque chose. Pour cela, nous allons devoir dire à SQL de supprimer les doubles du résultat de la sélection pour n'afficher les clients qu'une seule fois. Pour cela, nous allons utiliser la clause DISTINCT.

Nous allons d'abord faire une jointure entre les tables Clients et Commande, et ajouter la clause DISTINCT sur le champ ou la répétition peut se produire :

```
SELECT Client.num_client, nom, prénom
FROM Clients, Commande
WHERE Commande.num_client = Client.num_client;
```

Si on exécute cette requête directement, SQL va nous renvoyer une liste des numéros, prénom et nom correspondants aux noms et prénoms des clients ayant passé chaque commande, il est clair qu'un client ayant passé plusieurs commandes va se retrouver plusieurs fois dans cette liste.

```
SELECT DISTINCT Clients.num_client, nom, prénom
FROM Clients, Commande
WHERE Commande.num_client = Client.num_client AND num_commande = 1;
```

En indiquant la clause DISTINCT avant le champ num_client, on indique à SQL qu'on ne veut pas voir apparaître plusieurs fois un client ayant ce numéro dans la sélection renvoyée.

On peut même rendre le résultat de la sélection plus agréable à la lecture en utilisant la clause ORDERBY :

```
SELECT nom, prénom
FROM Clients, Commande
WHERE Commande.num_client = Client.num_client AND num_commande = 1
ORDER BY nom, prénom;
```

La sélection renvoyée va être classée alphabétiquement d'abord sur le nom, puis sur le prénom.

3.4.8 Les fonctions d'ensemble

SQL a cinq fonctions importantes : SUM, AVG, MAX, MIN et COUNT. On les appelle fonctions d'ensemble parce qu'elles résument le résultat d'une requête plutôt que de renvoyer une liste d'enregistrements.

Fonction	Signification
SUM ()	Donne le total d'un champ de tous les enregistrements satisfaisant la condition de la requête. Le champ doit bien sur être de type numérique
AVG ()	donne la moyenne d'un champ de tous les enregistrements satisfaisant la condition de la requête
MAX ()	donne la valeur la plus élevée d'un champ de tous les enregistrements satisfaisant la condition de la requête
MIN ()	Donne la valeur la plus petite d'un champ de tous les enregistrements satisfaisant la condition de la requête.
COUNT (*)	Renvoie le nombre d'enregistrements satisfaisant la requête.

Exemples :

```
SELECT
MIN(prix unitaire),MAX(prix unitaire),AVG(prix unitaire)
```

Va retourner le prix le plus petit de la table Produit, le prix le plus élevé et le prix moyen.

```
SELECT COUNT ( * )
FROM Produits
WHERE libelle LIKE 'P*';
```

Va retourner le nombre de produits dont le libellé commence par la lettre 'P'.

3.4.9 La clause GROUP BY

Une des utilisations les plus courantes de la clause GROUP BY est son association avec une fonction d'ensemble (le plus souvent COUNT, pour compter le nombre d'enregistrements dans chaque groupe). Par exemple, si nous voulons la liste des vendeurs, avec pour chaque vendeur le nombre de ventes qu'il a fait :

```
SELECT num_vendeur ,COUNT ( * )
FROM Commandes
GROUP BY num_vendeur ;
```

On peut ajouter une condition à la requête, par exemple, la liste des vendeurs avec leur nombre de vente pour le mois de janvier.

```
SELECT num_vendeur ,COUNT ( * )
FROM Commandes
GROUP BY num_vendeur
HAVING mois(date)=1;
```

On utilisera pour cela la clause HAVING.

3.4.10 Les sous-requêtes

On peut imbriquer autant de requêtes que l'on veut. La condition après la clause WHERE peut porter sur le résultat d'une autre requête (ou sous-requête).

Supposons les tables suivantes :

Vente
Num_acheteur
Num_produit
Prix

Acheteurs
Num_acheteur
Nom
Prénom

Cette table contient, pour chaque acheteur, le produit qu'il a acheté et le prix d'achat.

Nous voulons la liste des acheteurs ayant acheté des articles chers. Nous considérerons qu'un article cher est un article dont le prix est supérieur à la moyenne du prix des produits achetés + 100 francs.

```
SELECT Num_acheteur
FROM Vente
WHERE prix >
      ( SELECT AVG (prix) + 100
        FROM Vente);
```

Vous pouvez constater que condition de la requête est basée sur le résultat d'une autre requête. Dans cet exemple, à chaque fois qu'un acheteur aura acheté un article cher, son numéro apparaîtra, pour éviter cela, on utilise la clause DISTINCT num_acheteur pour éliminer les doubles.

Autre exemple avec ces deux tables : nous savons qu'il y a une erreur sur l'orthographe du nom de l'acheteur du produit n° 1, il devrait s'appeler 'Dupont' :

```
UPDATE Acheteurs
SET nom = 'Dupont'
WHERE num_acheteur =
      (SELECT num_acheteur
       FROM Ventes
       WHERE num_article = 1);
```

N'oubliez pas cette règle à propos des sous-requêtes : Lorsque vous faites une sous-requête dans la clause WHERE, la clause SELECT de cette sous-requête doit avoir un nombre et des types de champs correspondants à ceux se trouvant après la clause WHERE de la requête principale. Autrement dit, si vous avez "WHERE champ = (SELECT ...);", le résultat du SELECT doit être un seul champ puisqu'il n'y a qu'un seul champ après le WHERE, ET leur type doit correspondre (les deux doivent être numériques ou être des chaînes de caractères, etc.), sinon, la requête ne renverra rien ou sortira avec une erreur selon les systèmes.

3.4.11 Les UNIONS

Lorsqu'on veut que les résultats de plusieurs requêtes soient combinés entre eux, on utilise la clause UNION. UNION va fusionner les résultats des requêtes.

Par exemple, supposons que nous ayons une table pour les clients habitant Saint-Quentin (cette table s'appellera clients_stq) et une table pour les clients habitant Paris (clients_Paris). Pour obtenir les numéros des clients des deux tables, on tapera :

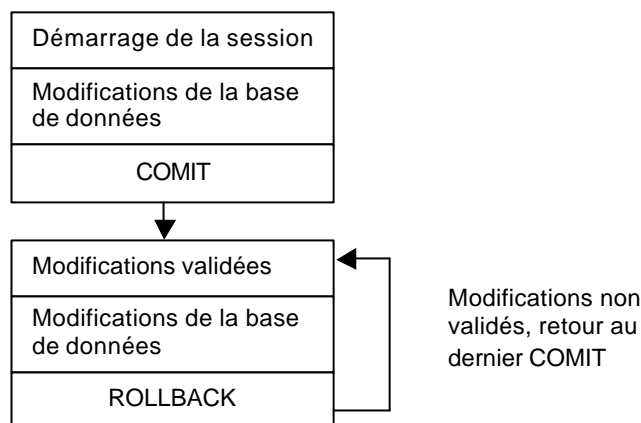
```
SELECT num_client
FROM clients_stq
UNION
SELECT num_client
FROM clients_paris;
```

Notez que SQL requiert que le type des champs sélectionnés dans les clauses SELECT corresponde, colonne par colonne (le premier champ du premier SELECT doit avoir le même type que le premier champ du deuxième SELECT, etc...). Notez aussi que SQL supprime automatiquement les doubles lorsque UNION est utilisé, là où il aurait fallu utiliser la clause DISTINCT pour une requête simple.

4. Les commandes de contrôle des bases de données

Ces commandes ne seront pas détaillées car elles ne sont pas implémentées (ou qu'en partie) dans Access, néanmoins, deux d'entre elles sont très souvent utilisées dans d'autres systèmes : les commandes COMMIT et ROLLBACK.

La commande COMMIT valide toutes les modifications faites depuis le dernier COMMIT, à partir du moment où la commande COMMIT a été entrée, tout ce qui a été fait sur la base est validé et ne peut plus être modifié. La commande ROLLBACK efface toutes les modifications qui ont été faites sur la base depuis le dernier COMMIT. C'est pour cela qu'il faut penser à faire un COMMIT des modifications régulièrement.



Certains systèmes, comme Access, font un COMMIT automatique après chaque action, aucun retour en arrière n'est donc possible.

Présentation	2
1. Créer un formulaire à partir d'une table	3
2. Les contrôles :	10
2.1 Le contrôle "Intitulé"	11
2.2 Le contrôle "Zone de Texte"	12
2.3 Le contrôle « Groupe d'options »	14
2.4 Les contrôles « Traits » et « Rectangle »	17
2.5 Les contrôles « Zone de Liste » et « Zone de Liste Modifiable »	18
2.6 Le contrôle « Bouton de commande »	23
2.7 Les contrôles « Sous-Formulaire »	30

Présentation

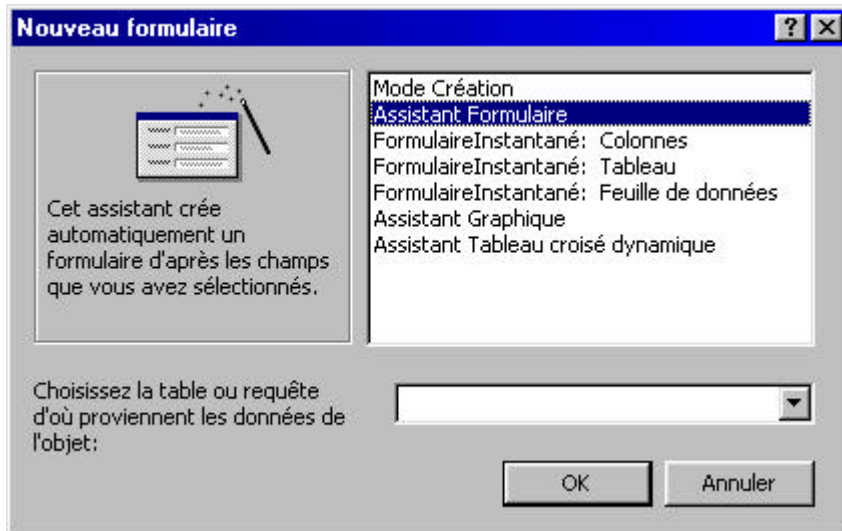
Le formulaire va nous permettre d'afficher et de modifier le contenu d'une table de façon bien plus agréable que le mode « feuille de données » qui ne permet qu'un affichage en lignes et colonnes.

De plus, le mode « feuille de données » ne permet l'affichage et la modification d'informations ne provenant que d'une seule table, le formulaire va nous permettre de manipuler au même endroit des informations provenant de plusieurs tables simultanément : par exemple, dans notre exemple de base de données magasin, nous pourrions avoir un formulaire qui affichera dans la même fenêtre toutes les informations concernant une commande : informations générales sur la commande (provenant de la table « commande », informations sur le client ayant passé cette commande (provenant de la table « clients » et le détail de cette commande (provenant des tables « lignes-commande » et « produits ») alors que précédemment, ces données étaient éclatées sur plusieurs feuilles de données.

Les informations saisies ou modifiées dans le formulaire seront modifiées dans les tables à partir desquelles le formulaire a été créé.

1. Créer un formulaire à partir d'une table

Nous allons créer le formulaire associé à la table « clients ». Pour créer un formulaire, on se place dans la fenêtre principale d'Access et on clique sur l'onglet « formulaire », puis sur le bouton « nouveau... »

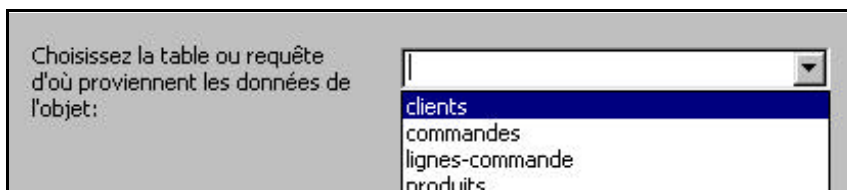


Il existe plusieurs méthodes pour générer un formulaire, nous nous intéresserons seulement aux deux premières méthodes :

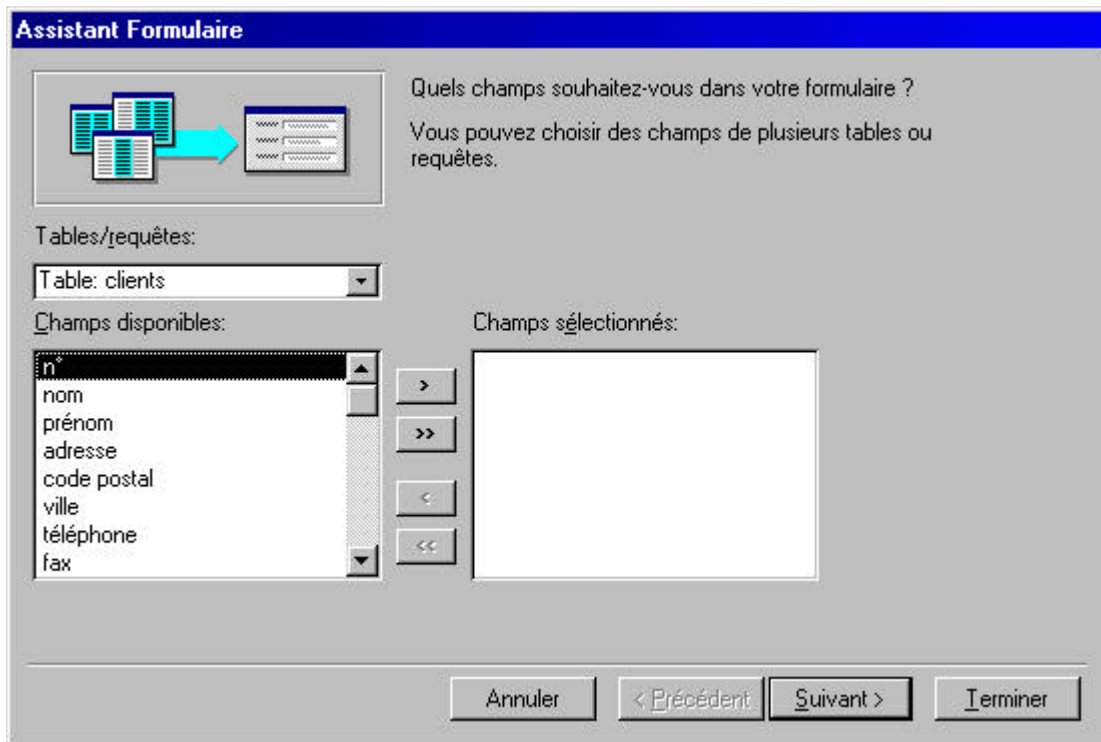
- ?? Mode création : on va tout faire seul « à la main » sans aucune aide d'Access
- ?? Assistant Formulaire : Access va nous guider pas à pas dans la réalisation de notre formulaire
- ?? Formulaires Instantanés : Ces trois méthodes vont générer rapidement un formulaire à partir d'une table sans nous poser de questions, son aspect sera rudimentaire et ne nous donnera pas de grandes possibilités de personnalisations.

Nous allons utiliser la méthode « **Assistant Formulaire** ».

Dans la partie inférieure de la fenêtre, nous allons choisir la table à partir de laquelle le formulaire va être généré : les informations provenant de cette table seront affichées dans le formulaire, les modifications ou ajouts que nous ferons dans le formulaire seront répercutées dans cette table :



Nous choisissons ici la table « clients » et on clique sur « OK »



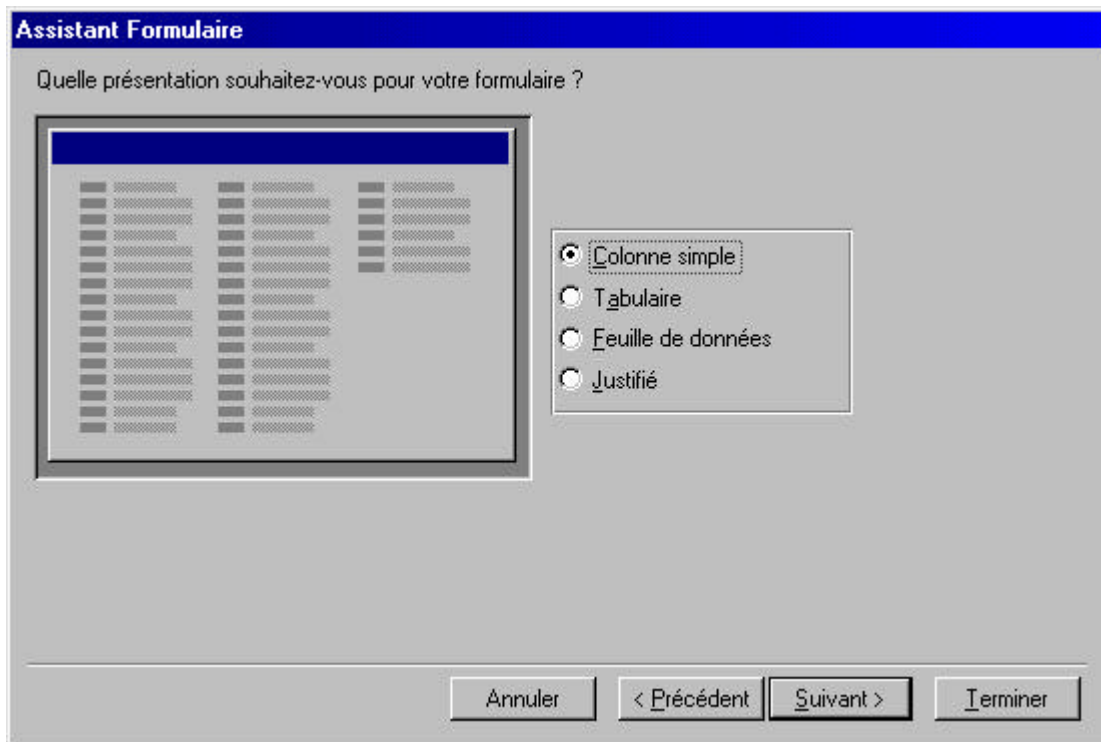
Nous allons choisir ici quels sont les champs de la table que nous voulons voir dans le formulaire, si nous voulons être en mesure de modifier tous les champs de chaque enregistrement de la table, il faudra tous les choisir, si, par exemple, notre formulaire ne doit servir qu'à afficher certaines informations, on choisira uniquement les champs pertinents.

Attention : Dans un formulaire, on va pouvoir, comme dans la feuille de données, ajouter des enregistrements dans la table, les champs non présents dans le formulaire ne seront pas initialisés (ils resteront vides) lors de l'ajout. (Le seul moyen alors pour remplir ces champs sera d'aller dans la feuille de données).

Pour choisir les champs à ajouter ou à enlever dans le formulaire, on va utiliser les boutons :

- ?? > : Ajouter le champ dans le formulaire
- ?? >> : Ajouter tous les champs dans le formulaire
- ?? < : Supprimer un champ du formulaire
- ?? << : Supprimer tous les champs du formulaire

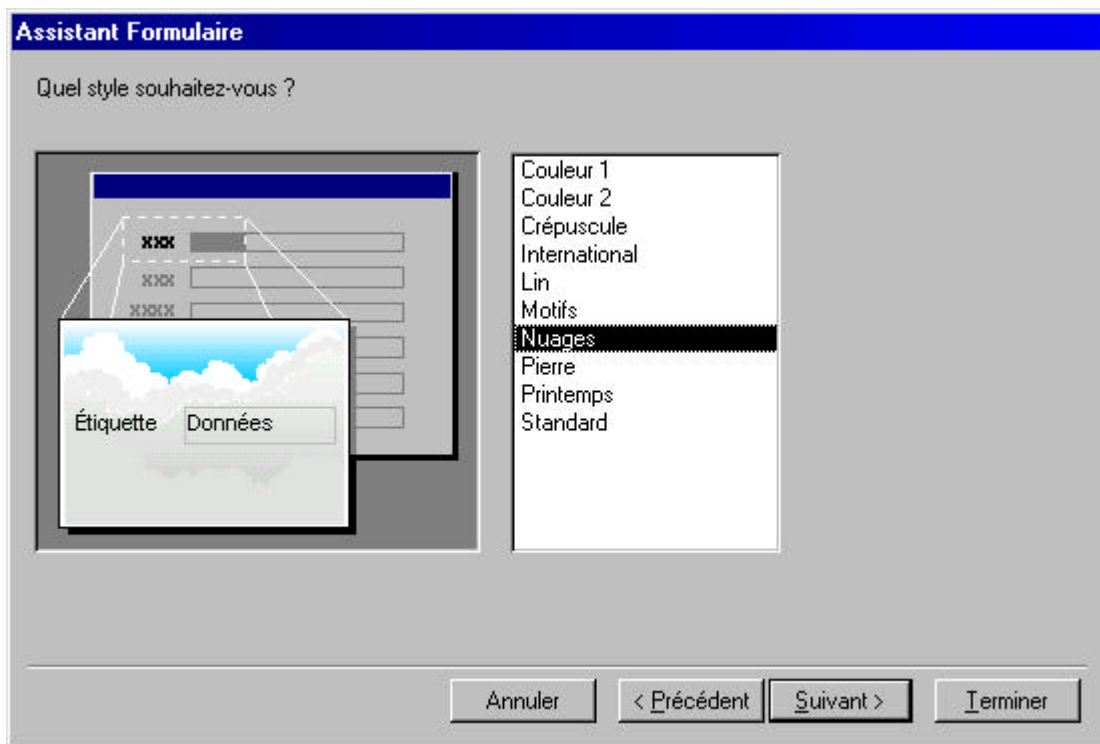
Nous allons ajouter tous les champs du formulaire, cliquez ensuite sur « suivant »



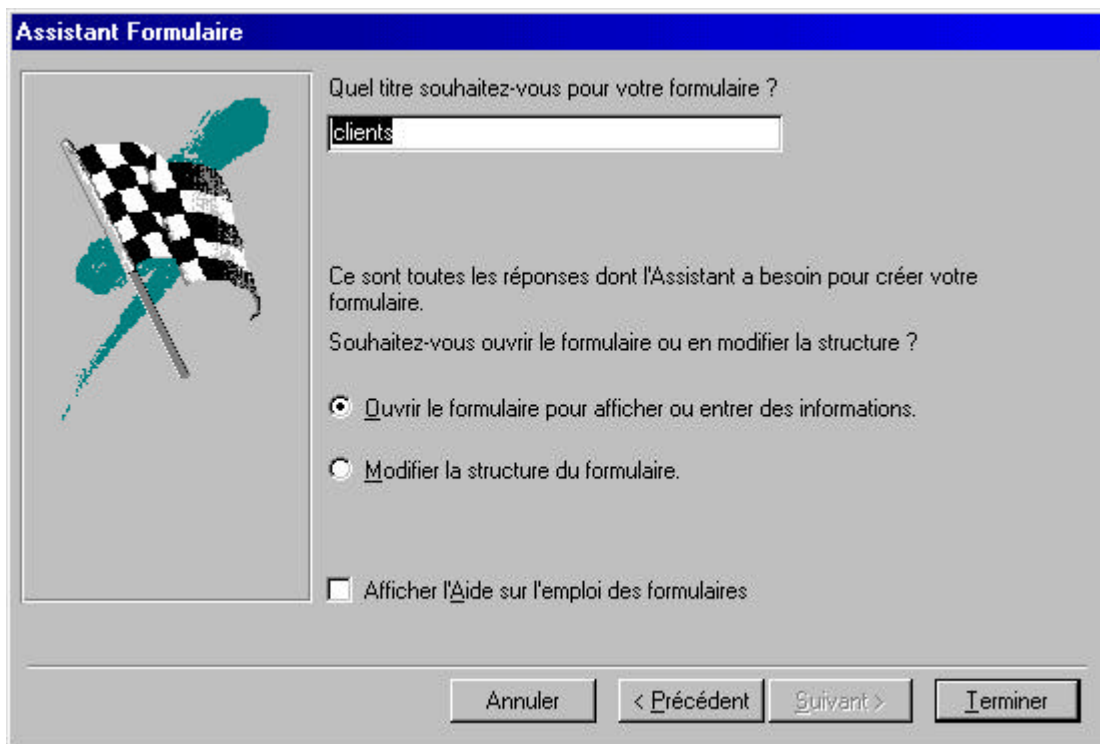
On choisit ici la façon d'afficher les champs choisis dans le formulaire, il y a quatre façons d'afficher les champs :

- ?? **Colonne simple** : Le formulaire va afficher les champs sous la forme : « nom du champ : contenu du champ ». Le formulaire va afficher les champs d'un enregistrement à la fois.
- ?? **Tabulaire** : Le formulaire va afficher les champs choisis sous la forme d'un tableau : en haut du formulaire vont être affichés les noms des champs, et, en dessous sous forme de tableau, le contenu, avec un enregistrement par ligne. Cette façon d'afficher les données ressemble à la feuille de données.
- ?? **Feuille de données** : C'est la même chose que la feuille de données utilisées pour saisir des informations dans une table, la seule différence est que ne sont affichés ici que les champs sélectionnés dans l'étape précédente.
- ?? **Justifié** : Va afficher les champs choisis les un à la suite des autres, ce n'est pas très beau (enfin c'est une question de goût).

Nous utiliserons le plus souvent (voire tout le temps) le mode « Colonne simple » ou le mode « Tabulaire ». Nous choisissons ici le mode « Colonne Simple », puis on clique sur « Suivant »



On choisit ici le "décor" que l'on va donner au formulaire, il existe une dizaine de décors prédéfinis, bien sur, on pourra, par la suite, modifier un décor choisi ici. Choisissez celui qui vous plaît le plus.



Enfin, on va nommer son formulaire, c'est sous ce nom qu'il apparaîtra dans l'onglet "Formulaires" de la fenêtre principale d'Access, puis cliquez sur "Terminer", le formulaire est créé.

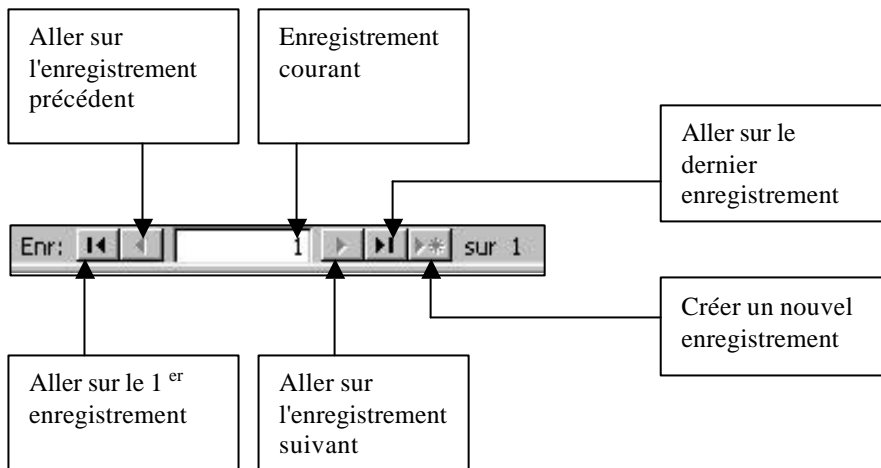


Access nous propose alors de commencer tout de suite la saisie dans notre nouveau formulaire.

Vous pouvez voir ici le formulaire tel qu'il est une fois créé : Il utilise les champs choisis dans la table sélectionnée, il affiche les champs sous forme de "Colonne simple" (un couple nom du champ - contenu du champ par ligne) et utilise le décor choisi.

Les informations qui vont être saisies ou modifiées dans ce formulaire le seront dans la table qui est associée à ce formulaire.

Pour se déplacer parmi les enregistrements dans le formulaire, on utilise les icônes fléchés en bas du formulaire :

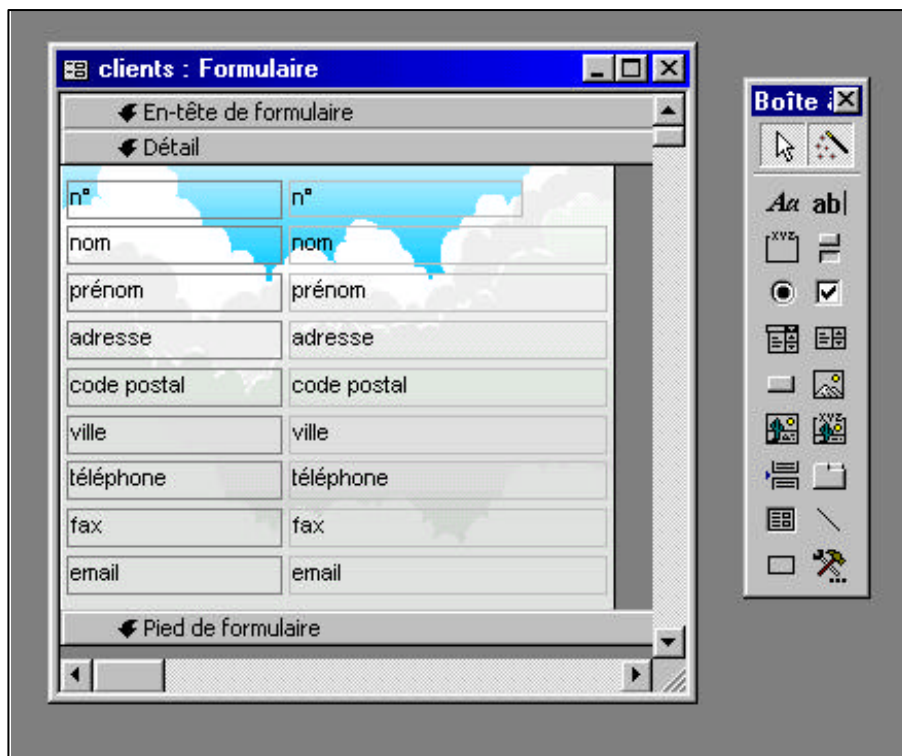


Les différents contrôles

Pour modifier l'aspect et le contenu du formulaire, on clique soit sur le bouton "Modifier" dans l'onglet "Formulaires" de la fenêtre principale d'Access, soit sur l'icône



On passe alors en mode modification :



Le formulaire est affiché en mode modification (à gauche) et une boîte à outils s'affiche (à droite), la boîte à outils peut apparaître sous forme de barre d'icône en dessous du menu de la fenêtre.

Le formulaire est constitué de plusieurs parties : L'en-tête, le détail et le pied. Dans le cas d'un formulaire de type "Colonnes Simples" (comme c'est le cas ici), l'en-tête et le pied n'ont aucune importance, ils sont utiles lorsque le formulaire est affiché sous forme de tableau (type Tabulaire ou Feuille de Données), dans ce cas, l'en-tête va contenir le titre des colonnes, le détail va définir la façon dont vont être affichées les informations dans chaque ligne du tableau (nous verrons ça plus tard) et le pied pourra contenir des informations supplémentaires, comme par exemple, après avoir affiché un tableau de nombres, il pourra afficher la somme de ces nombres.

Tout ce qui se trouve sur le formulaire s'appelle un **contrôle** : un texte affiché sur le formulaire est un contrôle, un champ d'un enregistrement est un contrôle, une image placée sur le formulaire est un contrôle : tout ce qui est affiché dans le formulaire est un contrôle. La boîte à outils à gauche affiche tous les contrôles possibles que l'on peut placer sur un formulaire. Ils sont assez nombreux et permettent d'afficher tous les types d'informations possibles (même du son ou de la vidéo !).

Ces contrôles peuvent être divisés en trois catégories :

?? **Les contrôles indépendants** : Ils n'ont aucune relation avec la table qui est liée au formulaire (n'oubliez pas que chaque formulaire manipule des informations provenant d'une table de la base). Par exemple une image placée sur le formulaire n'a pas de relation avec la table, c'est un contrôle indépendant, du texte affiché sur le formulaire (on pourrait par exemple placer un titre "Formulaire Clients" en haut du formulaire), ce texte ne provient pas de la table, il n'a aucun rapport avec elle, c'est donc un contrôle indépendant.

?? **Les contrôles dépendants** : Les contrôles dépendants sont liés à la table liée au formulaire, ils vont afficher le contenu d'un champ, toute modification dans ce contrôle ira modifier le champ auquel il est lié dans la table :



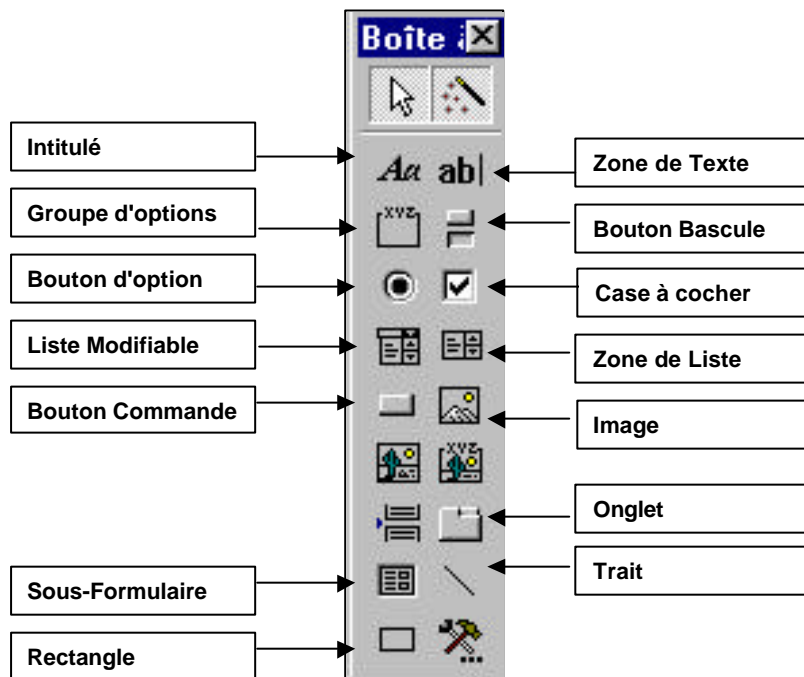
Dans notre exemple, le contrôle à droite est dépendant du champ "nom" de la table "Clients", il affichera, pour l'enregistrement courant, le contenu du champ "nom". Toute modification dans ce contrôle ira modifier le contenu du champ "nom" de la table "Client" pour l'enregistrement en cours.

Notez que le contrôle à gauche est un contrôle indépendant, il se contente d'afficher le texte "nom", tandis que celui de droite est dépendant et va chercher son contenu dans le champ de la table dont le nom est affiché (champ "nom"), nous verrons plus bas comment rendre un contrôle dépendant.

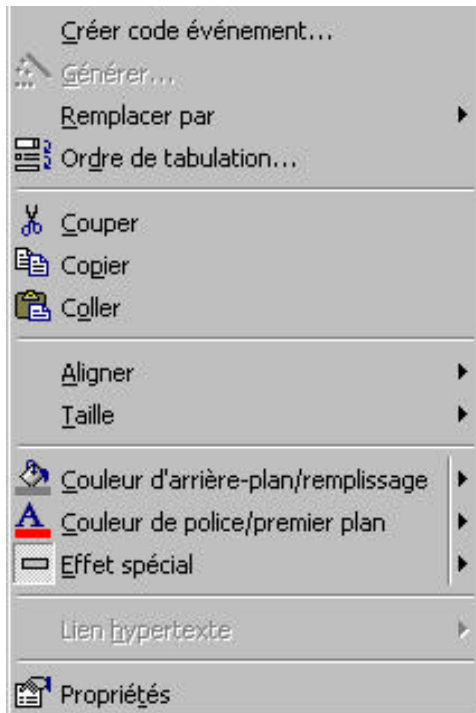
?? **Les contrôles calculés** : Les contrôles calculés ne vont pas chercher les informations qu'ils vont afficher dans un champ de la table lié au formulaire mais à partir du contenu d'autres contrôles du formulaire, par exemple un contrôle dans le formulaire lié à la table "Produit" pourra afficher le prix TTC à partir des contrôles affichant les prix hors taxe et la TVA.

2. Les contrôles :

Voici la liste des contrôles que nous pourrons utiliser, les contrôles non documentés servent, de façon générale, à insérer dans un formulaire des "objets" provenant d'autres applications Windows, comme par exemple un fichier sonore Wav, une vidéo AVI, un fichier au format Word, une page HTML, etc.



Pour tous les contrôles placés sur le formulaire, en cliquant dessus avec le bouton droit, on obtient le menu suivant :



Ce menu permet de modifier l'aspect (couleur, choix de la police, effets, etc.) ou le comportement (format d'affichage, masque de saisie, etc.) du contrôle, l'option "Propriétés" du menu permet l'affichage et la modification de toutes les caractéristiques du contrôle. Les propriétés varient selon le type de contrôle.

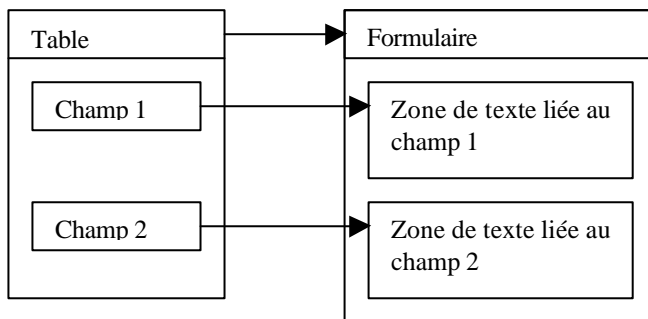
2.1 Le contrôle "Intitulé"

Il va être utilisé pour afficher du texte "statique", c'est à dire qui ne variera pas d'un enregistrement à l'autre, c'est donc un contrôle indépendant, qui servira typiquement à afficher par exemple le titre du formulaire, ou le nom d'un champ dont le contenu sera affiché à côté dans un contrôle dépendant. Dans notre formulaire, tous les noms des champs affichés à gauche sont des contrôles "Intitulé", ce sont des textes statiques ne dépendant de rien, on peut modifier ce qui est affiché sans affecter quoi que ce soit dans la table associée au formulaire.

2.2 Le contrôle "Zone de Texte"

Ce contrôle est l'un des contrôles les plus utilisés dans les formulaires, on va, dans ce contrôle, pouvoir saisir des données. Typiquement, ce contrôle est soit un contrôle dépendant, soit un contrôle calculé.

Pourquoi ? On peut, dans ce contrôle, saisir des informations, il serait alors judicieux que les informations saisies puissent servir à quelque chose, si le contrôle n'est lié à aucun champ de la table associée au formulaire, les informations que l'on va taper vont se perdre dans la nature dès qu'on sera passé sur l'enregistrement suivant. En revanche, si le contrôle est lié à un champ de la table, les informations entrées dans ce contrôle seront placées dans le champ lié au contrôle.



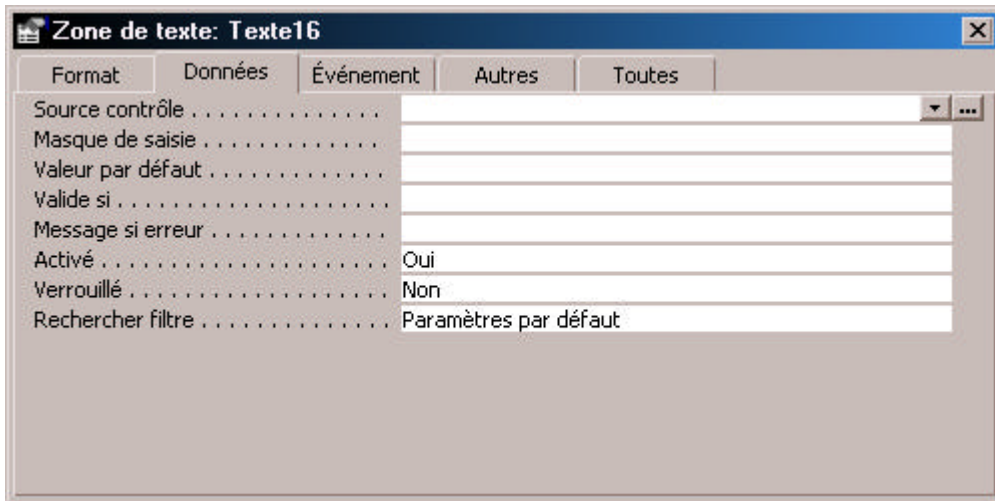
Tout ce qui sera entré dans un contrôle dépendant affectera le contenu du champ auquel il est lié. Bien sur, rien n'interdit de lier un même champ sur plusieurs contrôles dans le même formulaire (ce qui n'a pas, à priori, d'intérêt).

Dans notre formulaire, tous les contrôles dans lesquels on va saisir des informations sur les clients sont des contrôles dépendants liés à des champs de la table. Le nom du champ auquel est lié le contrôle est d'ailleurs écrit dans ce contrôle.

Pour relier un contrôle à un champ, procédez ainsi :

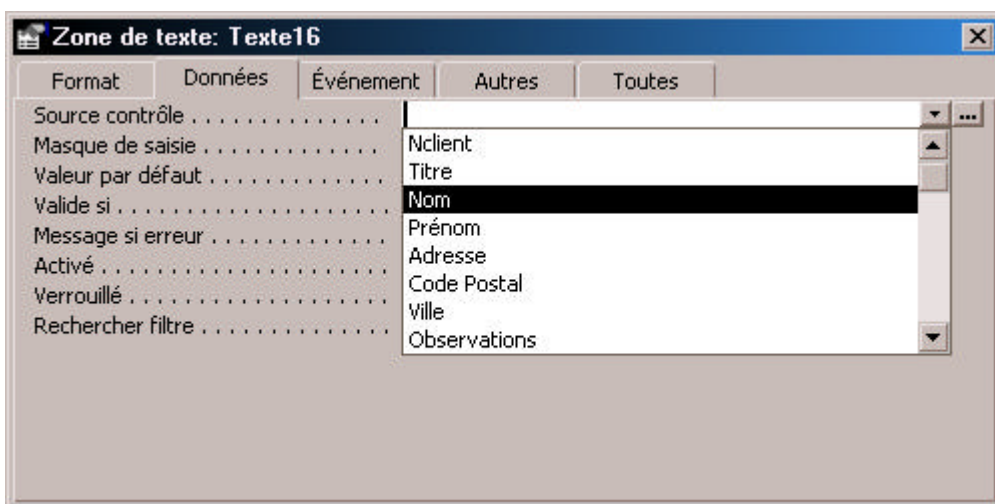
1. Cliquez dans la boîte à outils sur le contrôle zone de texte
2. Placez le contrôle sur le formulaire et dimensionnez-le
3. Access crée en fait DEUX contrôles : il suppose, à juste titre, que l'on va lier ce contrôle avec un champ, il crée donc un contrôle « Intitulé » qu'on a vu avant qui contient le texte « Texte xx » (où xx est un nombre quelconque) et notre fameux contrôle « Zone de Texte » qui contient la mention « indépendant ».
4. On va supposer ici qu'on veut lier ce contrôle au champ « Nom » de notre table Clients (même si il se trouve déjà sur le formulaire, c'est juste un exemple)
5. A la place de « Texte xx », on va taper « Nom : » (histoire de savoir ce qu'il va falloir saisir dans le contrôle d'à côté)

6. Vous avez pu noter la mention « indépendant », ce qui signifie que, pour l'instant, le contrôle « Zone de Texte » créé n'est lié à rien, on peut taper du texte dedans mais le texte ne sera sauvé nulle part. Pour dire à Access qu'on veut sauver ce texte dans le champ « Nom » de la table Clients, on clique avec le bouton droit sur le contrôle et on clique sur « Propriétés » :



Ici se trouvent toutes les propriétés qui vont définir l'aspect et le comportement de notre contrôle, il y en a beaucoup et je ne vais pas les détailler. Celle qui nous intéresse particulièrement est la propriété « Source Contrôle » de l'onglet « Données ». C'est ici qu'on indique à Access d'où viennent les informations qui vont être affichées dans le contrôle.

Il y a une flèche qui descend à droite, si on clique dessus, tous les champs de la (ou des) table(s) liés au formulaire vont s'afficher, on choisit dedans le champ « Nom », et on ferme la fenêtre de propriétés.



7. Le contrôle n'est plus indépendant, à la place de la mention « Indépendant », il est affiché « Nom », c'est-à-dire le nom du champ avec lequel il va être lié. A partir de maintenant, lors de l'affichage d'un enregistrement dans le formulaire, le nom du client sera affiché dans ce contrôle, et toute modification de son contenu sera répercutée dans la table.

2.3 Le contrôle « Groupe d'options »

Le contrôle « Groupe d'option » va servir à faire un choix limité entre plusieurs options et d'affecter l'option choisie à un champ d'une table. Par exemple, supposons maintenant que nous ajoutons un champ « vendeur » à la table « Commandes » pour savoir qui a effectué une vente. Nous allons supposer que notre magasin à trois vendeurs nommés Dupond (vendeur 1), Durant (vendeur 2) et Martin (vendeur 3). Nous allons d'abord ajouter le champ « num_vendeur » à la table « Commandes », puis sur le formulaire « Commandes » liée à cette table, nous ajoutons un contrôle « Groupe d'options »

Assistant Groupe d'options

Un groupe d'options contient un ensemble de cases d'option, de cases à cocher ou de boutons bascule. Vous ne pouvez choisir qu'une option.

Quelle étiquette souhaitez-vous pour chaque option ?

Noms d'étiquette :	
	Dupont
	Durant
✎	Martin
*	

Annuler < Précédent Suivant > Terminer

Nous entrons ici les différents noms des vendeurs et nous cliquons sur « Suivant »

Assistant Groupe d'options

Souhaitez-vous sélectionner une option par défaut pour le groupe d'options ?

Oui, la valeur par défaut est : Dupont

Non, je ne veux pas définir de valeur par défaut.

Annuler < Précédent Suivant > Terminer

Nous pouvons choisir ici le vendeur qui sera toujours proposé par défaut, nous pouvons également ne choisir aucun vendeur par défaut.

Assistant Groupe d'options

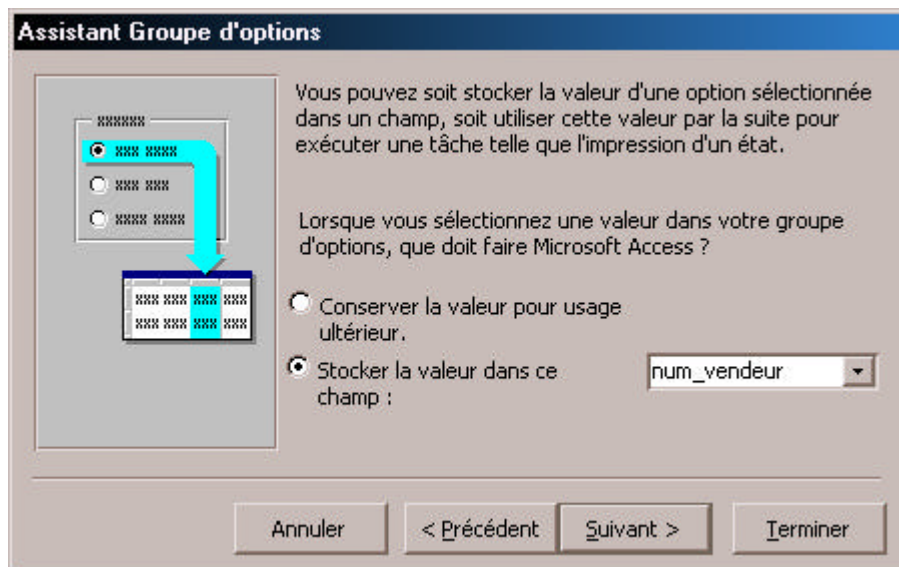
Cliquer sur une option dans un groupe d'options positionne la valeur de ce groupe d'options à la valeur de l'option sélectionnée.

Quelles valeurs souhaitez-vous assigner à chaque option ?

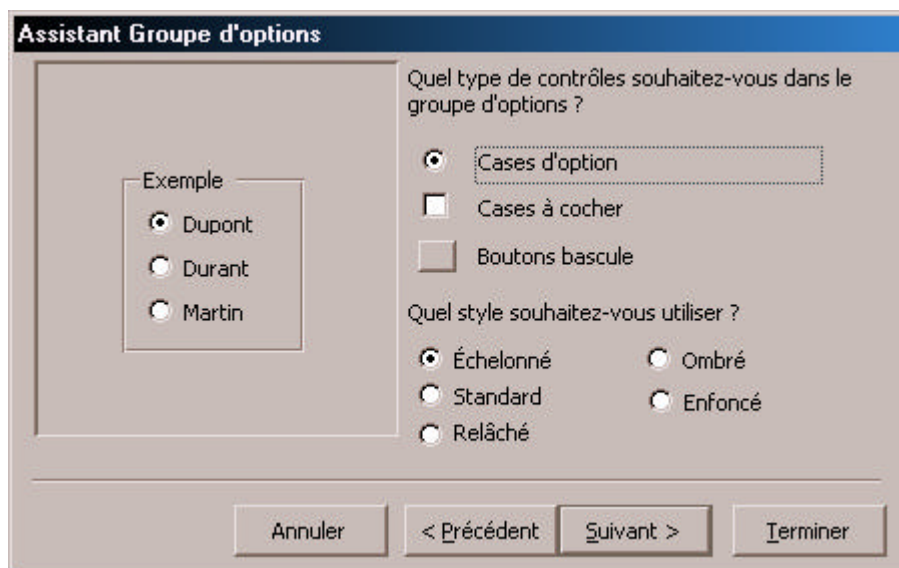
	Noms étiquettes :	Valeurs :
▶	Dupont	1
	Durant	2
	Martin	3

Annuler < Précédent Suivant > Terminer

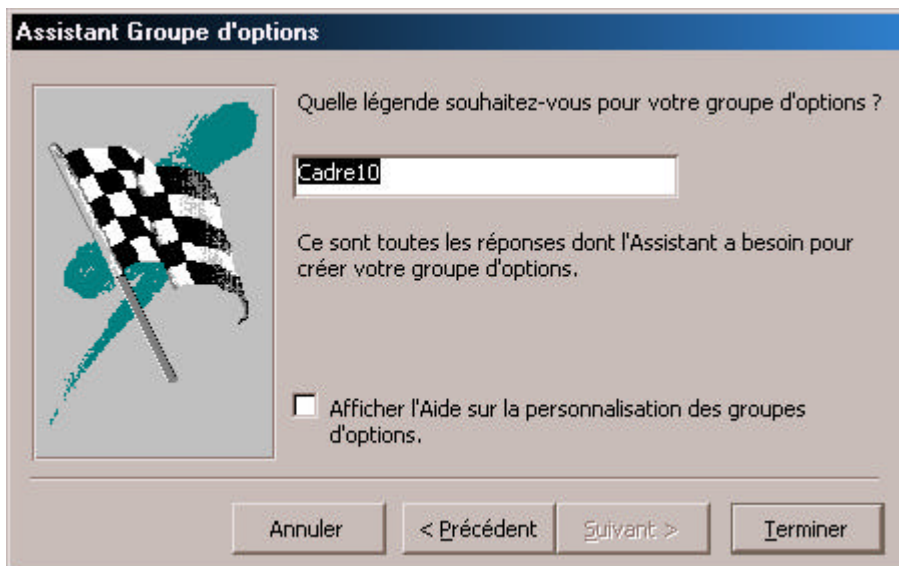
Nous avons vu que chaque vendeur à un numéro, c'est ce numéro, et non le nom du vendeur qui va être sauvé dans le champ « num_vendeur » de la table « Commande », on indique ici, pour chaque vendeur, son numéro.



On indique enfin à Access en choisissant dans le menu quel est le champ de la table qui va contenir le numéro correspondant au vendeur qu'on a choisi.



On choisit l'aspect de notre groupe d'options



Assistant Groupe d'options

Quelle légende souhaitez-vous pour votre groupe d'options ?

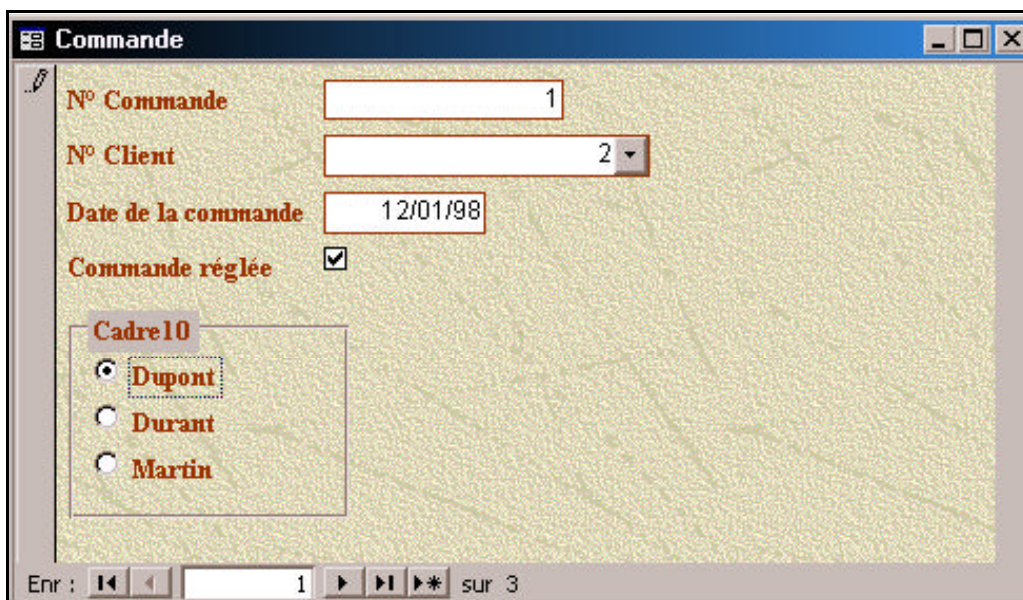
Cadre10

Ce sont toutes les réponses dont l'Assistant a besoin pour créer votre groupe d'options.

Afficher l'Aide sur la personnalisation des groupes d'options.

Annuler < Précédent Suivant > Terminer

On le nomme, et c'est terminé :



Commande

N° Commande 1

N° Client 2

Date de la commande 12/01/98

Commande réglée

Cadre10

Dupont

Durant

Martin

Enr : 1 sur 3

Vous pouvez voir en bas le groupe d'options avec les trois vendeurs proposés, en choisissant un, on sauve dans le champ lié à ce groupe (num_vendeur), le numéro associé au vendeur choisi.

2.4 Les contrôles « Traits » et « Rectangle »

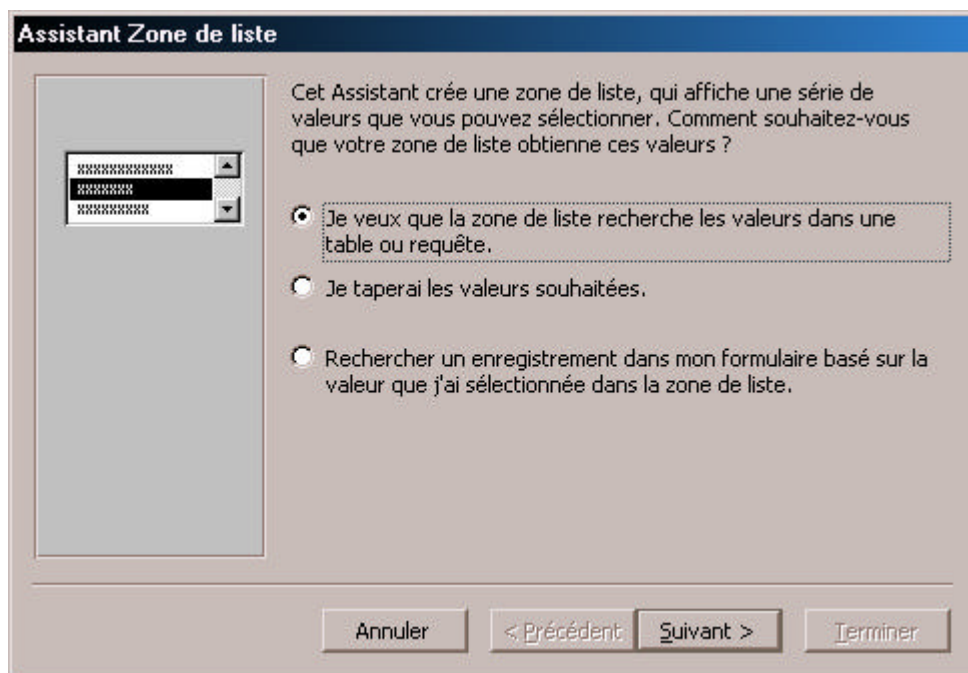
Ces deux types de contrôles permettent de tracer des traits ou des rectangles sur le formulaire pour encadrer ou souligner.

2.5 Les contrôles « Zone de Liste » et « Zone de Liste Modifiable »

Vous vous souvenez de l'assistant «Liste de choix» dans les tables qui permettait, au lieu de saisir une valeur, de pouvoir choisir dans une liste. Ce type de menu existe également dans les formulaires, on va pouvoir, grâce à ces deux types de contrôles, pouvoir, par exemple dans le formulaire « Commande », choisir parmi la liste des clients au lieu de saisir son numéro.

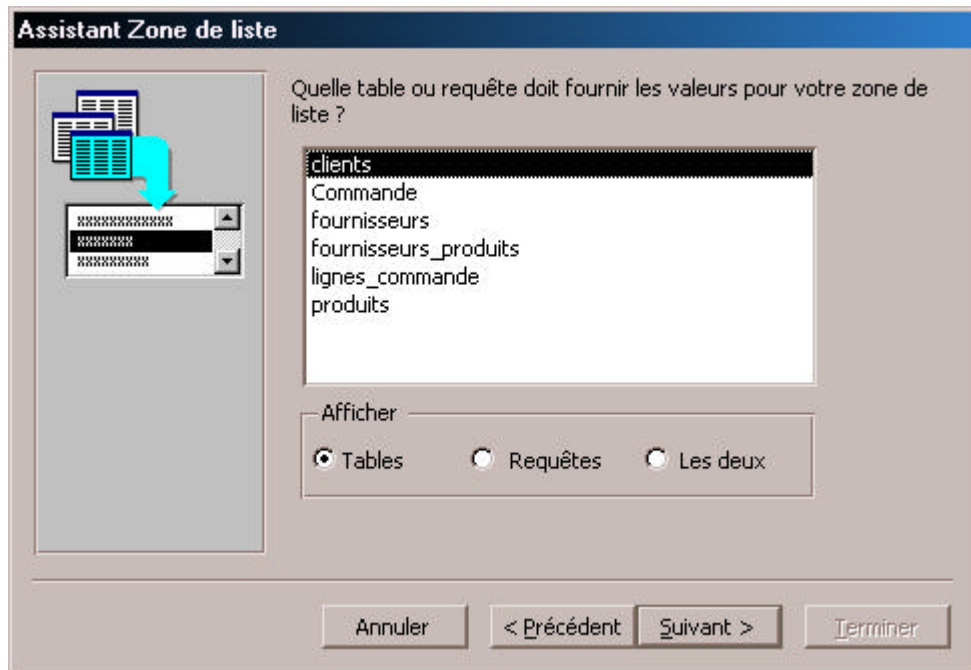
Pour cela, on procède ainsi :

1. Ouvrir le formulaire « Commande » en modification
2. Choisir le formulaire « Zone de Liste » dans la boîte à outils
3. Le placer sur le formulaire



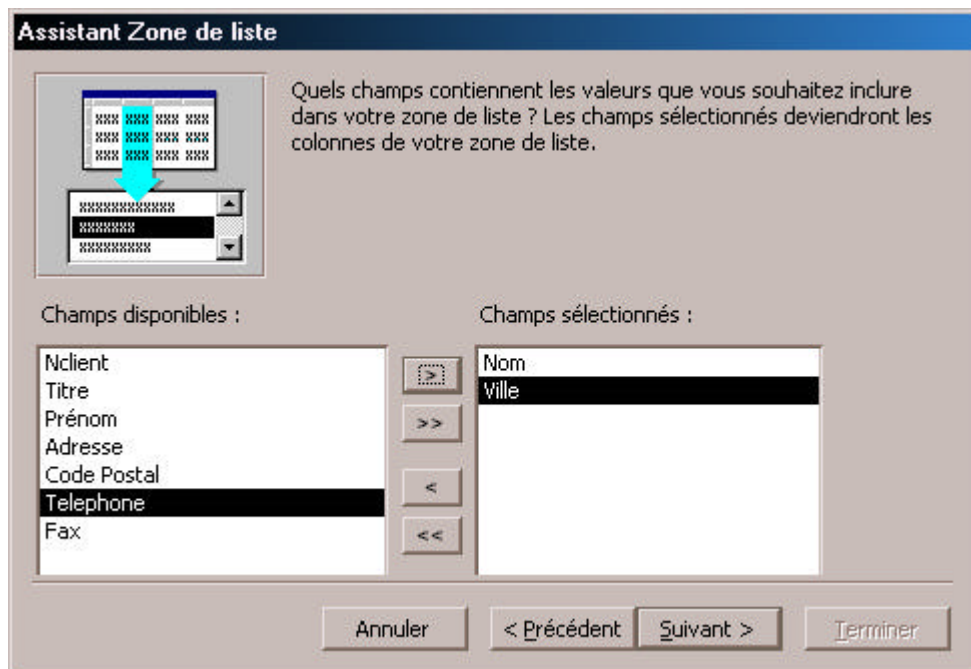
Il y a trois façon de générer cette zone de liste, comme pour la feuille de données, nous voulons afficher une liste des clients existants dans la table client, pour cela, on sélectionne la 1^{ère} option «Je veux que la zone de liste recherche les valeurs dans une table» (en l'occurrence, la table clients), et on clique sur suivant

4.



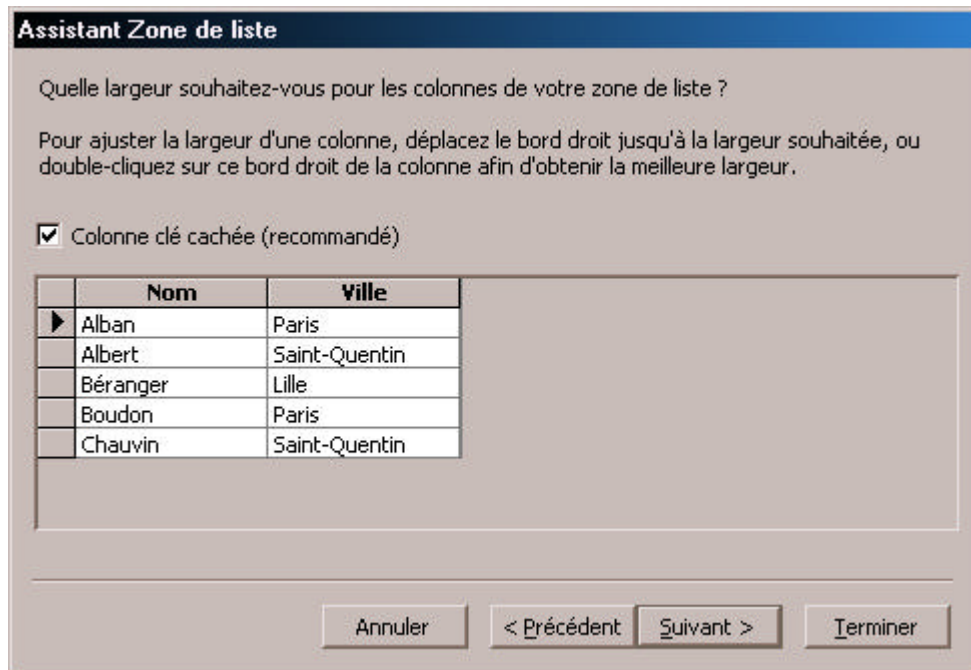
Access nous affiche les tables existantes : On choisi la table « Clients »

5.



On choisit ici les champs qui seront affichés dans la zone de liste, nous pouvons juger que seuls les nom et ville sont nécessaires

6.



Assistant Zone de liste

Quelle largeur souhaitez-vous pour les colonnes de votre zone de liste ?

Pour ajuster la largeur d'une colonne, déplacez le bord droit jusqu'à la largeur souhaitée, ou double-cliquez sur ce bord droit de la colonne afin d'obtenir la meilleure largeur.

Colonne clé cachée (recommandé)

	Nom	Ville
▶	Alban	Paris
	Albert	Saint-Quentin
	Béranger	Lille
	Boudon	Paris
	Chauvin	Saint-Quentin

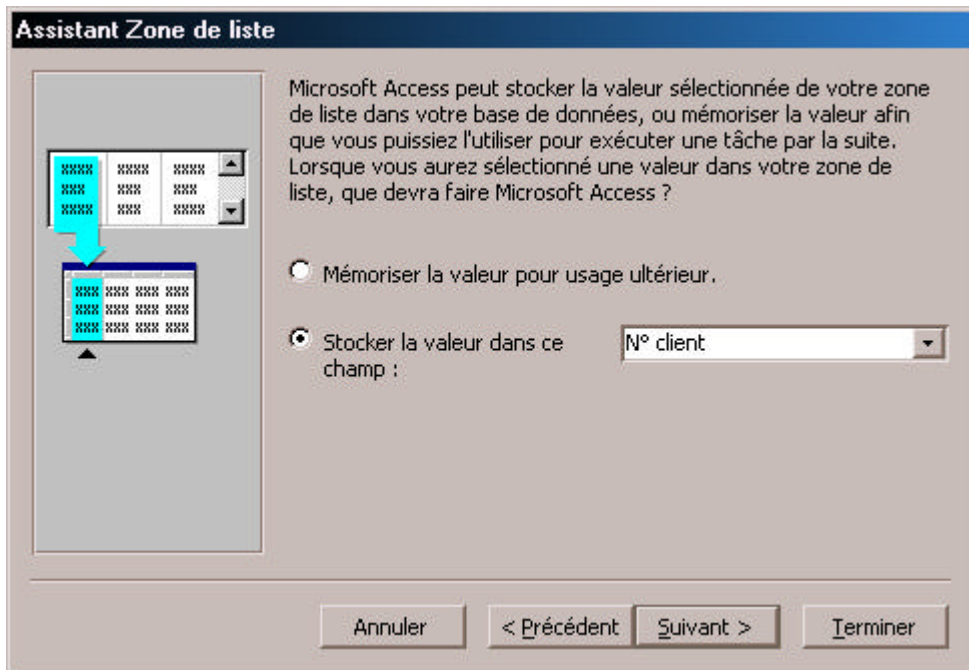
Annuler < Précédent Suivant > Terminer

Access nous affiche à quoi va ressembler notre zone de liste.

Notez que l'option « Colonne clé cachée » est cochée, qu'est-ce que cela signifie ?

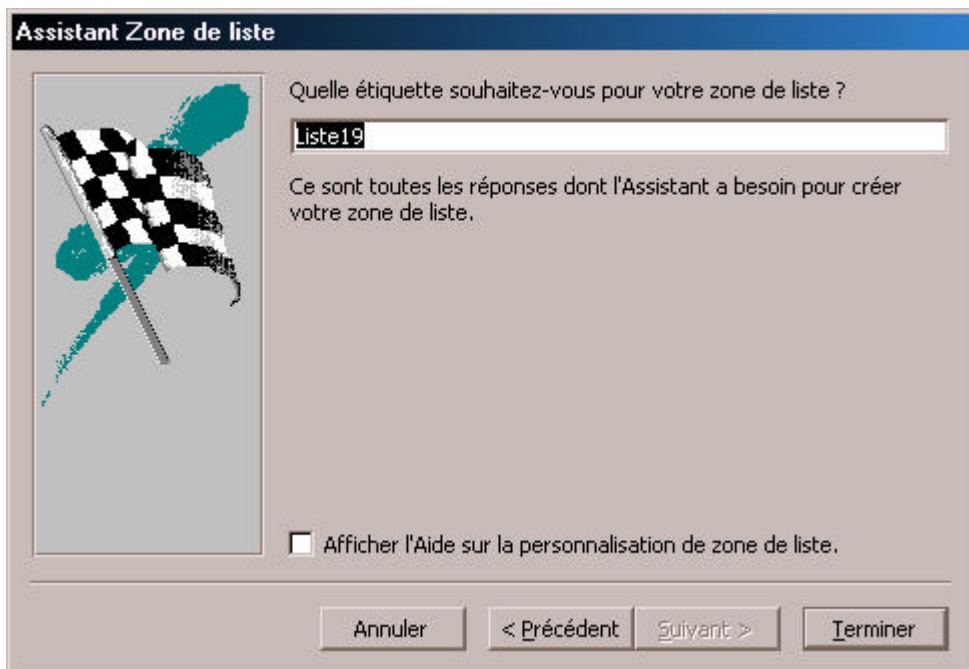
Nous sommes d'accord que le but de l'opération est de choisir parmi une liste un client et d'affecter son numéro au champ « n° client » de la table « Commande » afin d'éviter d'avoir à taper ce numéro. Or, nous avons choisi uniquement les champs Nom & Ville de la table « Client », et pas le numéro. Access a remarqué que le numéro du client est la clef de la table « Client », il en déduit que c'est donc le seul moyen d'identifier de façon unique un client, il va donc ajouter une colonne à notre liste, cette colonne sera cachée, et c'est le contenu de cette colonne cachée qui sera affectée au champ « n° client » de la table « Commande ».

7.



On va indiquer à Access dans quel champ de la table liée au formulaire on veut stocker cette valeur (ici n° client)

8.



On nomme la zone de liste, et c'est terminé



The screenshot shows a Microsoft Access form window titled "Commande". The form contains several fields and controls:

- N° Commande**: A text box containing the number "1".
- Date de la commande**: A date picker showing "12/01/98".
- Commande réglée**: A checked checkbox.
- Cadre10**: A group box containing three radio buttons labeled "Dupont", "Durant", and "Martin". The "Dupont" radio button is selected.
- Client**: A list control (Zone de liste modifiable) displaying a list of clients. The list is organized into two columns: names and cities. The selected item is "Albert" with the city "Saint-Quentin".

Client	City
Alban	Paris
Albert	Saint-Quentin
Béranger	Lille
Boudon	Paris
Chauvin	Saint-Quentin

At the bottom of the form, there is a navigation bar with the text "Enr : 1 sur 3" and navigation icons.

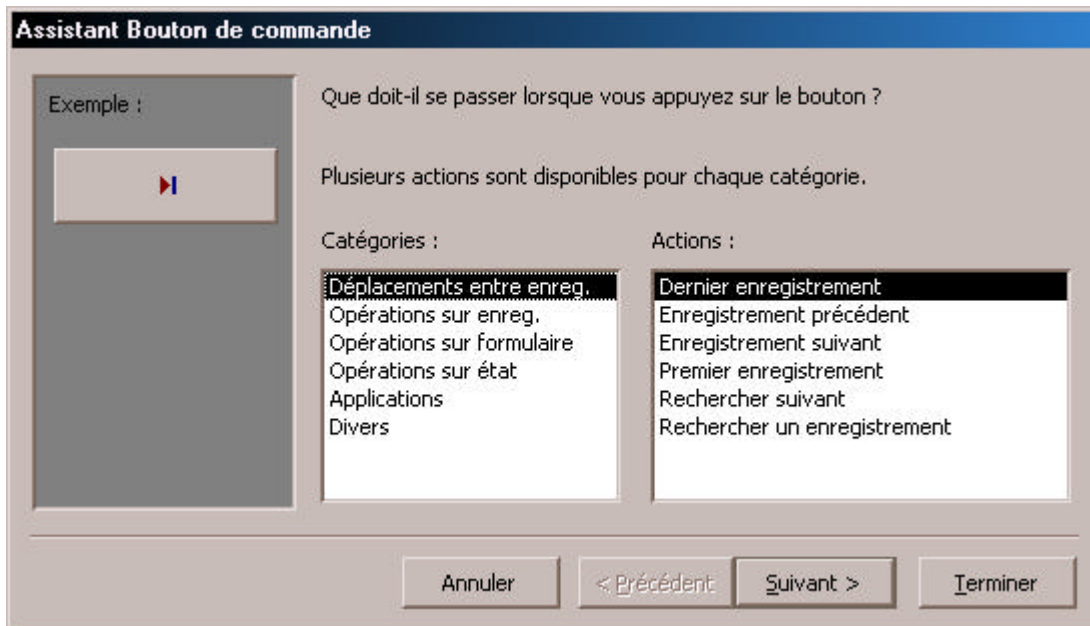
Vous voyez qu'au lieu de saisir un numéro, nous avons une liste dans laquelle nous pouvons choisir un client, le numéro de ce client, présent dans la colonne cachée de la zone de liste, sera affecté au champ choisi plus haut (n° client). Magique non ?

Le contrôle « Zone de liste modifiable » agit exactement de la même façon, il est différent parce que c'est un menu déroulant (il faut cliquer sur une petite flèche pour le faire apparaître) au lieu d'être fixe. Essayez le...

2.6 Le contrôle « Bouton de commande »

Ce contrôle va permettre d'exécuter simplement en cliquant dessus n'importe quelle action d'Access. Toutes les actions possibles d'Access, normalement accessibles par les menus déroulant ou par la barre d'icônes peuvent être reproduites par l'intermédiaire de ce contrôle. Pour comprendre son fonctionnement, nous allons créer un nouveau formulaire vide dans lequel nous allons simplement poser un contrôle « Bouton de commande ».

Dès que le contrôle est posé sur le formulaire, l'assistant suivant apparaît :



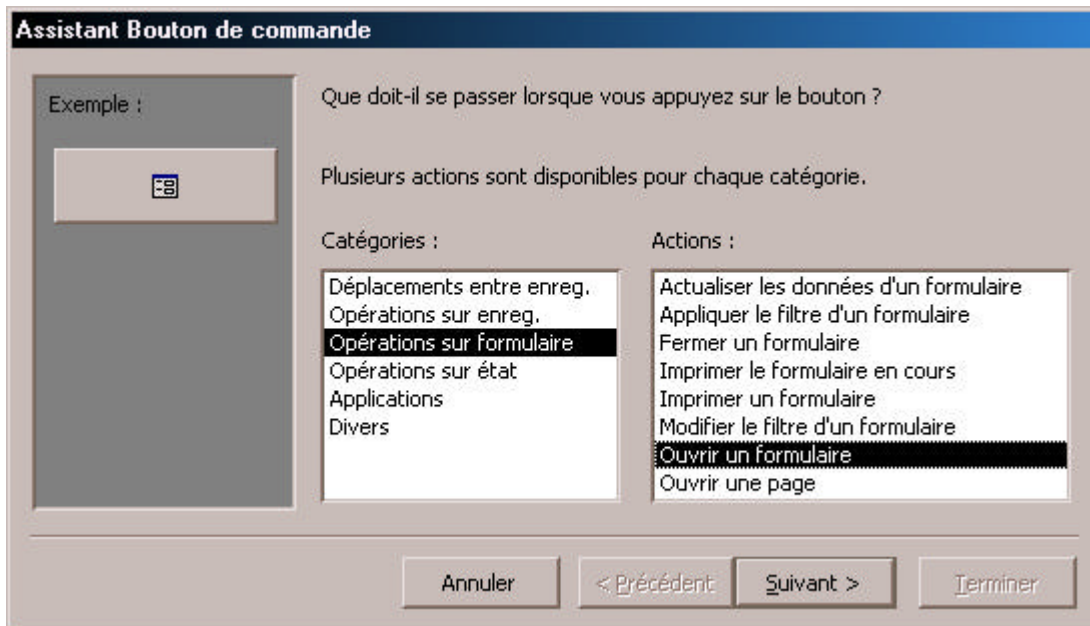
Il y a deux colonnes dans cet assistant :

Le menu **Catégories** affiche les principales fonctions d'Access :

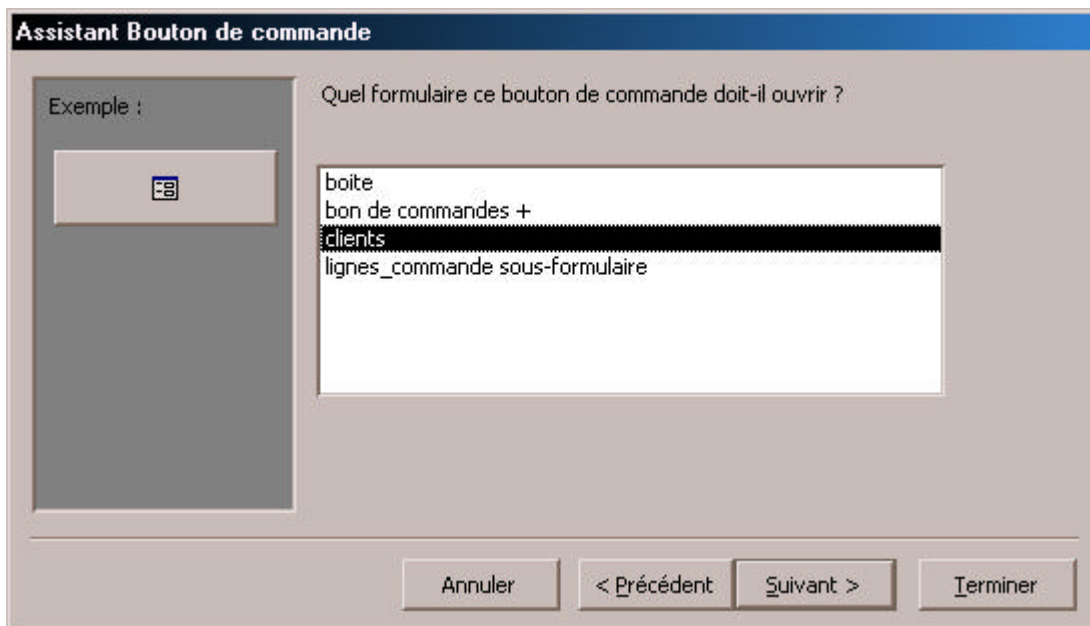
- ?? Déplacement entre enregistrements : Pour se positionner sur un enregistrement précis de la table.
- ?? Opération sur enregistrement : Ajouter, Supprimer, Copier, Imprimer un enregistrement
- ?? Opération sur formulaire : Ouvrir un formulaire, le fermer, appliquer un filtre dessus, l'imprimer
- ?? Opérations sur Etat : Imprimer, Sauver
- ?? Applications : Exécuter une autre application comme Word ou Excel
- ?? Divers : Exécuter une macro, Imprimer une table complète, ...

Comme vous pouvez le voir, toutes les actions possibles d'Access sont réunies ici, il suffit d'en choisir une et le fait de cliquer sur le contrôle exécutera cette action. Nous allons prendre pour exemple l'ouverture du formulaire « Clients ».

Nous choisissons donc la Catégorie « Opération sur Formulaire » puis l'action « Ouvrir un formulaire »



Access nous demande alors quel formulaire nous voulons ouvrir.



(ne vous préoccupez pas des noms des formulaires affichés ici, ils ne correspondent pas à nos exemples)

On choisit « Clients »

Assistant Bouton de commande

Exemple :

Souhaitez-vous que le bouton trouve des informations spécifiques à afficher dans le formulaire ?

Par exemple, le bouton peut ouvrir un formulaire et afficher les données pour un employé ou client spécifique.

Ouvrir le formulaire et trouver des informations spécifiques à afficher.

Ouvrir le formulaire et afficher tous les enregistrements.

Annuler < Précédent Suivant > Terminer

Access nous propose deux choix, nous verrons le premier plus tard, nous lui demandons d'afficher tous les enregistrements dans le formulaire

Assistant Bouton de commande

Exemple :

Souhaitez-vous du texte ou une image sur le bouton ?

Si vous choisissez du texte, vous pouvez taper le texte à afficher. Si vous choisissez une image, vous pouvez cliquer sur Parcourir pour trouver une image à afficher.

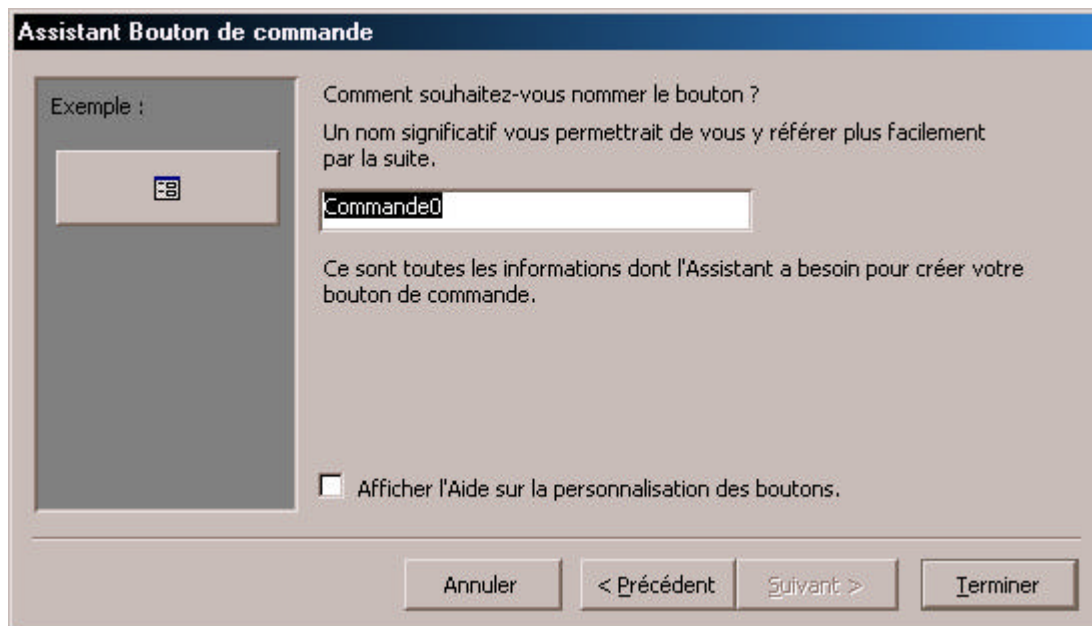
Texte : Ouvrir formulaire

Image : Formulaire MS Access Parcourir...

Afficher toutes les images

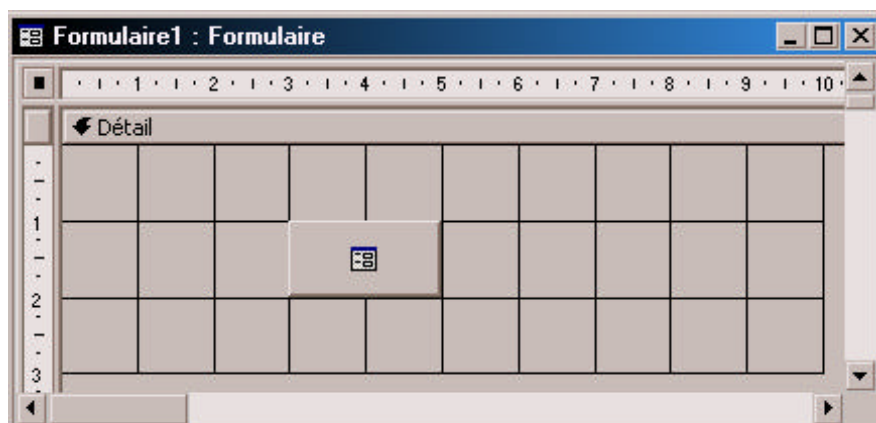
Annuler < Précédent Suivant > Terminer

On peut ici choisir ce qui va être affiché dans le contrôle, par défaut Access propose un icône, on peut afficher à la place un texte ou choisir un autre icône.



On nomme le contrôle et c'est terminé.

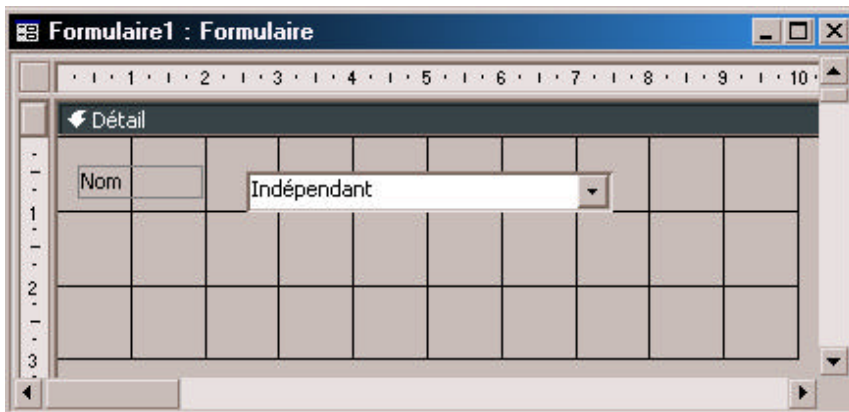
Notre formulaire ressemble à ça



Quand on cliquera sur le bouton de commande créé, le formulaire « Clients » s'ouvrira et affichera tous les clients de la table « Clients ».

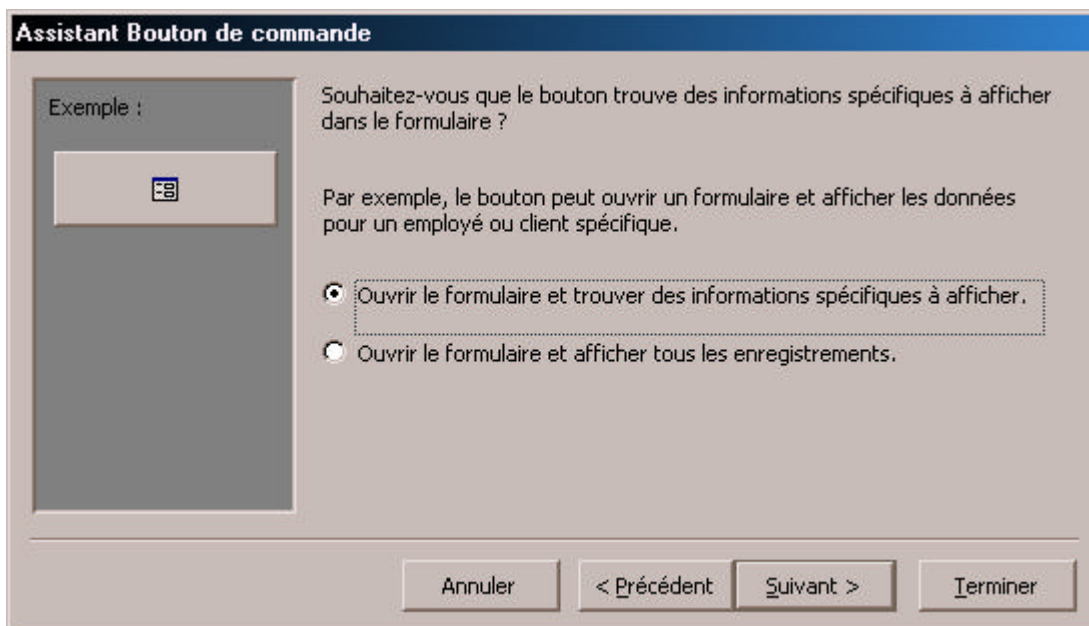
Nous pouvons améliorer ce formulaire grâce à quelques contrôles et en faire quelque chose de plus intéressant : Nous allons afficher tous les clients dans un contrôle « Zone de Liste », et ajouter sur le formulaire un bouton de commande comme celui que nous venons de créer avec cette fois, un petit plus : le formulaire qui va s'afficher quand on cliquera dessus n'affichera plus la totalité des clients mais seulement celui que nous aurons choisi dans la zone de liste.

Pour cela, on crée un nouveau formulaire vide (directement en mode création) et, comme plus haut, on crée une zone de liste dedans :



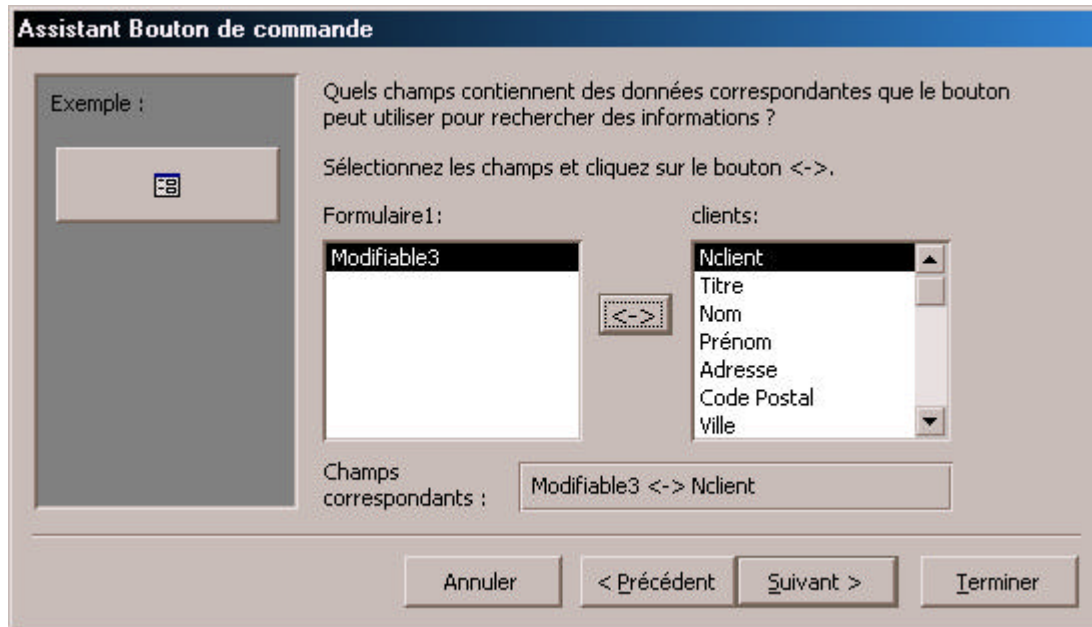
Notez cette fois-ci que, comme le formulaire a été créé à partir d'aucune table, Access ne demande pas à quel champ de la table nous voulons rattacher ce contrôle. Il est donc indépendant : il va se contenter d'afficher le numéro, le nom et le prénom des clients : le fait d'en choisir un dans cette liste ne modifiera aucune table.

Nous posons ensuite un contrôle « Bouton de Commande » sur le formulaire. Nous répétons les étapes précédentes, mais cette fois nous nous arrêtons à la question suivante :



Nous avons, précédemment, affiché tous les enregistrements de la table « Clients » dans le formulaire « Clients ». Nous voulons maintenant, afficher dans ce formulaire que le seul client correspondant à celui que nous avons choisi dans la liste.

La fenêtre suivante s'affiche :



A gauche nous avons les contrôles se trouvant sur notre formulaire : il n'y en a qu'un, il s'agit d'une zone de liste modifiable qui s'appelle « Modifiable3 » (le nom choisi par Access lorsqu'on a créé le contrôle. Il peut être différent).

A droite se trouvent tous les champs de la table associée au formulaire que nous voulons ouvrir.

Suivez bien le mouvement :

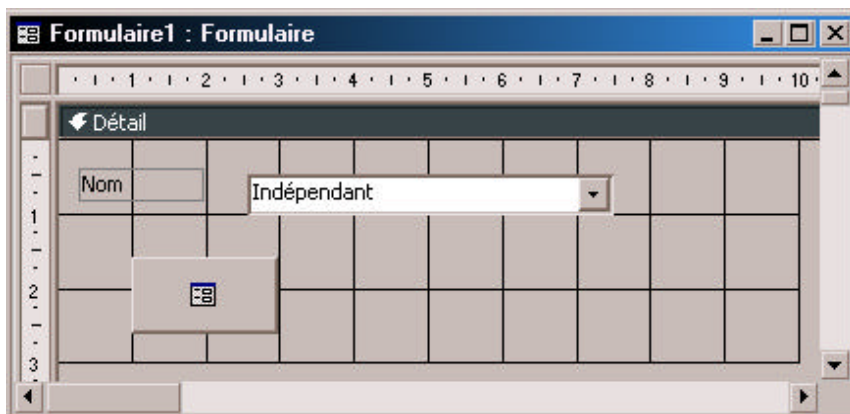
Lorsque nous avons créé notre contrôle « Zone de Liste » contenant la liste des clients, nous avons choisi de faire apparaître dans cette liste le numéro du client, son nom et son prénom. Access ayant noté que le numéro est la clef de la table « Clients », il ne l'affichera pas (voir page 19). Il ne l'affichera pas mais il sera toujours présent dans la liste mais il sera caché.

Bon, quand on choisit une des lignes du menu de la zone de liste, Access fait l'opération suivante : il « regarde » sur quelle ligne on a cliqué, il trouve une ligne « numéro, de client + nom + prénom », le numéro étant caché. Il ne s'occupe alors que de la première colonne (en l'occurrence la colonne cachée) : pour Access maintenant, le fait d'avoir cliqué sur cette ligne correspond au fait d'avoir cliqué sur le numéro seulement. Si vous n'aviez pas mis la colonne numéro de client, il aurait pris la première colonne (nom) et le fait de cliquer sur toute la ligne aurait correspondu au fait de cliquer sur le nom. OK ? Si vous n'aviez pas compris la page 19, j'espère que maintenant c'est plus clair.

Bien, maintenant, il faut dire à Access comment il va faire le lien entre ce qu'on a cliqué dans la liste et la table «Client », on choisit donc à gauche notre contrôle (on a pas trop le choix) et à droite le champ « nclient » (numéro de client) et on clique sur « <-> » .

Pourquoi ? comme on vient de le voir, le fait de cliquer sur un élément de la liste, correspond au fait de cliquer sur le numéro, on va lui dire avec cette opération : « Affiche dans le formulaire les enregistrements dont le champ « nclient » est égal à ce qu'on vient de cliquer dans la liste ».

On choisit ensuite l'icône et on doit se retrouver avec quelque chose comme ça :



Quand on va ouvrir le formulaire, on va voir ceci :



La liste des clients s'affiche (le nom & le prénom, le numéro étant caché)

Une fois qu'on a choisi un, on clique sur le bouton de commande et comme par magie, le formulaire client s'ouvre avec seulement les informations concernant le client choisi dans la liste.

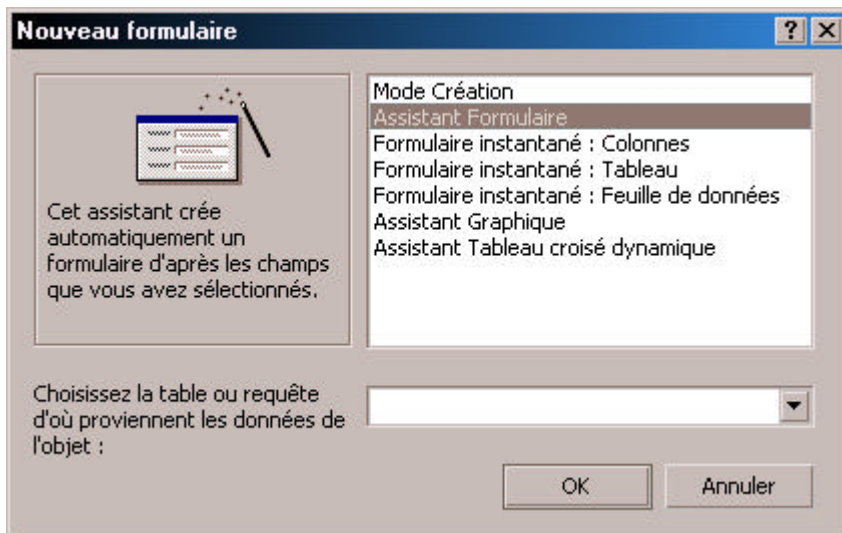
Ceci n'est qu'un petit exemple de ce que l'on peut réaliser avec le contrôle « bouton de commande », vous pouvez essayer d'autres commandes.

2.7 Le contrôles « Sous-Formulaire »

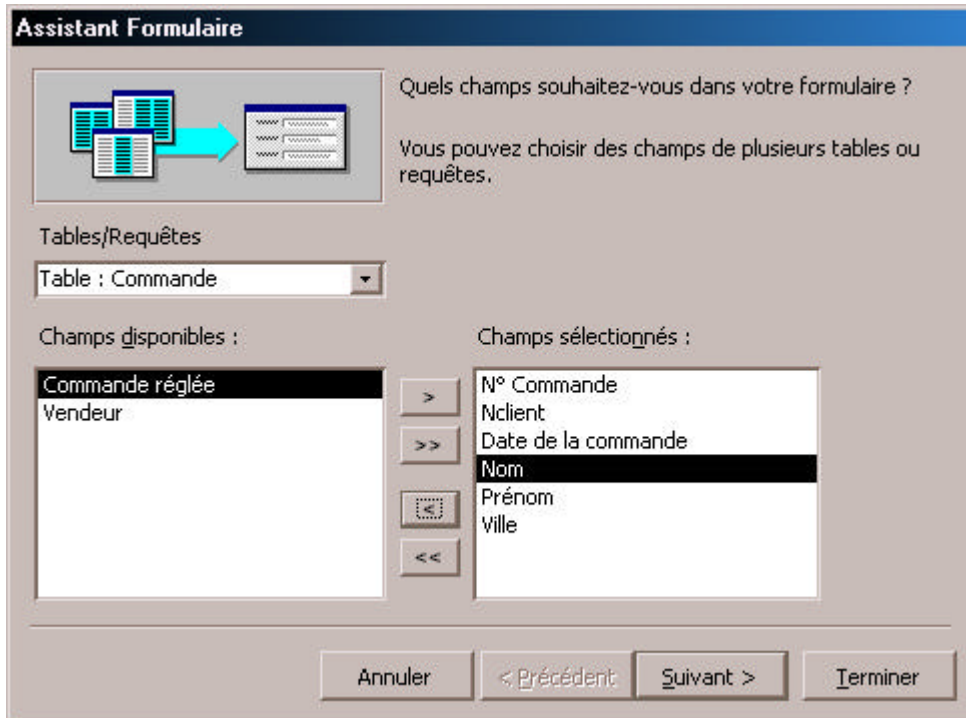
Grâce à ce contrôle, nous allons pouvoir afficher des formulaires imbriqués dans d'autres, par exemple pour afficher une liste d'enregistrement correspondant à un seul enregistrement : pour une commande donnée, on pourra, dans le même formulaire, afficher toutes les lignes de cette commande.

Le but de la manœuvre va être de réaliser un formulaire complexe que l'on va nommer « facture ». Il contiendra toutes les informations relatives à une commande : en en-tête le numéro de la commande, sa date, les informations sur le client, et en dessous le détail de cette commande, avec en bas le montant total de la commande.

Pour cela, on crée le formulaire «facture ». Ce formulaire va être l'en-tête de notre formulaire final : il va contenir les informations sur la commande et sur le client qui l'a passé. On va donc aller chercher des informations dans deux tables : « Clients » et « Commandes ».



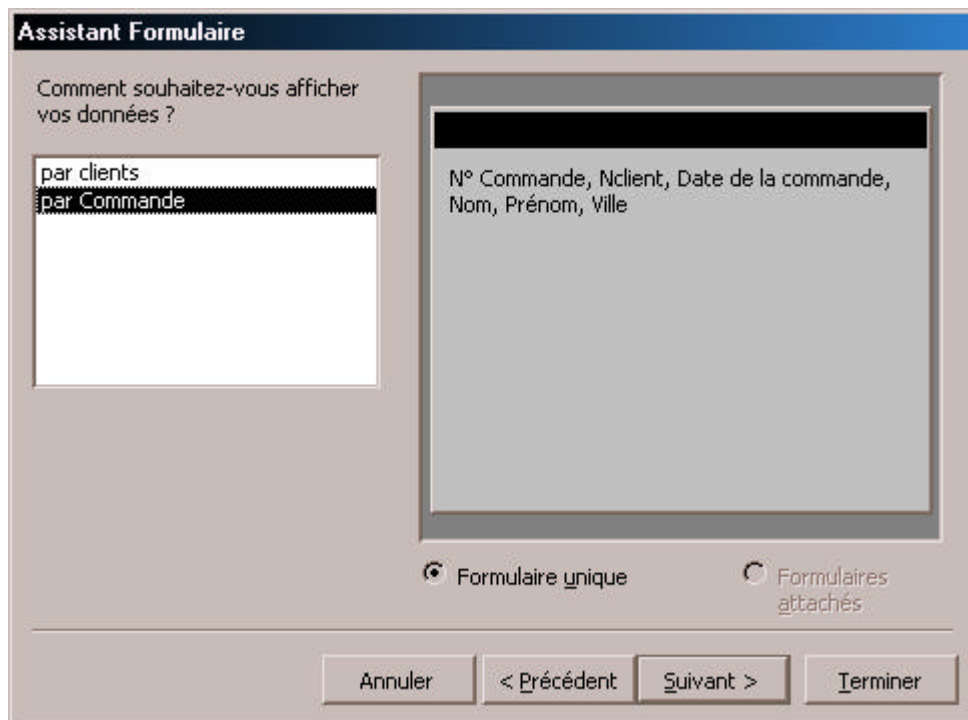
Comme on l'a vu, le formulaire va être généré à partir de deux tables, on ne choisit donc rien dans le menu « Choisissez la table ».



C'est ici qu'on va choisir les tables : En haut à gauche se trouvent les tables de la base, en dessous les champs de la table choisie plus haut et à gauche les champs que nous allons utiliser dans notre formulaire.

Pour notre en-tête, nous avons besoin des informations suivantes : n° de la commande, n° du client qui a passé cette commande, la date de la commande, ces informations provenant de la table «Commande» et le nom, le prénom et la ville du client qui a passé cette commande, ces informations provenant de la table « Clients ».

A partir de maintenant, nous supposons que les relations ont été bien réalisées entre les tables, c'est une condition nécessaire au bon déroulement de la suite des opérations. Si les relations ne sont pas faites ou mal faites, Access va « tenter » de relier comme il peut les tables entre elles, et c'est la source de problèmes...



Les relations étant correctement faites, Access, dans une crise d'intelligence remarque qu'à partir des champs choisis précédemment, on peut faire deux choses : soit afficher les données par clients, c'est à dire que, avec les champs choisis, on peut afficher un client et en dessous afficher toutes les commandes qu'il a passé. Soit afficher les données par commande, c'est à dire que pour une commande choisie, on va afficher les informations concernant le client qui a fait cette commande. A droite s'affiche la façon dont les données seront affichées en fonction du mode qu'on aura choisi.

J'espère que votre esprit affûté aura remarqué que la deuxième proposition est la bonne, nous la choisissons donc.

Les étapes suivantes sont classiques : choix de la disposition, de la décoration et du nom, on arrive à la fin à quelque chose comme ça :

En-tête de formulaire	
Détail	
N° Commande	N° Commande
N° Client	Nclient
Date de la commande	Date de la co
Nom	Nom
Prénom	Prénom
Ville	Ville
Pied de formulaire	

Bon, étape 1 terminée. Nous allons maintenant créer un nouveau formulaire, ce formulaire va être une sorte de « super formulaire ligne-commandes ». Dans notre formulaire ligne commande, nous avons seulement le numéro de la commande, le numéro du produit et la quantité.

Nous allons créer un formulaire « super ligne-commande » qui va contenir sur chaque ligne : le numéro de la commande, le numéro du produit, sa quantité (tout cela provenant de la table « lignes-commande »), mais en plus le nom du produit, son prix unitaire, son taux de TVA (tout cela provenant de la table « produits ») et pour corser l'affaire, nous ajouterons le calcul du montant TTC de la ligne, et hop !

Comme pour notre précédent formulaire, nous allons utiliser deux tables :

Assistant Formulaire

Quels champs souhaitez-vous dans votre formulaire ?
Vous pouvez choisir des champs de plusieurs tables ou requêtes.

Tables/Requêtes
Table : produits

Champs disponibles :
num_prod

Champs sélectionnés :
num_cmde
num_prod
quantite
libelle
prix_unitaire
tva

Annuler < Précédent Suivant > Terminer

Nous choisissons les champs numéro de commande, numéro de produit et quantité de la table « ligne-commande » et les champs libellé, prix unitaire et tva de la table « Produits ».

Assistant Formulaire

Comment souhaitez-vous afficher vos données ?

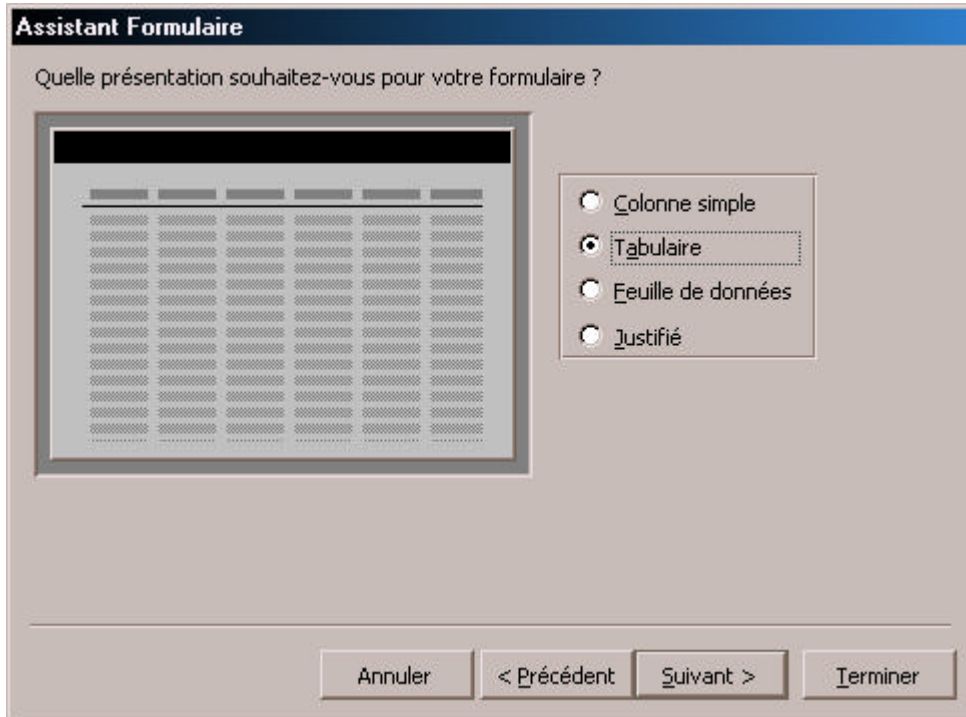
par produits
par lignes_commande

num_cmde, num_prod, quantite, libelle,
prix_unitaire, tva

Formulaire unique Formulaires attachés

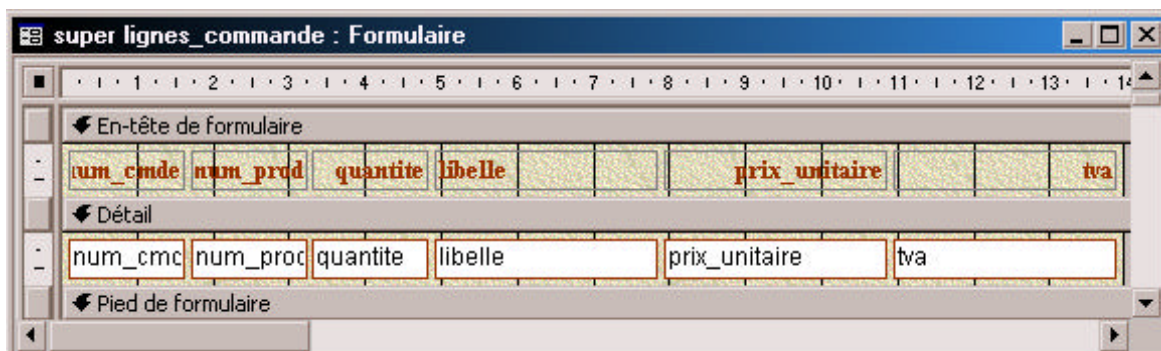
Annuler < Précédent Suivant > Terminer

Une fois de plus, Access fait preuve d'intelligence ! Il se rend compte qu'avec les champs que nous avons choisis, nous pouvons afficher les données sous deux formes : soit par produit, dans ce cas il va afficher toutes les lignes de commandes où ce produit est présent, pourquoi pas ? mais ça ne nous intéresse pas spécialement, soit par ligne de commande, et il va dans ce cas, afficher les informations sur le produit présent dans cette ligne de commande. Voilà qui est beaucoup plus intéressant.



Cette fois-ci, nous allons afficher le formulaire sous forme « Tabulaire ». Comme ce formulaire va contenir la liste des lignes d'une commande, c'est la meilleure façon d'afficher une liste.

A la fin, on obtient un formulaire qui a cette forme :



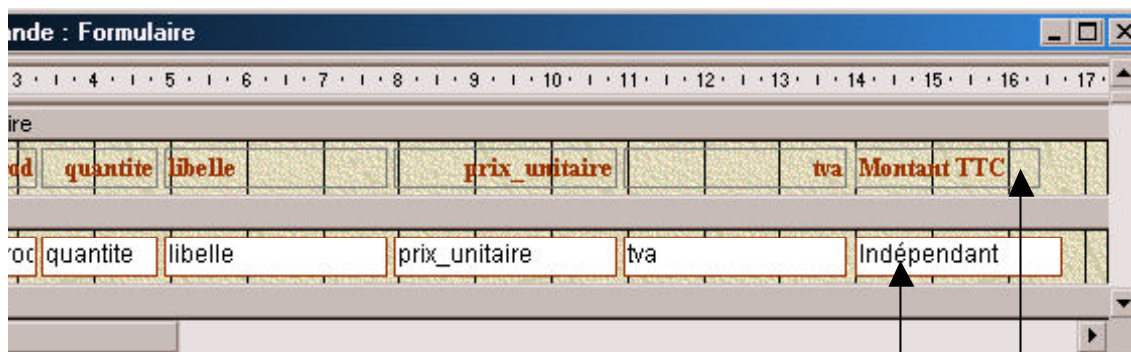
Dans l'en-tête se trouvent les titres des colonnes, et dans le détail, les lignes du formulaire. Une fois ouvert, il aura cet aspect :



num_cmde	num_prod	quantite	libelle	prix_unitaire	tva
2	1	14	prod1	110.00 F	20.6
3	1	10	prod1	110.00 F	20.6
2	2	11	prod2	220.00 F	20.6
1	2	12	prod2	220.00 F	20.6
1	2	13	prod2	220.00 F	20.6
3	2	1	prod2	220.00 F	20.6

Nous sommes bien d'accord, ce formulaire affiche le détail de toutes les commandes. Comme on peut le voir dans la 1^{ère} colonne, il concerne plusieurs commandes.

Bien, maintenant, nous allons ajouter un contrôle dans ce formulaire pour calculer le montant TTC de chaque ligne :



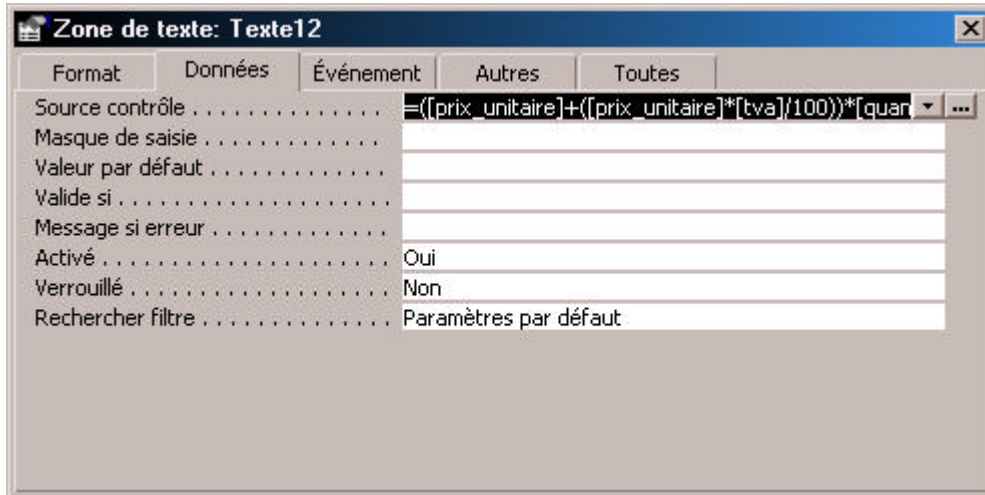
num_cmde	quantite	libelle	prix_unitaire	tva	Montant TTC
2	14	prod1	110.00 F	20.6	
3	10	prod1	110.00 F	20.6	
2	11	prod2	220.00 F	20.6	
1	12	prod2	220.00 F	20.6	
1	13	prod2	220.00 F	20.6	
3	1	prod2	220.00 F	20.6	

Zone de texte

Intitulé

Nous avons ajouté deux contrôles : un contrôle « Intitulé » pour afficher l'en-tête de la colonne, et un contrôle « Zone de Texte » dans lequel on va calculer le montant TTC.

Pour calculer le montant TTC, on affiche les propriétés du contrôle et dans la propriété « Source » de l'onglet « Données », on tape la formule de calcul :



La formule de calcul est :

$([\text{prix unitaire}] + ([\text{prix unitaire}] * [\text{tva}] / 100)) * [\text{quantite}]$

Si on affiche maintenant le formulaire, on obtient :

num_cmde	num_prod	quantite	libelle	prix_unitaire	tva	Montant TTC
2	1	14	prod1	110.00 F	20.6	1857.24
3	1	10	prod1	110.00 F	20.6	1326.6
2	2	11	prod2	220.00 F	20.6	2918.52
1	2	12	prod2	220.00 F	20.6	3183.84

Enr : 1 sur 9

Ca marche !

Plus fort, maintenant, nous allons afficher le montant total commandé. Nous sommes toujours d'accord, le montant total va être le montant total pour toutes les commandes, notre formulaire pour l'instant ne fait pas de distinction entre les commandes.

Vous avez pu remarquer que le formulaire a en plus de l'en-tête et du détail, une partie «Pied de formulaire». Cette partie ne sera affichée que en bas du formulaire, une fois toutes les lignes affichées. Comme précédemment, nous mettons dans cette partie du formulaire un contrôle « Zone de Texte » dans lequel nous allons faire le calcul de la somme.

Que contient la formule ? La même que tout à l'heure, mais nous avons ajouté le mot-clef « somme » avant : Access va calculer la somme de toutes les lignes ici.

num_cmde	num_prod	quantite	libelle	prix_unitaire	tva	Montant TTC
2	1	14	prod1	110.00 F	20.6	1857.24
3	1	10	prod1	110.00 F	20.6	1326.6
2	2	11	prod2	220.00 F	20.6	2918.52
1	2	12	prod2	220.00 F	20.6	3183.84
Montant Total:						36674.88

Et voilà le travail : en bas du formulaire apparaît la somme totale.

Arrivé ici, nous avons notre formulaire « facture » qui contient les informations générales sur la commande et le formulaire « super ligne commande » qui contient les détails de toutes les commandes. Il serait bien de pouvoir insérer ce formulaire dans le premier et de lui faire afficher seulement les lignes commandes correspondant à la commande choisie.

Pour cela, on va rouvrir notre formulaire « facture » et poser dessus un contrôle « Sous-Formulaire ». L'assistant suivant s'affiche alors :

Assistant Sous-formulaire

Vous pouvez utiliser un formulaire existant pour créer votre sous-formulaire ou sous-état, ou en créer un nouveau à l'aide de tables et/ou de requêtes.

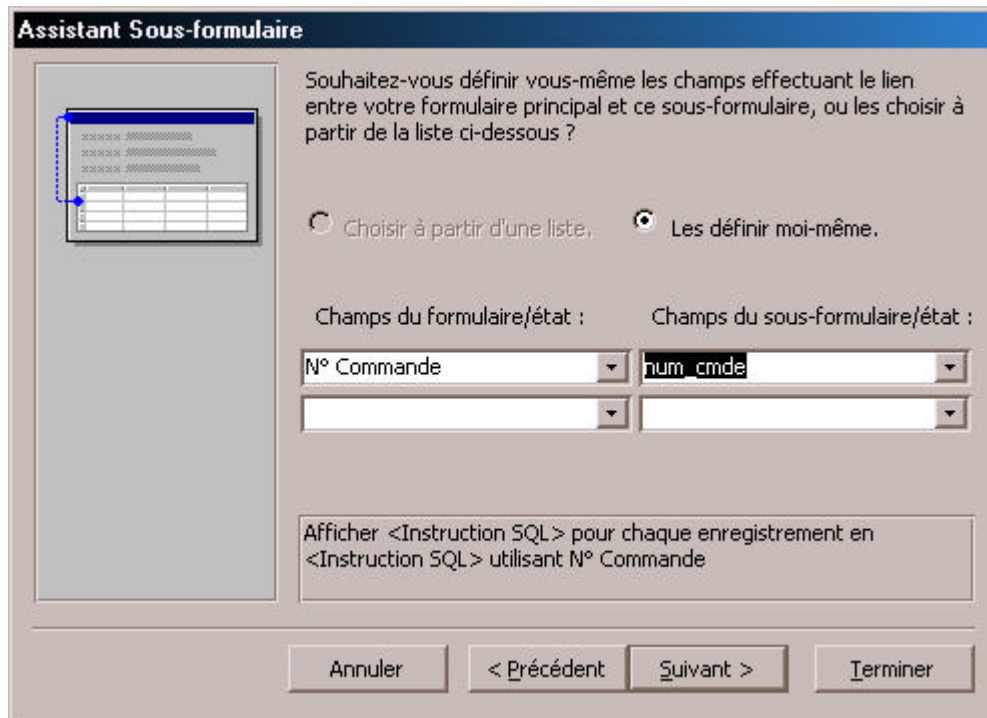
Quelles données souhaitez-vous utiliser pour votre sous-formulaire ou sous-état ?

Utiliser les tables et les requêtes existantes
 Utiliser un formulaire existant

boite
 bon de commandes + clients
 Formulaire1
 lignes_commande sous-formulaire
 super lignes commande

Annuler < Précédent Suivant > Terminer

Access nous demande à partir de quoi il va afficher ce sous-formulaire ? d'une table ou d'un formulaire existant déjà ? Nous choisissons la deuxième solution et nous choisissons notre formulaire « super lignes commande ».



C'est ici que tout se fait. Il va falloir indiquer à Access quel est le rapport entre le formulaire et le sous formulaire qu'il contient. Si on se contentait d'insérer le sous formulaire tel quel dans le formulaire, il continuerait à afficher tout le temps la totalité des lignes commandes quelle que soit la commande affichée dans le formulaire principal.

Nous voulons qu'il affiche dans le sous-formulaire les lignes commande correspondant à la commande affichée dans le formulaire principal.

Quel est le rapport entre les deux ? Comment faire une relation entre les deux ? Simplement : le formulaire contient le numéro de la commande en cours, le sous-formulaire contient pour chaque ligne de commande affichée le numéro de la commande concernée. Il va donc falloir dire à Access que les lignes affichées dans le sous formulaire vont être celles dont le numéro de commande est celui qui est affiché dans le formulaire.

Pour cela, on indique dans la colonne de gauche quel est le champ qui contient le numéro de commande dans le formulaire principal et quel est le champ qui contient aussi un numéro de commande dans le sous-formulaire.

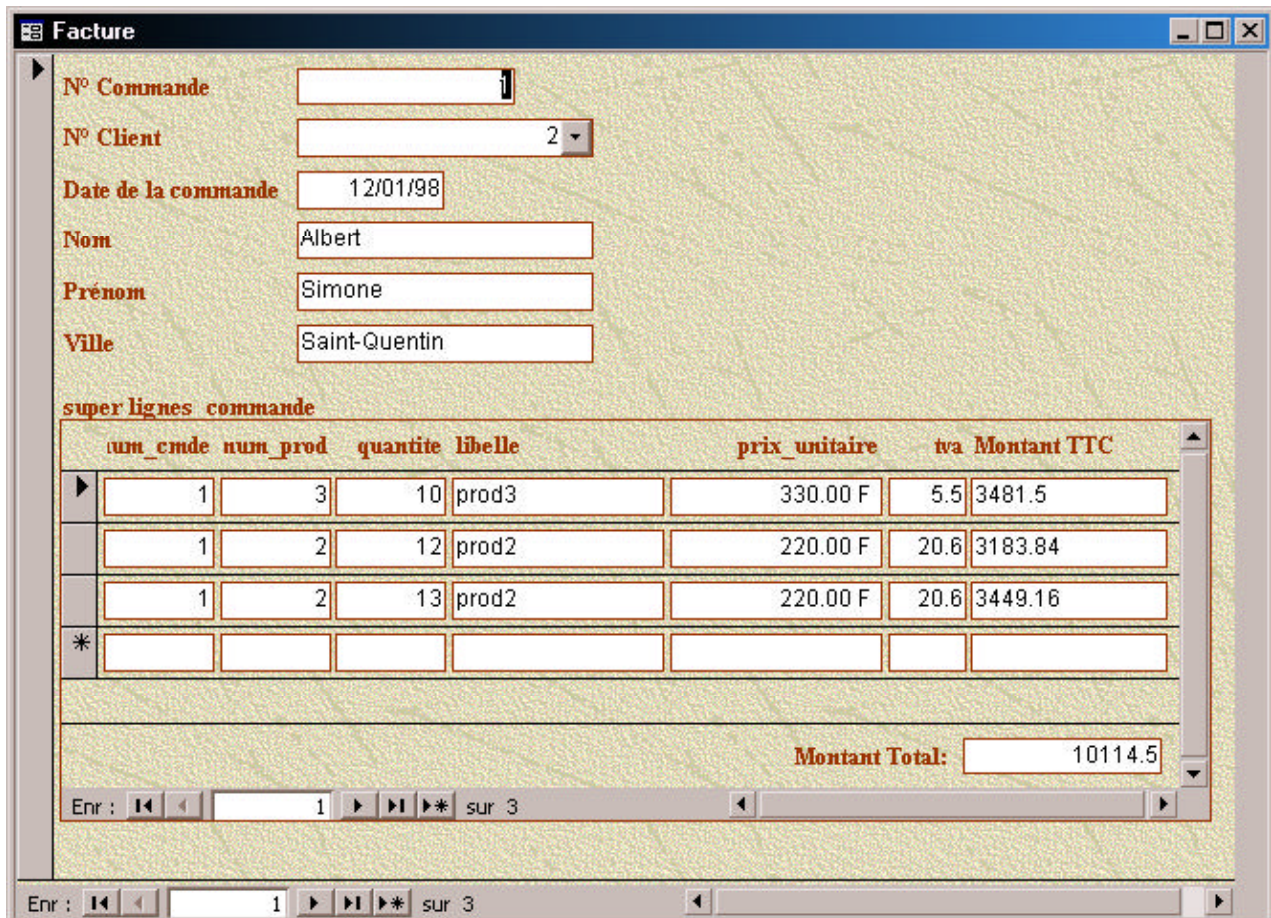
The screenshot shows an Access form window titled "Facture : Formulaire". The form is divided into sections: "En-tête de formulaire" (Form Header) and "Détail" (Detail). The "Détail" section contains a table with the following fields:

N° Commande	N° Commande
N° Client	Nclient
Date de la commande	Date de la co
Nom	Nom
Prénom	Prénom
Ville	Ville
super lignes commande	

Below the "super lignes commande" field, there is a large white rectangular area, which is a subform placeholder. The form also has a "Pied de formulaire" (Form Footer) section at the bottom.

A la place du sous-formulaire, il y a un rectangle blanc. Ce qui implique que toute modification dans ce sous-formulaire ne pourra se faire ici, il faudra la faire directement dans le formulaire « super ligne commandes ». Dans Access 2000, on peut directement modifier le sous formulaire à partir d'ici, il n'y a plus de rectangle blanc mais directement le formulaire.

Maintenant, lorsque vous ouvrez le formulaire principal :



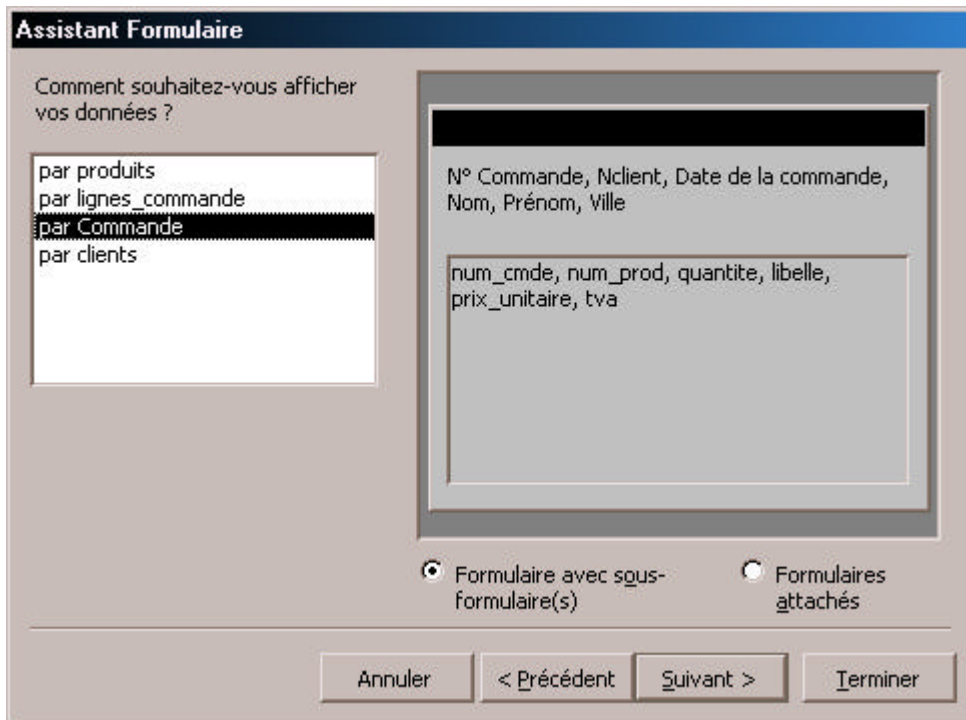
The screenshot shows an Access form titled 'Facture'. It contains several text boxes for customer information: 'N° Commande' (1), 'N° Client' (2), 'Date de la commande' (12/01/98), 'Nom' (Albert), 'Prénom' (Simone), and 'Ville' (Saint-Quentin). Below this is a table titled 'super lignes commande' with columns: 'num_cmde', 'num_prod', 'quantite', 'libelle', 'prix_unitaire', 'tva', and 'Montant TTC'. The table contains three rows of data and one empty row marked with an asterisk. At the bottom right, a 'Montant Total' field shows 10114.5. The form has a navigation bar at the bottom with 'Enr : 1 sur 3' and navigation buttons.

num_cmde	num_prod	quantite	libelle	prix_unitaire	tva	Montant TTC
1	3	10	prod3	330.00 F	5.5	3481.5
1	2	12	prod2	220.00 F	20.6	3183.84
1	2	13	prod2	220.00 F	20.6	3449.16
*						

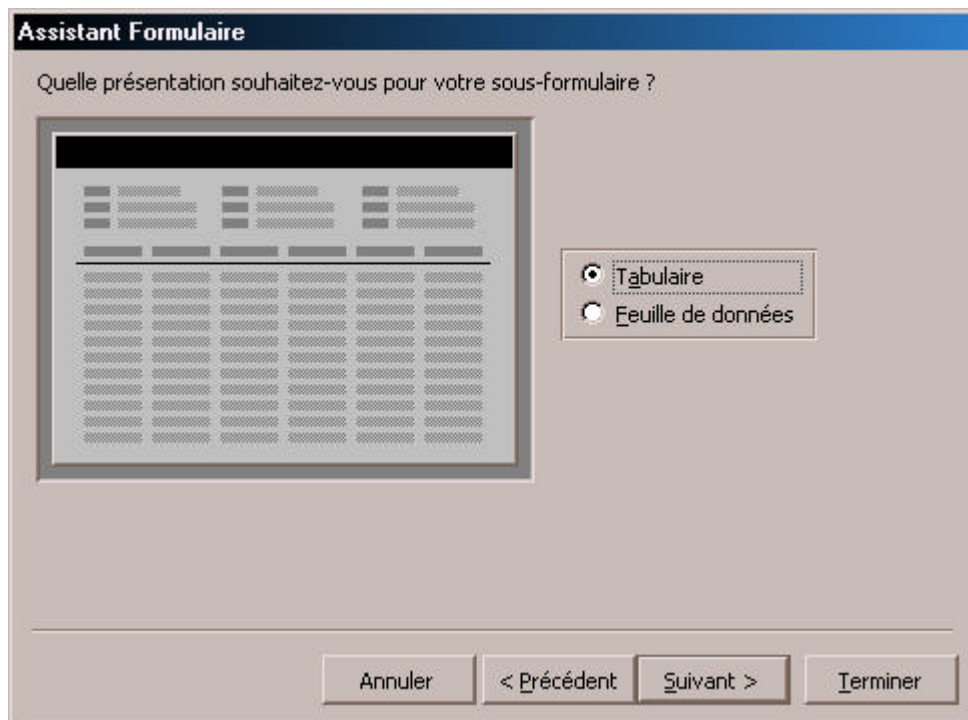
Vous avez, pour une commande donnée en haut, son contenu exact et son montant. Vous noterez que le numéro de commande est répété pour chaque ligne, vous pouvez, comme il est inutile maintenant puisque déjà présent dans l'en-tête, le supprimer du formulaire.

Voilà pour les sous-formulaires. Il existe une méthode plus simple pour faire ce que nous venons de faire. Il suffit pour cela, d'abord de bien relier les tables entre elles, ensuite lorsqu'on crée le premier formulaire de mettre toutes les champs de toutes les tables concernées dans le même formulaire : n° de commande, n° de client, date venant de la table « Commande », nom, prénom, adresse venant de la table « Client », n° de commande, n° de produit, quantité venant de la table « Lignes-commande » et libellé, prix unitaire et tva venant de la table « Produits ».

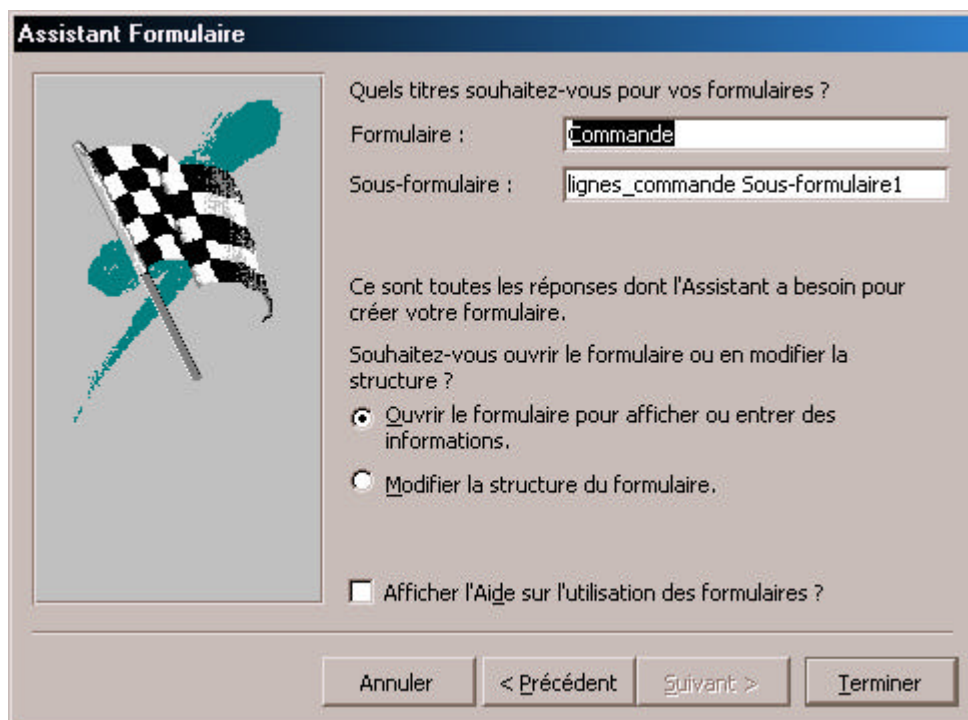
Access affiche alors :



Et là, ô magie ! Access a deviné tout seul ce que nous voulions faire (grâce quand même aux relations), et nous propose de faire tout seul ce que nous avons fait avant. Vous remarquez que l'exemple qu'il nous propose à droite ressemble étrangement à ce que nous avons fait «à la main ».



Il nous propose le format d'affichage



Ici on nomme le formulaire, vous pouvez noter qu'il a créé tout seul le sous formulaire et qu'il en profite pour nous demander son nom.

The screenshot shows an Access form window titled "Commande". The main form contains the following fields:

- N° Commande: 1
- Ville: Saint-Quentin
- N° Client: 2
- Date de la commande: 12/01/98
- Nom: Albert
- Prénom: Simone

The subform "lignes_commande" contains a table with the following data:

num_cmde	num_prod	quantite	libelle
1	3	10	prod3
1	2	12	prod2
1	2	13	prod2

Navigation controls are visible at the bottom of both the main form and the subform, showing "Enr : 1 sur 3".

Et voilà le travail ! En cinq secondes, il a fait ce qui nous a pris beaucoup de temps !

Il nous faudra simplement ajouter ensuite dans le sous formulaire le calcul des montant TTC et le montant total.

Comme quoi, si les relations sont bien faites, Access fait les trois quart du boulot tout seul.

Ici se termine l'histoire des formulaires. Il reste d'autres contrôles comme les boutons bascule, les images, les objets OLE, etc... Nous ne les utiliserons pas souvent.

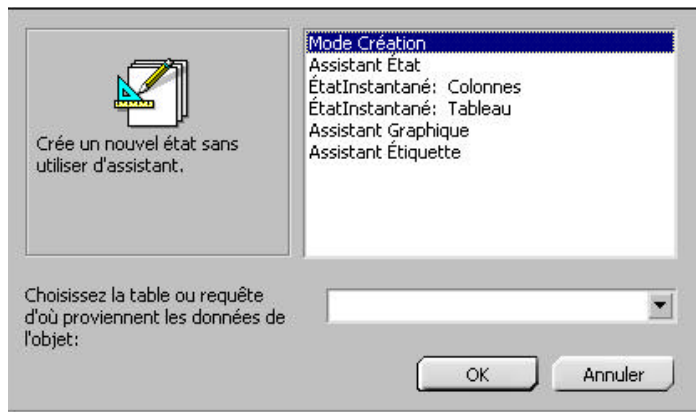
1. Création d'un état.....	2
2. Création d'un état Instantané Colonnes.....	3
3. Création d'un état Instantané Tableau.....	4
4. Création d'un état avec plusieurs tables.....	9
5. Modifier la structure d'un état.....	11
6. Insertion d'un sous-état.....	16

LES ETATS

Les états vont permettre l'impression d'enregistrements selon une présentation qui aura été définie préalablement. La création d'un état ressemble fortement à la création d'un formulaire.

1. Création d'un état

Pour créer un état, dans la fenêtre principale d'Access, cliquez sur l'onglet "Etats", puis sur "Nouveau" :

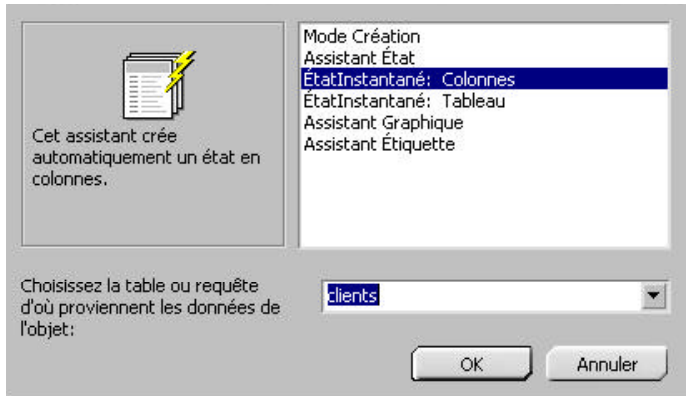


Il y a plusieurs méthodes pour créer un état :

- ?? **Mode création** : on crée tout de A à Z, on n'utilisera pas cette méthode
- ?? **Assistant Etat** : Access va nous guider pas à pas dans la création d'un état complexe
- ?? **Etat Instantané : Colonnes** : Access va générer automatiquement un état à partir du contenu d'une table, la présentation se fera comme dans un formulaire, sous forme de colonnes : à gauche on aura le nom du champ et à droite, son contenu.
- ?? **Etat Instantané : Tableau** : Access va générer automatiquement un état à partir du contenu d'une table mais sous forme de tableau, on aura en haut de la page le nom des champs et en dessous, le contenu pour chaque enregistrement, comme dans la présentation d'une table
- ?? **Assistant Graphique** : Pour créer un graphique, nous verrons cela plus tard
- ?? **Assistant Etiquette** : Pour créer automatiquement des étiquettes à partir du contenu d'une table, nous ne verrons pas cette option.

2. Création d'un état Instantané Colonnes

Nous allons créer un état rapidement à partir de la table Clients :



Notez qu'on choisi la table dans le menu du bas.

Access nous génère automatiquement l'état suivant (en mode prévisualisation)

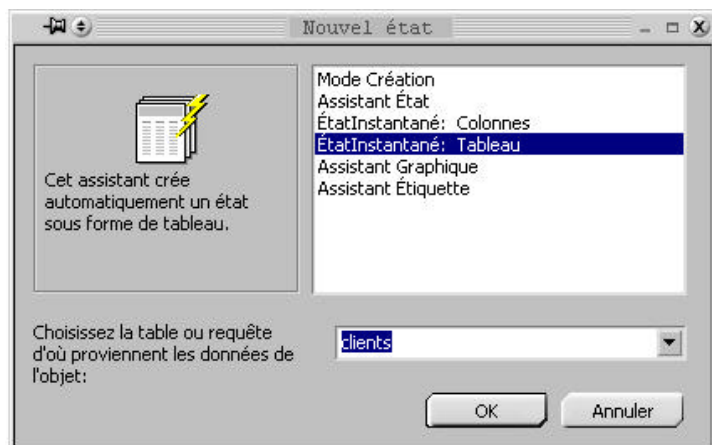
Titre	Monsieur
Nom	Albert
Prénom	Jacques
Adresse	Rue Barbe
Code Postal	75000
Ville	Paris
Observations	

Titre	Monsieur
Nom	Albert
Prénom	Sébastien
Adresse	Rue Maitre
Code Postal	02100
Ville	Saint-Quentin
Observations	

Titre	Mademoiselle
Nom	Hénriette
Prénom	Huguette
Adresse	Rue Tanc
Code Postal	59000

On a en haut le titre de la table, à gauche le nom du champ et à droite le contenu, chaque enregistrement est séparé par un trait.

3. Création d'un état Instantané Tableau



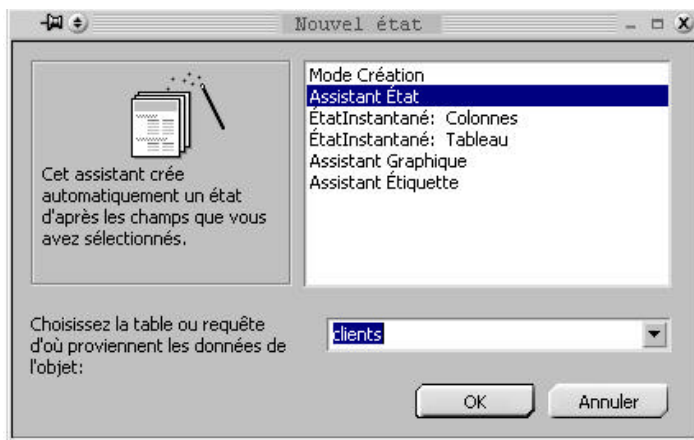
Et Access génère l'état suivant :

clients				
N°	Titre	Nom	Prénoms	Adresse
1	Monsieur	Albert	Jacques	rue Boire
2	Madame	Albert	Suzette	rue Miron
3	Mademoiselle	Bénigne	Huguette	rue Tanc
4	Monsieur	Boudon	Arthur	rue Tibon

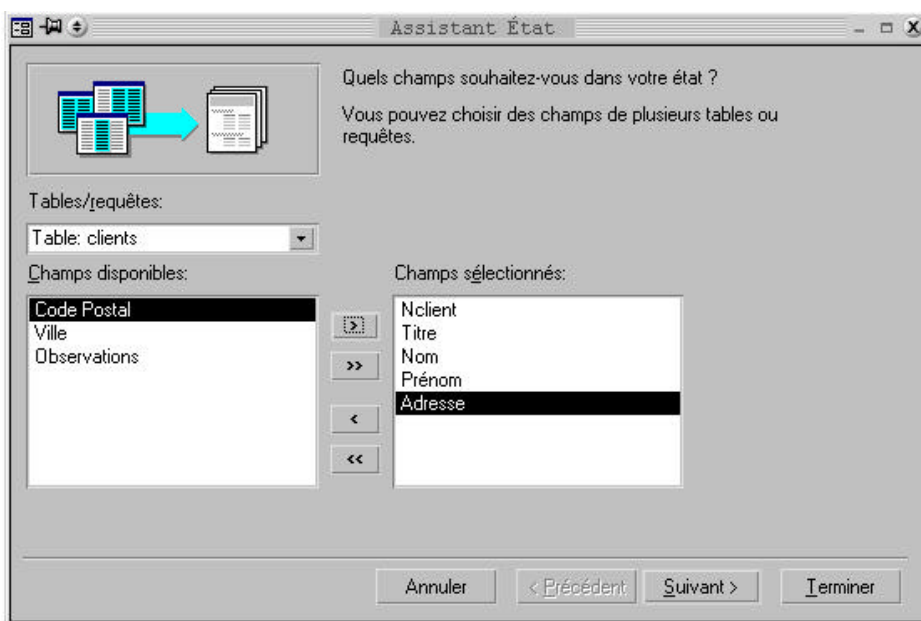
En haut, le nom de la table, suivi du nom des champs et sur chaque ligne, le contenu des champs pour chaque enregistrement.

1. Création d'un état à partir de l'assistant

Nous utiliserons désormais cette méthode pour créer un état, les deux méthodes précédentes étant trop restrictives.

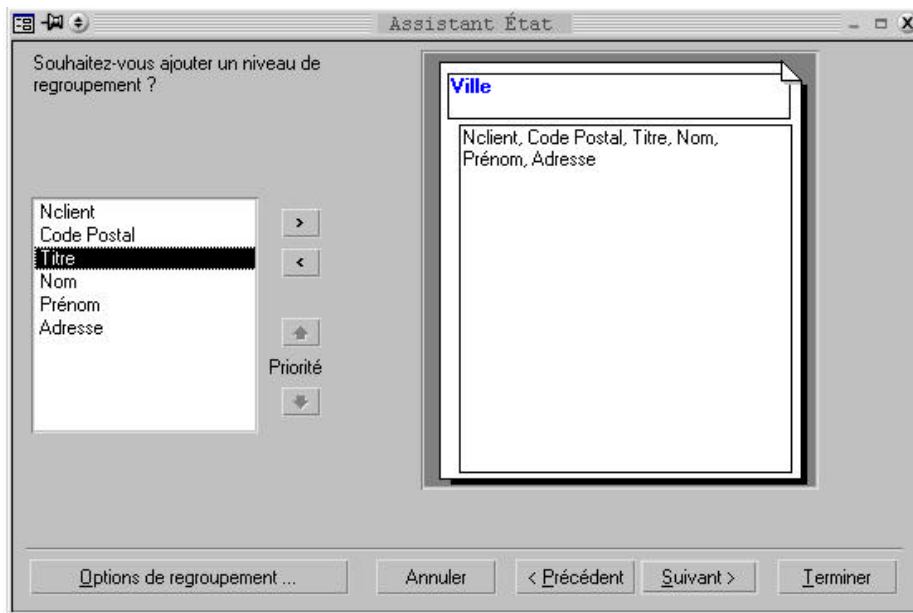


Comme précédemment, on choisit à partir de quelle table va être créé l'état, si on veut créer un état à partir de plusieurs tables, on ne met rien ici, on renseignera la provenance des données dans l'écran suivant :



Comme pour la création d'un formulaire, on choisit à gauche les champs que l'on veut voir apparaître dans l'état, notez que si vous voulez afficher des champs provenant de plusieurs tables, vous choisirez dans le menu "**Table/requêtes**" une nouvelle table, les champs correspondants s'afficheront en dessous. Comme dans les formulaires, pour ajouter un champ, on utilise l'icône ">", pour les ajouter tous, on utilise l'icône ">>" et dans l'autre sens pour les retirer, cliquez ensuite sur "Suivant >".

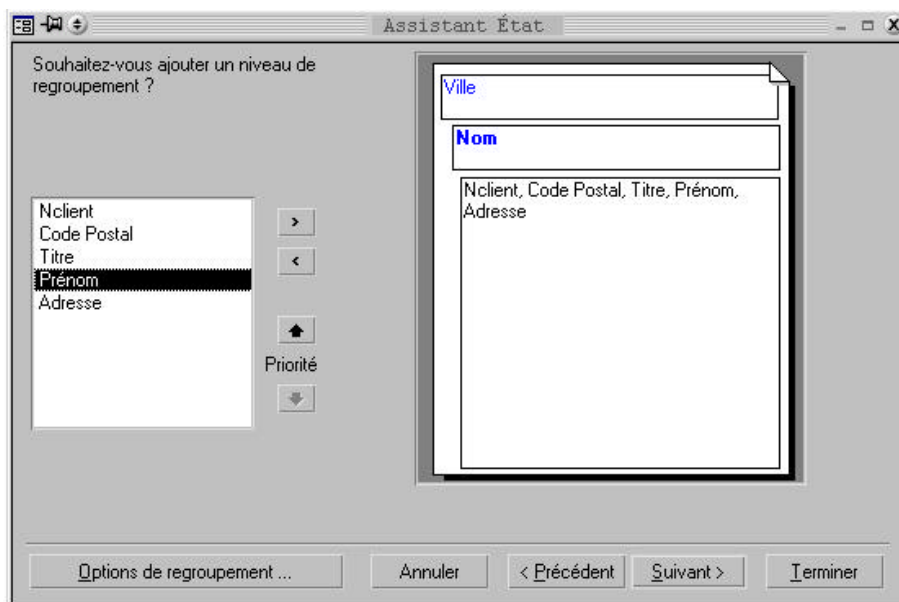
Notez que la source de l'état n'est pas nécessairement une table, on peut très bien générer un état à partir d'une requête. Le fonctionnement reste le même sauf qu'au lieu d'aller chercher des informations dans une table, Access ira les chercher dans la table résultant de la requête.



Ici, on va indiquer comment se fait le regroupement.

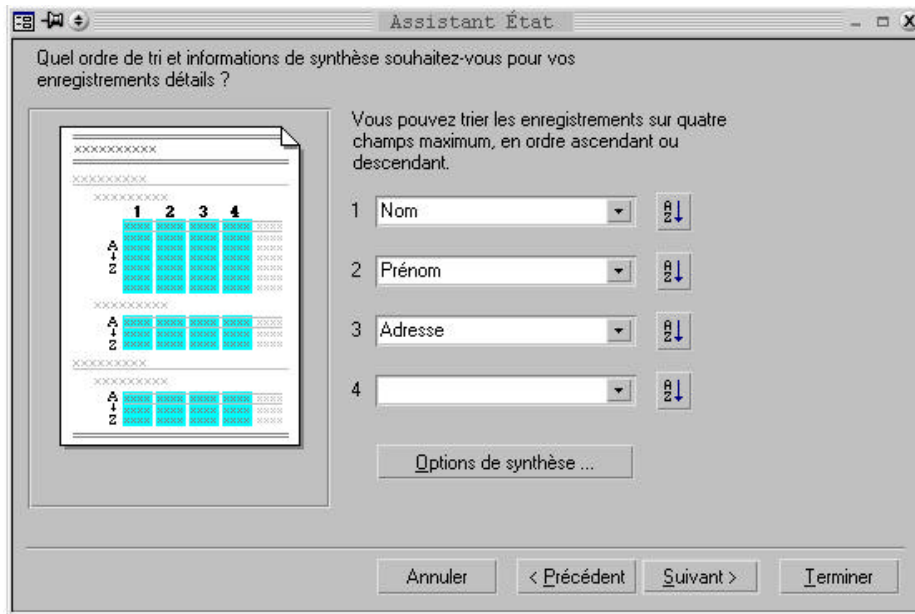
A quoi sert le regroupement ?

Par exemple, ici, on veut créer un état qui va afficher tous nos clients, on a choisit comme regroupement le champ "Ville", ce qui signifie que dans l'état qui va être généré, Access va afficher d'abord le nom de la ville, puis tous les clients habitant cette ville, puis la ville suivante et les clients et ainsi de suite. On peut choisir plusieurs niveaux de regroupement, on aurait pu ajouter un niveau supplémentaire avec par exemple le nom, dans ce cas, Access affichera d'abord la ville, puis le nom, puis tous les clients portant ce nom dans cette ville, puis le nom suivant, puis la ville suivante, etc ... :



On va se contenter de faire un regroupement par Ville, un regroupement pas nom n'a pas, en soi, tellement de sens.

On clique sur Suivant :

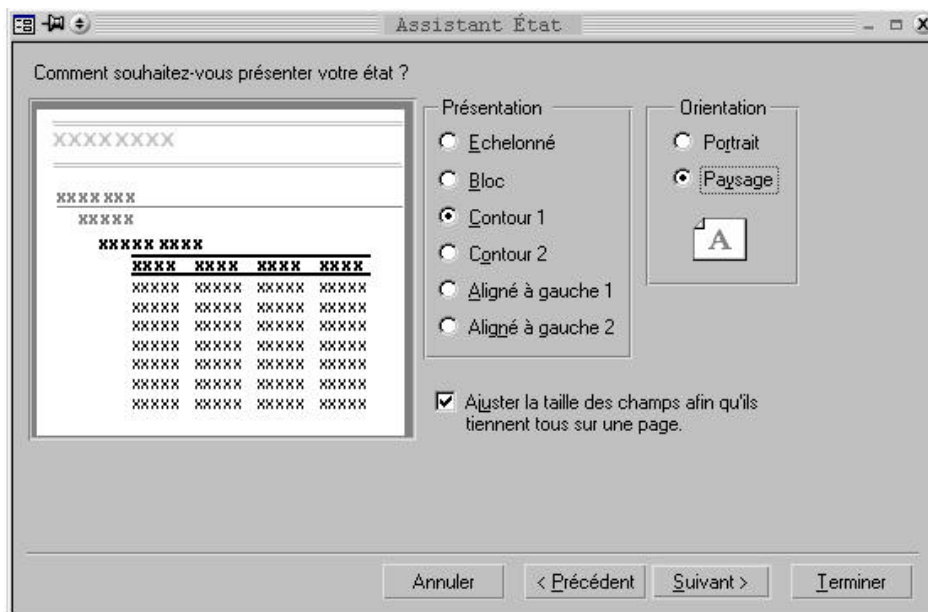


On va déterminer ici si on désire que les lignes de l'état soient triées et comment elles vont l'être :

On peut trier chaque ligne de détail (la ligne de détail est la ligne qui sera affichée pour chaque enregistrement) selon quatre critères, ces quatre critères sont des champs. On va choisir sur quels champs vont se faire ces tris, les lignes seront ainsi d'abord triées par nom, puis par prénom, etc.

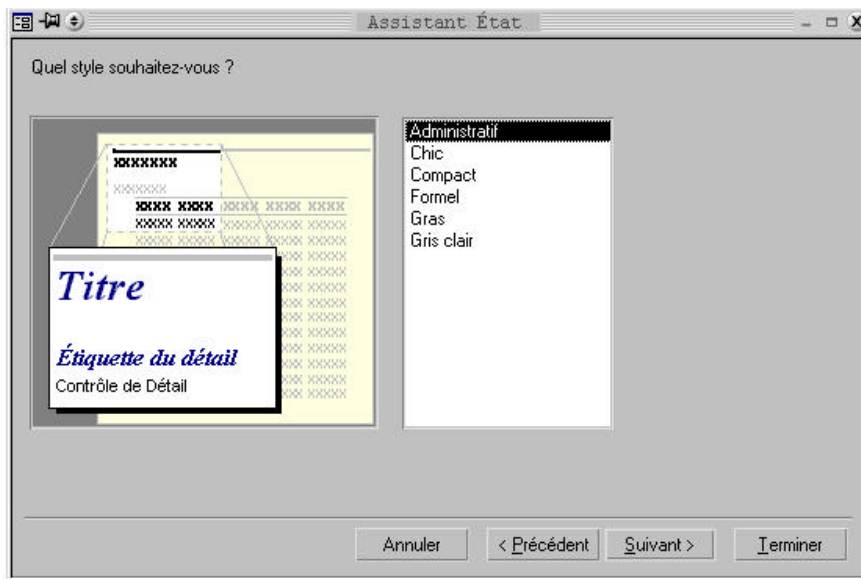
Notez que l'on peut trier soit par ordre alphabétique normal (par défaut) ou à l'envers (de Z à A) en cliquant sur l'icône (AZ) placée à droite du champ.

On clique ensuite sur "Suivant >"



On choisit ici le modèle de présentation de l'état.

Et enfin le style de l'état :



On nomme l'état, et Access va le générer automatiquement :

Ville					
	Titre				
Titre	Prénom	Adresse	N° client	Code Postal	Ville
Lille					
Benoît	Hugotte	142 Tr. L.	3	59000	Monsieur
Rois					
Alain	Jacques	142 Bis Tr.	1	75000	Monsieur
Georges	Arthur	142 Tr. C.	4	75000	Monsieur
Saint-Quentin					
Alain	Siroux	142 Tr. D.	2	02100	Monsieur
Charles	Pauline	142 Tr. E.	5	02100	Monsieur

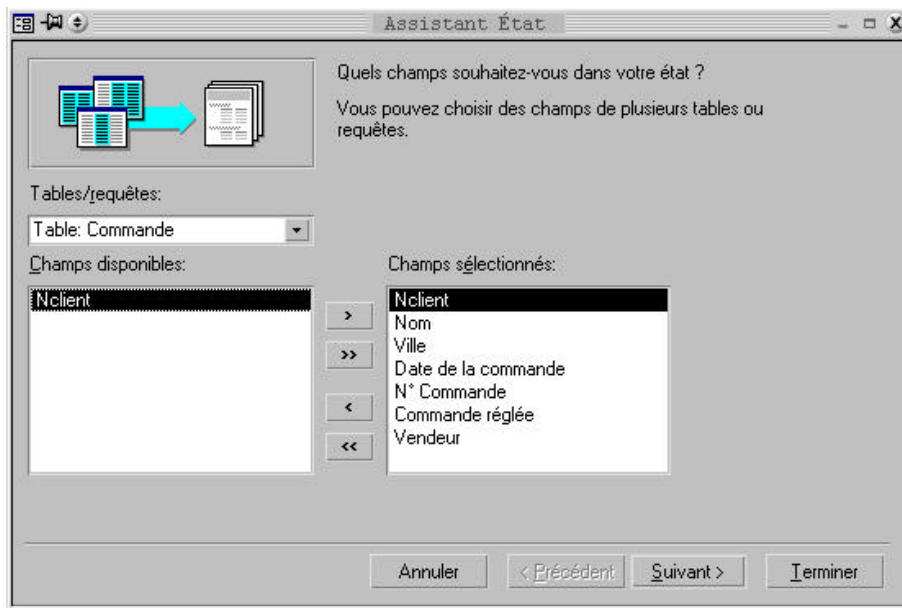
Comme vous le constatez, les clients, classés par nom, prénom et adresse, sont regroupés par Ville.

4. Création d'un état avec plusieurs tables

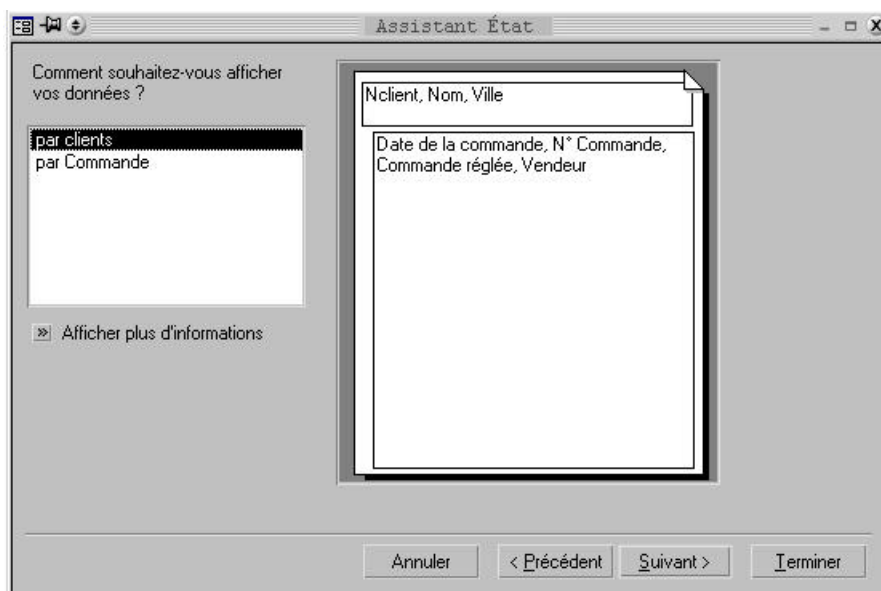
Jusqu'à présent, nos états n'affichaient que le contenu d'une table, nous allons créer un état qui va afficher la liste des clients, avec, pour chaque client, la liste des commandes qu'il a passé.

Attention, pour que ceci fonctionne, vous devez avoir créé correctement les relations entre les tables, sinon, Access risque de se perdre...

Comme avant, on crée un état grâce à l'assistant :

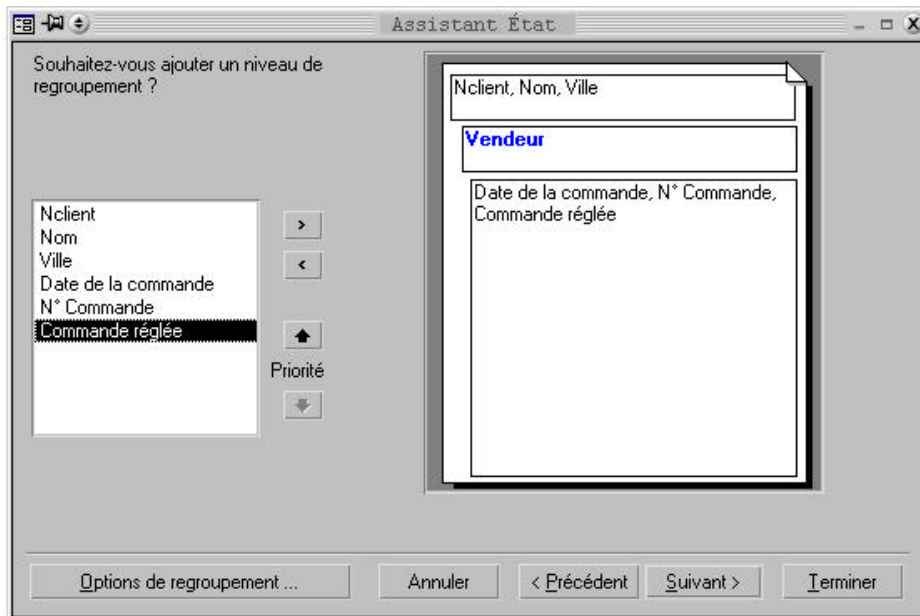


La différence est qu'ici, on va chercher des données appartenant à deux tables : Nclient, Nom et Ville proviennent de la table "Clients", N° Commande, Commande réglée et Vendeur proviennent de la table "Commandes".

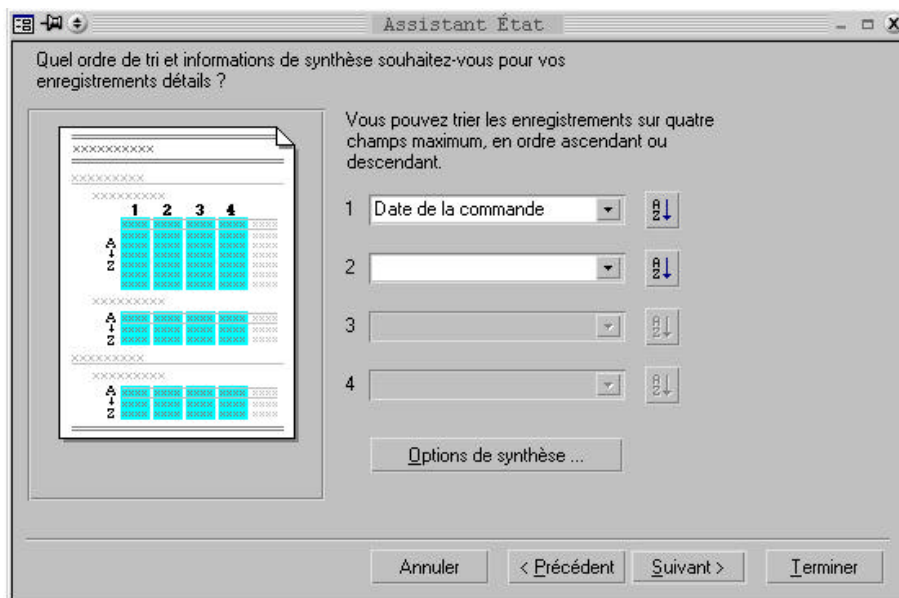


Et là, on voit que si on a bien fait les relations entre les tables, Access est intelligent, il a vu qu'à un client peut correspondre plusieurs commandes, il nous propose donc de générer un état qui va soit afficher tous les clients, avec, pour chaque client les commandes correspondantes, ou toutes les commandes, avec, pour chaque commande, le client qui l'a passé.

Ce qui nous intéresse n'est pas d'avoir la liste des commandes, car un client peut avoir passé plusieurs commandes, on retrouvera donc son nom plusieurs fois dans l'état, on va donc regrouper par client.



On peut ajouter ici un nouveau niveau de regroupement, je veux par exemple, pour chaque client la liste des commandes, mais en plus, je veux avoir cette liste groupée par vendeur.



Ici, on va indiquer comment va se faire le tri, on va trier sur le Date (pourquoi pas), on aura donc un état qui va afficher pour chaque client, les commandes qu'il a passé, triées par date et regroupées par vendeur.

On choisit ensuite la présentation et on donne un nom à l'état et Access génère :

<i>clients1</i>			
Nclient	2		
Nom	Albert		
Ville	Saint-Quentin		
<i>Vendeur</i>	<i>1</i>		
	<u>Date de la commande</u>	<u>N° Commande</u>	<u>Commande réglée</u>
	12/01/98	1	<input checked="" type="checkbox"/>
<i>Vendeur</i>	<i>3</i>		
	<u>Date de la commande</u>	<u>N° Commande</u>	<u>Commande réglée</u>
	12/05/55	3	<input checked="" type="checkbox"/>
Nclient	4		
Nom	Boudon		
Ville	Paris		
<i>Vendeur</i>	<i>1</i>		
	<u>Date de la commande</u>	<u>N° Commande</u>	<u>Commande réglée</u>
	15/04/57	2	<input type="checkbox"/>

Comme on peut le constater, on a ici le client, suivi de la liste de ses commandes, regroupées par Vendeur.

5. Modifier la structure d'un état

A présent, nous allons modifier l'état, nous allons ajouter, pour chaque client, une ligne qui va indiquer combien il a passé de commandes :

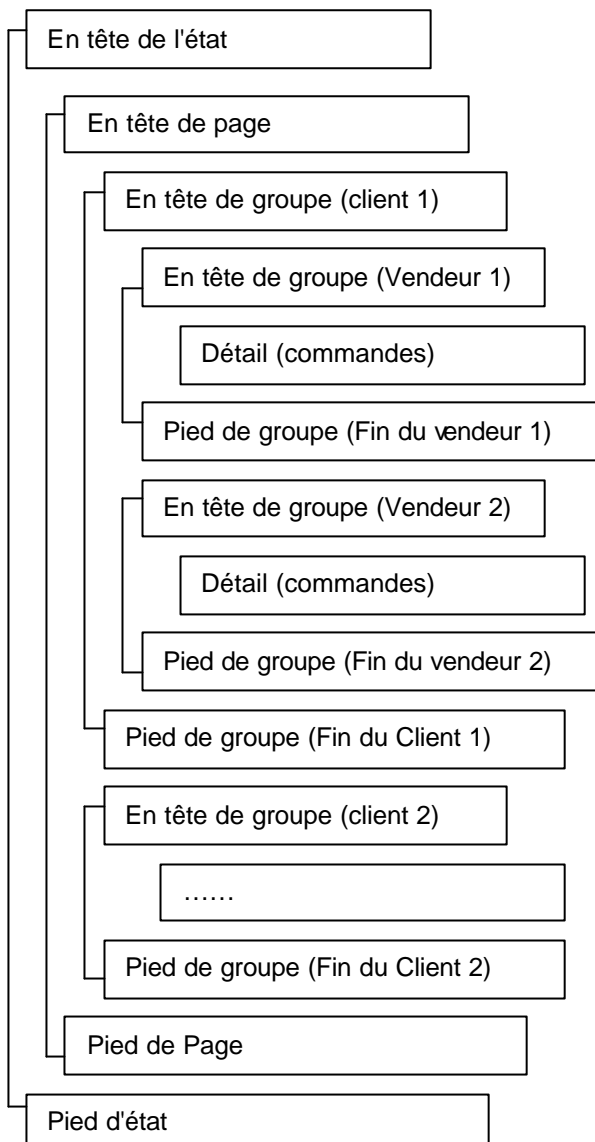
On clique sur l'état que l'on vient de créer et sur le bouton "Modifier" dans la fenêtre principale d'Access :

Comme on peut le constater, il y a plusieurs parties dans un état :

- ?? **L'en-tete de l'état** : c'est ce qui va être imprimé au début de l'état, ce ne sera imprimé qu'une seule fois.
- ?? **L'en-tete de page** : c'est ce qui sera imprimé au début de chaque page (un titre par exemple).
- ?? **L'en-tete de groupe** : c'est ce qui va être imprimé au début de chaque groupe, ici l'état est groupé par client, en en-tête du groupe client, on va donc imprimer les informations sur le client, il y a un sous-groupe dans le groupe client, c'est le groupe "Vendeur", on va donc imprimer dans l'en-tete de ce sous-groupe les informations sur les vendeurs.
- ?? **Le détail** : c'est ici que vont être imprimées chaque ligne de l'état, en l'occurrence ici les commandes
- ?? **Le pied de page** : c'est ce qui sera imprimé en bas de chaque page (souvent la date et la numérotation des pages)
- ?? **Le pied d'état** : c'est ce qui sera imprimé une seule fois à la fin de l'état.

A chaque en-tête peut correspondre un pied, ici les pieds pour les groupes Clients et Vendeur ne sont pas affichés, mais si ils l'étaient, à la fin de chaque groupement, le contenu du pied de groupe s'imprimerait, par exemple, après chaque client, avant d'imprimer l'en-tete pour le client suivant, Access imprimerait le pied pour le client, idem pour le groupe vendeur, avant d'imprimer l'en-tete du prochain vendeur, Access imprimerait le pied du groupe vendeur.

L'impression se fait ainsi :



On va afficher le nombre de commandes passées dans le pied du groupe client, ce qui est logique : à la fin d'un client, on affiche le nombre de commandes qu'il a passé :

Le pied du groupe client n'existe pas, il va falloir l'afficher, pour cela on clique sur l'icône :



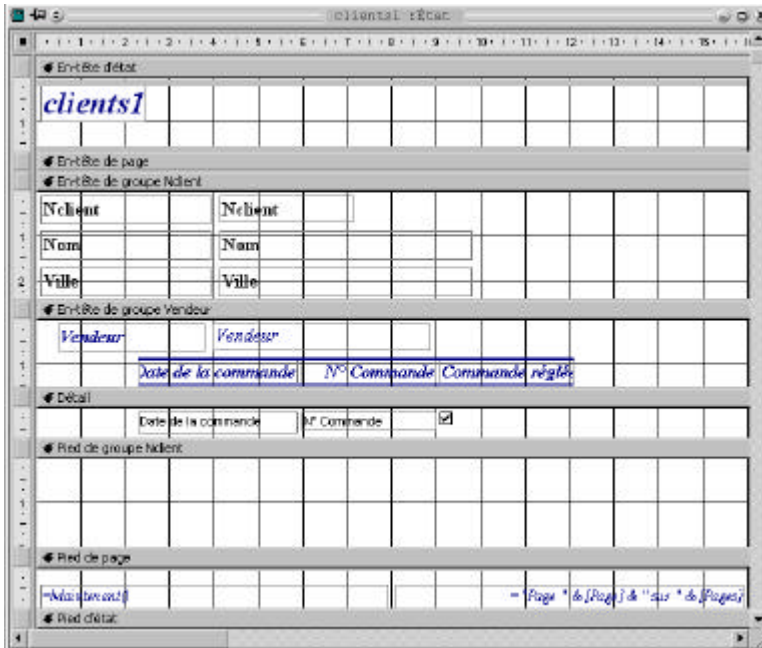
Access affiche alors :



Qu'avons-nous dans cette fenêtre ?

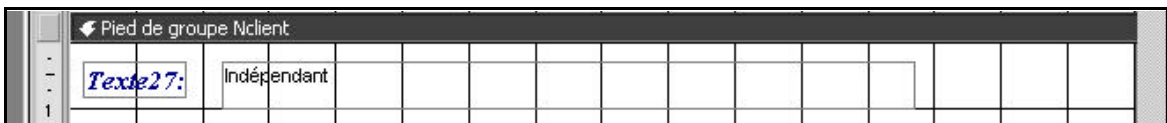
En haut, la liste des champs sur lesquelles sont groupées ou triées les informations dans l'état, lorsque les informations sont groupées sur un champ, le nom de ce champ est précédé d'un icône (ici l'état est groupé sur les champs Nclient et Vendeur), lorsque le champ est uniquement utilisé pour trier l'état, il n'est précédé d'aucun icône (le champ Date de Commande sert uniquement au tri, il n'a pas d'icône). On peut, dans cette fenêtre, ajouter un nouveau critère de tri ou un nouveau groupe.

Dans la partie inférieure se trouvent les propriétés de chaque champ. Pour le champ Nclient, on peut voir que, entre autres, la propriété "Pied de groupe" est à "Non", ce qui signifie qu'il n'y aura pas de pied de groupe pour ce groupe là, nous voulons qu'il y en ait un pour que nous puissions imprimer dans ce pied le nombre de commandes passées pour ce client. Nous choisissons "Oui".



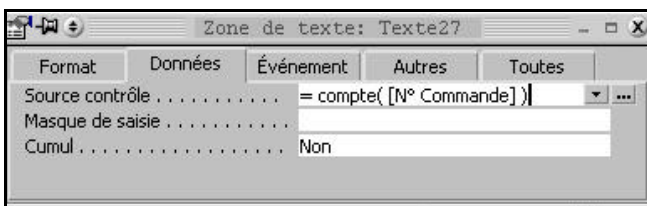
On peut constater que le pied de groupe "Nclient" est apparu.

Dans ce pied, nous allons ajouter les informations que nous désirons voir apparaître. Pour cela, nous procédons comme avec les formulaires, nous allons utiliser un contrôle "Zone de Texte" qui va nous permettre d'afficher le résultat d'un calcul (le calcul étant ici le nombre de commandes passées).



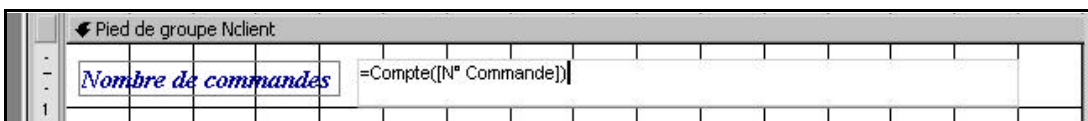
Nous avons notre contrôle qui, pour l'instant n'est lié à rien (il est indépendant).

Nous allons indiquer à Access que ce qui doit être affiché dans ce contrôle est le nombre de commandes passées pour ce client :



Pour cela nous utilisons la fonction "Compte" que nous avons vu dans les formulaires, cette fonction compte le nombre de fois que le champ passé en paramètre va être affiché.

On obtient donc dans le pied de groupe :



Lorsqu'on demande à visualiser l'état, on obtient :

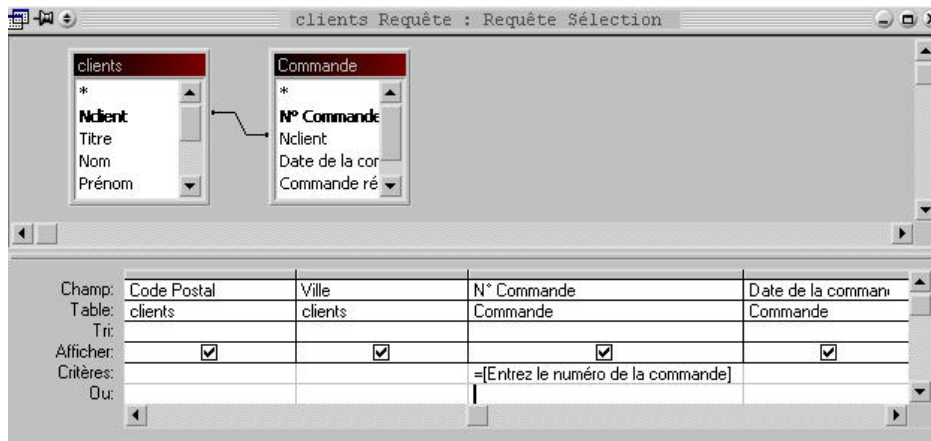
<i>clients1</i>		
Nclient		2
Nom	Albert	
Ville	Saint-Quentin	
Vendeur		1
	<u>Date de la commande</u>	<u>N° Commande</u> <u>Commande réglée</u>
	12/01/98	1 <input checked="" type="checkbox"/>
Vendeur		3
	<u>Date de la commande</u>	<u>N° Commande</u> <u>Commande réglée</u>
	12/05/55	3 <input checked="" type="checkbox"/>
Nombre de commandes :		2
Nclient		4
Nom	Boudon	
Ville	Paris	
Vendeur		1
	<u>Date de la commande</u>	<u>N° Commande</u> <u>Commande réglée</u>
	15/04/57	2 <input type="checkbox"/>
Nombre de commandes :		1

Vous pouvez voir, qu'en dessous du détail de chaque client, se trouve le nombre de commandes qu'il a passé.

6. Insertion d'un sous-état

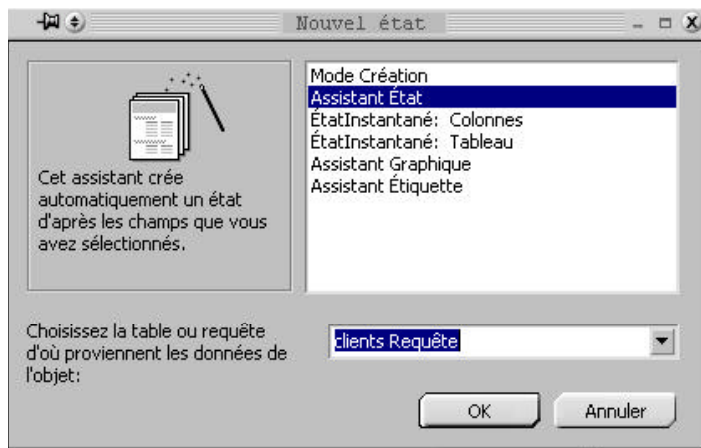
Comme pour les formulaires, on peut insérer dans un état un sous état. Les données du sous état seront imprimées en fonction de celles contenues dans l'état. Nous allons imprimer un bon de commande, le bon de commande va contenir toutes les informations concernant cette commande : les coordonnées du client, et le détail de chaque ligne de la commande.

Pour varier les plaisirs, nous allons baser cet état non plus sur une table comme précédemment mais sur une requête, cette requête va afficher, pour un numéro de commande donné, les informations la concernant : Date, réglée, n° du vendeur, nom et coordonnées du client :

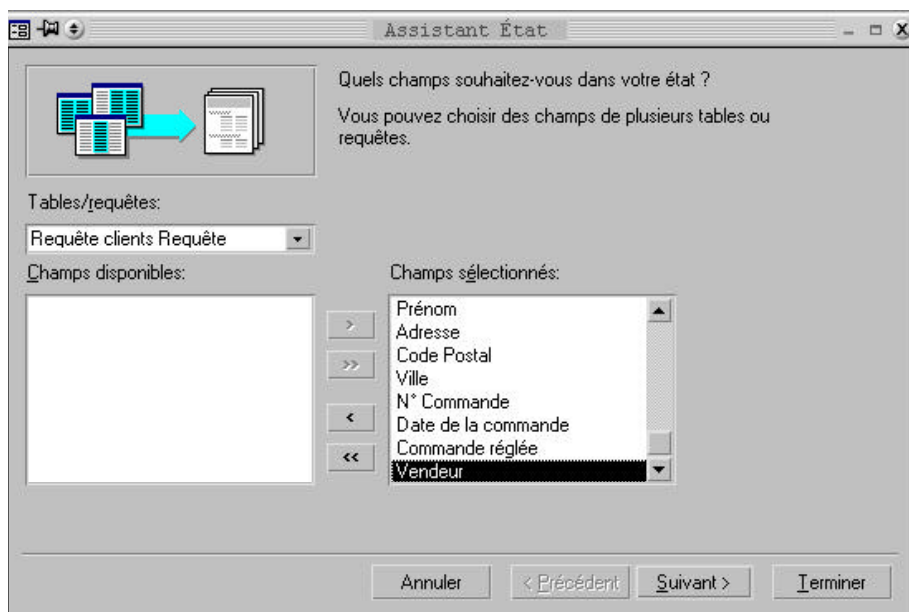


Je n'entre pas dans le détail de la requête, (vous êtes censé savoir comment on fait), sachez seulement que pour pouvoir saisir le numéro de commande, on place comme critère au champ "N° Commande" le critère "=[Entrez le numéro de la commande]."

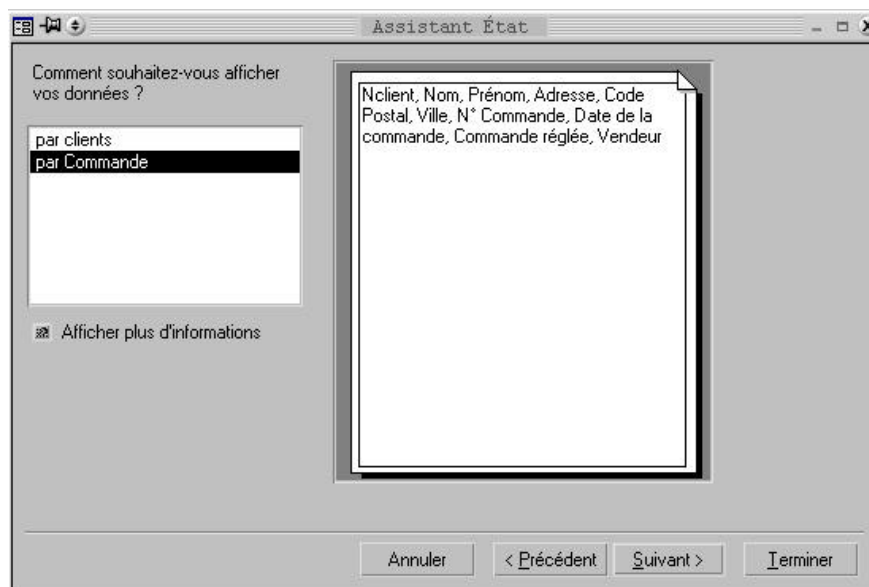
Une fois la requête créée, on va créer un état en le basant sur cette requête :



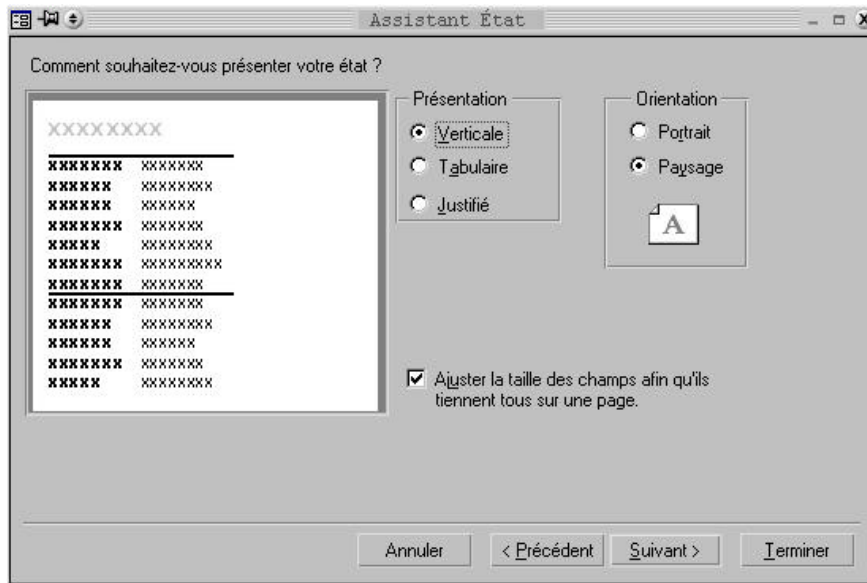
On choisit tous les champs de la requête :



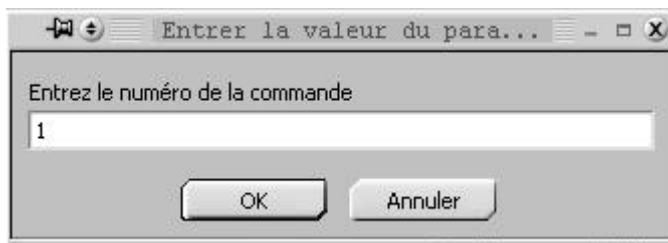
Cette fois-ci, on veut que l'état affiche les renseignements par commande (on veut imprimer un bon de commande) et non pas par client :



On indique dans l'écran suivant qu'on ne veut pas de niveau de regroupement supplémentaire et on choisit le format d'impression :



Choisissez la décoration et donnez un nom à votre état, Access génère l'état et affiche :



Pourquoi ? simplement parce que comme l'état est basé sur une requête, Access appelle d'abord la requête pour construire l'état, et cette boîte de dialogue est générée par la requête.

Il génère alors un état tout simple :

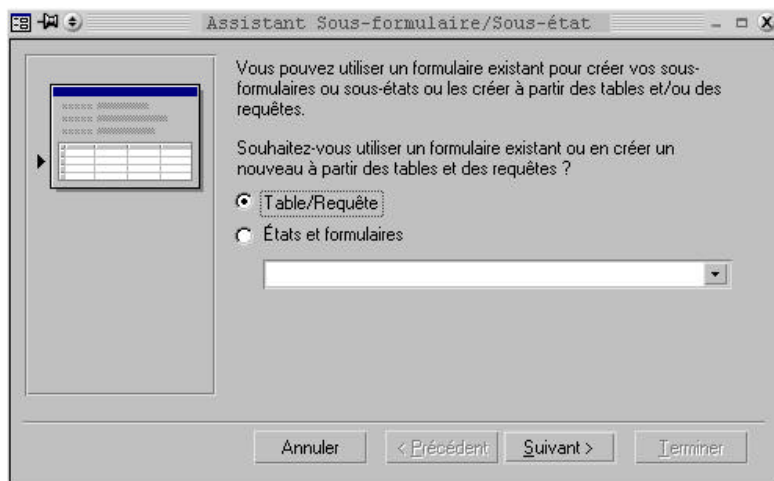
Commande

<i>Nclient</i>	<input type="text" value="2"/>
<i>Nom</i>	<input type="text" value="Albert"/>
<i>Prénom</i>	<input type="text" value="Simone"/>
<i>Adresse</i>	<input type="text" value="rue Minant"/>
<i>Code Postal</i>	<input type="text" value="02100"/>
<i>Ville</i>	<input type="text" value="Saint-Quentin"/>
<i>N° Commande</i>	<input type="text" value="1"/>
<i>Date de la comman</i>	<input type="text" value="12/01/96"/>
<i>Commande réglée</i>	<input checked="" type="checkbox"/>
<i>Vendeur</i>	<input type="text" value="1"/>

Nous allons, comme avec le sous-formulaire, ajouter un sous état à l'état actuel, ce sous état va afficher, pour la commande le détail de chaque ligne de la commande :

Pour cela, on passe en mode modification de l'état et on insère dans la partie "Détail" de l'état un contrôle sous-état :

Access nous demande à partir de quoi va être généré notre sous-état, il va l'être à partir de tables.



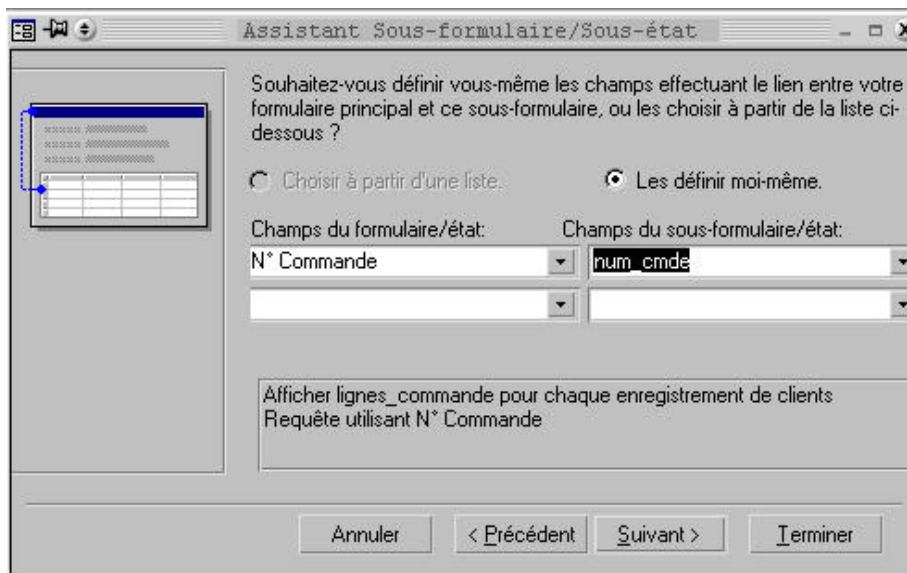
Access nous demande ensuite quels vont être les champs qui vont être affichés dans le sous-état.



On retrouve ici ce qu'on avait fait dans les formulaires, on indique ici les champs et les tables d'où proviennent les informations : num_prod, quantite et num_cde viennent de la table "lignes_commande" et libelle et prix unitaire viennent de la table "Produits"

Notez que le champ num_prod (numéro du produit) vient de la table "Lignes Commande" et non pas de la table "Produits". Pourquoi ? Parce qu'on veut la liste des produits commandés (donc présents dans la table Lignes Commandes"), si on avait choisi le numéro de produit venant de la table "Produit", cela aurait voulu dire qu'on voulait la liste de tous les produits présents dans la table "Produits".

Il va falloir maintenant indiquer à Access comment faire la relation entre l'état et le sous-état. De deux choses l'une, soit les relations sont bien faites et Access va savoir seul comment faire la relation, soit elles sont mal faites, ou Access ne voit pas comment faire et il va afficher la fenêtre suivante :



Ici, on va simplement indiquer à Access quel est le champ qui va être commun à l'état et au sous-état, c'est à partir de ce champ commun que le sous état va pouvoir être généré. On veut afficher, pour chaque commande le détail de chaque ligne de la commande, ce détail se trouve dans la table "Lignes_commande", dans la table "Ligne_Commande" se trouve le champ "num_cde" qui indique à quelle commande appartient cette ligne de commande.

La "jointure" entre ces deux tables va donc se faire sur le numéro de commande, d'un côté le champ "N° commande" dans la table "Commande" avec le champ "num_cde" dans la table "Lignes_commande" de l'autre.

D'ailleurs, on peut voir dès qu'on a choisi ces champs, qu'Access nous affiche en français la nature de cette relation entre les tables.

Access génère l'état et affiche :

Commande

Nclient

Nom

Prénom

Adresse

Code Postal

Ville

N° Commande

Date de la comman

Commande réglée

Vendeur

lignes_commande sous-état

<i>num_prod</i>	<i>quantite</i>	<i>num_cmde</i>	<i>libelle</i>
2	12	1	prod2
2	13	1	prod2
3	10	1	prod3

Vous pouvez constater que maintenant, pour la commande, s'affiche, en plus le sous état contenant le détail de chaque commande.

1. Introduction	2
2. Création d'une macro autonome	2
3. Exécuter la macro pas à pas.....	5
4. Modifier une macro.....	5
5. Création d'une macro associée à un formulaire	6
6. Exécuter des actions en fonction de conditions.....	7
7. Afficher un message.....	7
8. Déplacer le curseur	8
9. Affectation de la macro à l'événement.....	9
10. Définir la valeur d'un champ dans une macro	10
11. Mettre un bouton de commande dans un formulaire.....	11
12. Ouvrir un formulaire et afficher un enregistrement	12
13. Créer une boîte de dialogue	13

LES MACRO COMMANDES

1. Introduction

La macro commande permet d'automatiser certaines tâches d'Access, elle peut simuler une suite d'actions qui auraient du être faites par l'utilisateur.

Une macro est composée d'Actions, chaque action correspond à une tâche : lorsque vous exécutez la macro, Access exécute automatiquement les actions qu'elle contient. Certaines de ces actions, plus complexes, vous permettent d'afficher des boîtes de dialogue, de tester la réponse fournie par l'utilisateur, d'afficher une barre de menus personnalisée... et de développer une application autonome sans avoir besoin de programmer des modules en Visual Basic (bien que les actions fassent référence à des instructions en Visual Basic).

La méthode de création d'une macro est liée à deux facteurs importants :

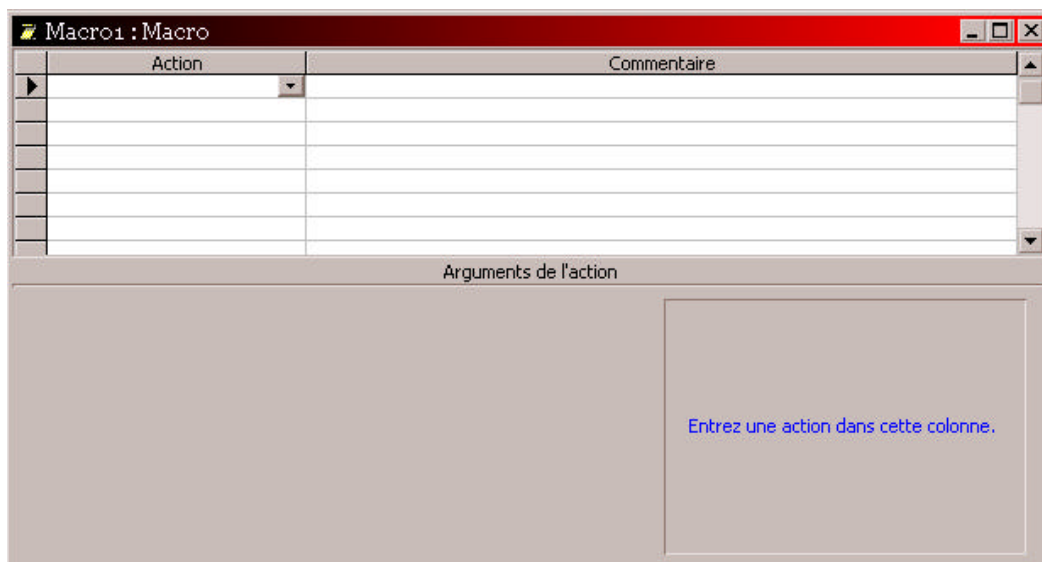
- ?? L'environnement de départ de la macro : certaines macros peuvent être exécutées quelle que soit la fenêtre active, d'autres sont liées à un objet de la base de données (formulaire, état, ...)
- ?? L'événement qui va déclencher l'exécution de la macro : dans un formulaire, ce peut être un clic sur un bouton, ce peut être l'ouverture du formulaire, la valeur du contenu d'un contrôle, ou encore en fonction de la mise en page d'un état.

2. Création d'une macro autonome

Une macro autonome n'est pas liée à un événement spécifique et peut être exécutée quelle que soit la fenêtre active.

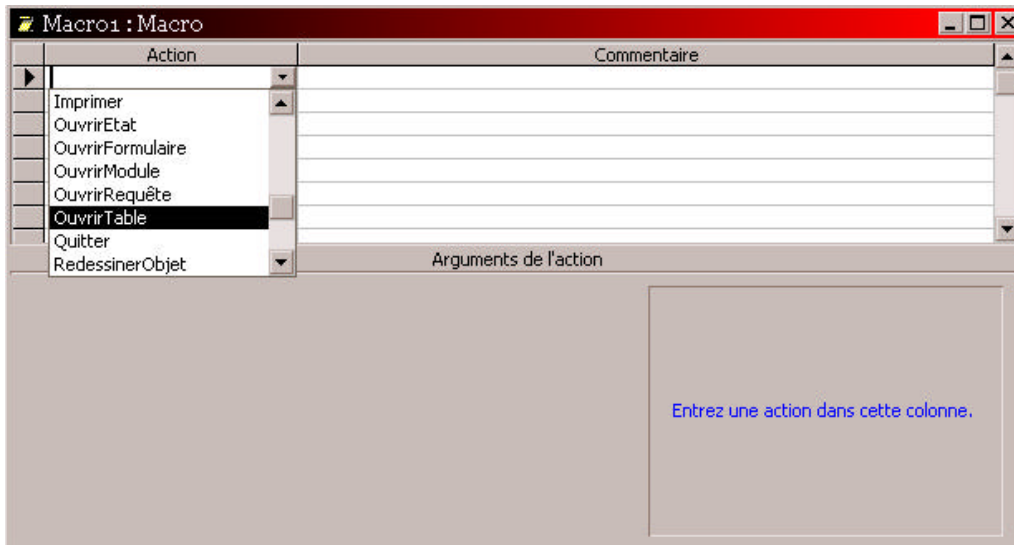
Nous allons créer une macro qui va ouvrir la table Clients et son formulaire :

1. Dans la fenêtre principale d'Access, cliquez sur l'onglet Macro et sur Nouveau :



La partie supérieure de la fenêtre (le tableau) est destinée aux différentes actions qui vont composer la macro. La plupart des actions ont des paramètres (par exemple l'action OuvrirTable qui ouvre une table demande en paramètre le nom de la table qu'elle va ouvrir), les paramètres (par exemple ici, le nom de la table) sera indiqué dans la partie inférieure de la fenêtre.

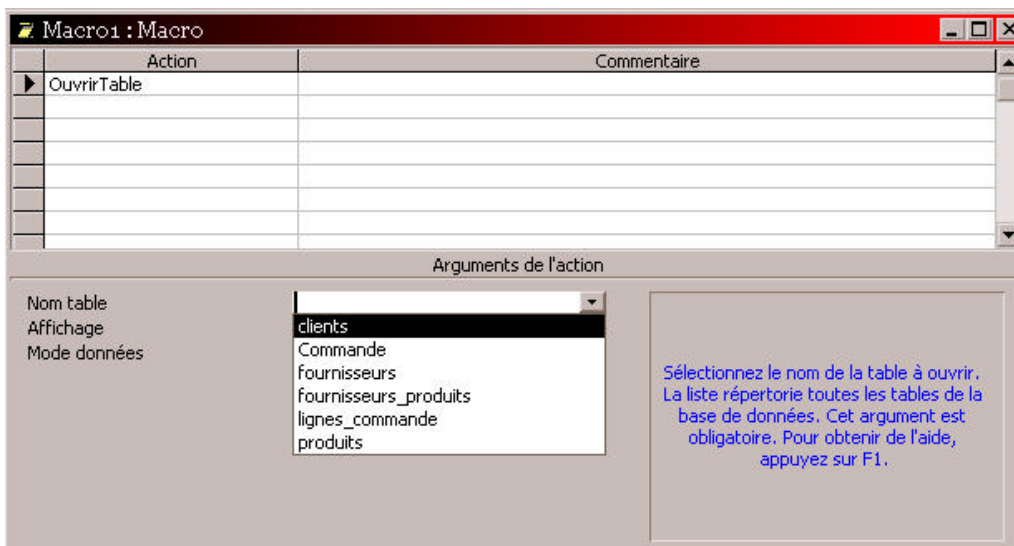
2. On choisit dans le menu déroulant l'action que l'on veut faire exécuter par la macro :



Comme vous pouvez le constater, il existe un très grand nombre d'actions possible, en fait, tout ce qu'on peut faire dans Access se retrouve dans ce menu. Pour avoir la liste des actions possibles, utilisez l'aide intégrée à Access qui décrit le fonctionnement de chaque action.

Nous voulons ouvrir une table, nous choisissons l'action "OuvrirTable".

3. Saisie des paramètres de l'action



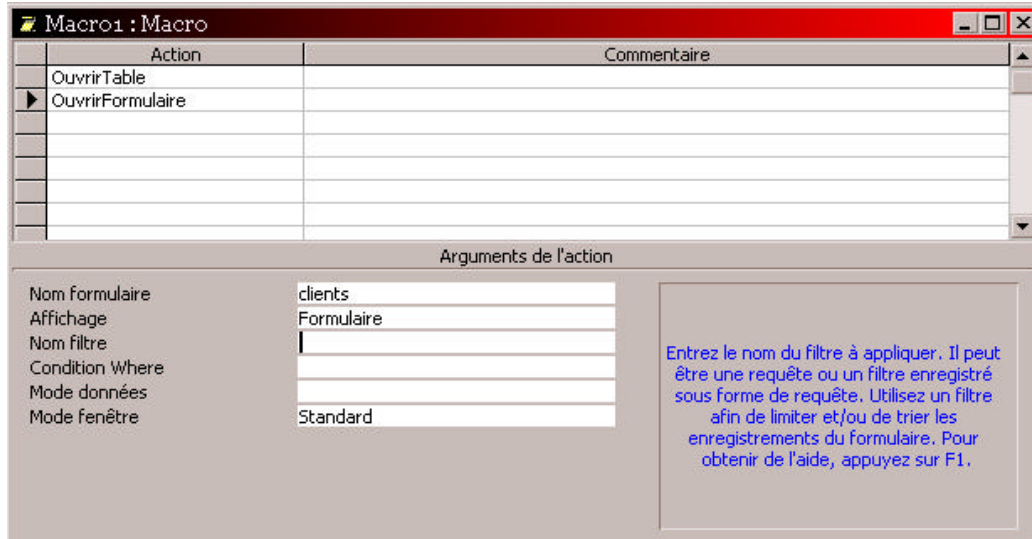
Comme vous pouvez le constater, dans le champ "Nom table" apparaissent les noms de toutes les tables de la base, on choisit "Clients".

Deux autres champs peuvent paramétrer l'action :

- ?? Affichage : détermine si la table sera ouverte en mode création ou en mode feuille de données
- ?? Mode données : Trois choix : Ajout (on peut ajouter des enregistrements à la table), Modification (on peut ajouter, modifier ou supprimer des enregistrements) et Lecture Seule (seule la consultation est permise).

4. Une deuxième action

Vous l'avez remarqué, le tableau de la fenêtre "Macros" contient plusieurs lignes, on peut exécuter plusieurs actions les unes à la suite des autres dans la même macro, maintenant, après avoir ouvert notre table Clients, nous allons ouvrir le formulaire associé à cette table :



L'Action "OuvrirFormulaire" a plus de paramètres :

- ?? Affichage : identique aux tables
- ?? Mode données : identique aux tables
- ?? Nom Filtre : on peut indiquer ici le nom d'une requête contenant les critères nécessaires à la sélection des enregistrements.
- ?? Mode fenêtre : indique comment le formulaire va être affiché : Standard (comme défini dans Affichage), Masquée (le formulaire est ouvert et automatiquement caché), Icône (le formulaire est réduit en icône), Boîte de dialogue (nous verrons ça plus tard)
- ?? Condition Where : plus tard...

5. Exécution de la macro

Sauvez votre macro.

Comme pour les requêtes, cliquez sur l'icône 'Point d'exclamation'. Access va alors exécuter toutes les lignes de la macro et s'arrêter dès qu'il va rencontrer une ligne vide, ou dès qu'il va rencontrer l'action "ArretMacro".

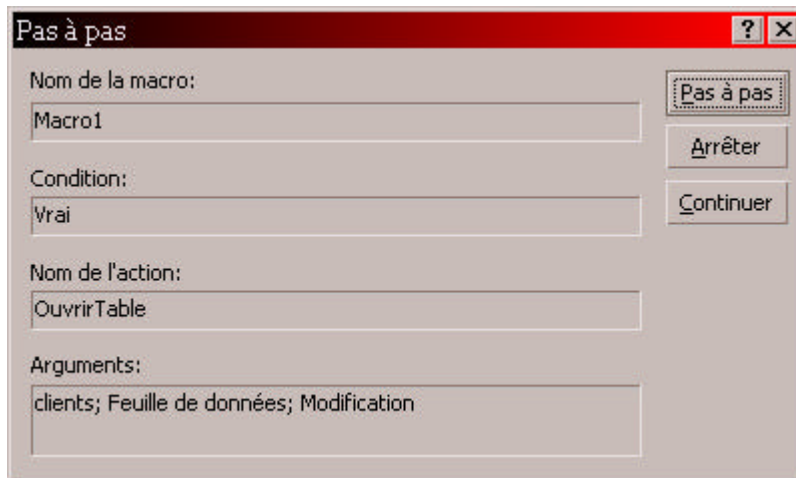
3. Exécuter la macro pas à pas

Pour analyser le déroulement d'une macro, on peut l'exécuter pas à pas, c'est-à-dire, action après action. Pour cela :

1. Ouvrir la macro en mode création
2. Cliquer sur l'icône :



3. Exécuter ensuite la macro



Access affiche la première action ainsi que ses paramètres, cliquez sur :

- ?? Pas à pas : pour exécuter cette action et passer à la suivante
- ?? Arrêter : pour stopper l'exécution de la macro
- ?? Continuer : Pour poursuivre l'exécution de la macro en mode normal

4. Modifier une macro

Pour modifier une macro, ouvrez là en mode création et positionnez vous sur l'action à modifier, cliquez dessus avec le bouton droit : vous pouvez alors la supprimer ou en insérer une autre.

5. Création d'une macro associée à un formulaire

Certaines macros doivent être exécutées en réponse à un événement lié au formulaire (ouverture, fermeture, fermeture...), d'autres dépendent d'un événement lié à un contrôle du formulaire (clic, valeur du contrôle, etc.). Dans le premier cas, la macro doit être insérée dans la feuille de propriétés du formulaire, dans le second cas, elle doit apparaître dans la feuille de propriétés du contrôle.

Il faut définir précisément quel est l'événement qui va déclencher l'exécution de la macro (quand tel événement se produit, telles actions vont se dérouler).

Prenons un exemple, nous voulons refuser l'enregistrement d'une commande si la date de cette commande n'a pas été saisie. (Nous pourrions dans les propriétés du champ date, indiquer dans la propriété Null Interdit Oui).

La question à se poser est : "Quand va-t-on vérifier que le champ date est vide ?", Il n'y a qu'une réponse : "Quand on passera à un nouvel enregistrement, que ce soit pour retourner sur un enregistrement précédent ou pour en créer un nouveau". Donc, lorsqu'on passera à un nouvel enregistrement, si la date est vide, on va afficher un message d'erreur, et positionner le curseur à nouveau sur le champ date pour pouvoir le saisir.

L'événement "passer à un nouvel enregistrement" est lié au formulaire et pas au contrôle "Date", l'événement sera donc associé au formulaire.

Voici la liste des principaux événements qui peuvent être associés à un formulaire :

Evénement	La macro doit s'exécuter
Sur Ouverture	A l'ouverture du formulaire
Sur Fermeture	A la fermeture du formulaire
Sur Activation	Lorsque Access accède à un enregistrement du formulaire
Sur Insertion	Lorsqu'on précise la valeur d'un champ d'un nouvel enregistrement
Sur Suppression	Lorsqu'on va supprimer un enregistrement
Avant MAJ	Après avoir quitté un enregistrement modifié et avant l'enregistrement des modifications
Après MAJ	Après avoir quitté un enregistrement modifié et après l'enregistrement des modifications.

En résumé, ces événements concernent l'ouverture, la fermeture du formulaire et la gestion des enregistrements.

Les événements liés aux contrôles d'un formulaire sont les suivants :

Evénement	La macro doit s'exécuter :
Sur Clic	Quand on clique sur le contrôle
Sur Double Clic	Quand on double-clique sur un contrôle
Sur Entrée	Avant d'accéder à un contrôle
Sur Sortie	Lorsqu'on quitte un contrôle
Avant MAJ	Après avoir quitté un contrôle modifié et avant qu'Access ne fasse la mise à jour
Après MAJ	Après avoir quitté un contrôle modifié et après sa mise à jour

Dans notre exemple, il faut que la macro s'exécute lorsqu'on va accéder à un nouvel enregistrement, on va donc l'affecter à l'événement "AvantMAJ" du formulaire : il faut que la vérification de la bonne saisie de la date se fasse lorsque l'on veut changer d'enregistrement mais avant qu'Access n'enregistre quoi que ce soit.

6. Exécuter des actions en fonction de conditions

1. On va créer une nouvelle macro pour vérifier la date, une fois la fenêtre de la macro ouverte, cliquez sur l'icône :



2. Une nouvelle colonne apparaît dans la fenêtre de la macro, c'est la colonne "Condition", c'est dans cette colonne qu'on va taper la condition qui décidera si oui ou non l'action de la macro doit s'exécuter.
3. On veut qu'un message d'erreur s'affiche lorsque la date de la commande est vide, la condition pour que le message s'affiche est donc : [date de la commande] Est Null. L'action qui va être déclenchée si la condition s'avère vraie va être l'affichage d'un message d'erreur.

7. Afficher un message

Pour afficher un message pendant une macro, on utilise l'action "BoîteMsg", "BoîteMsg" va afficher une boîte de dialogue Windows avec le texte et les caractéristiques qu'on va indiquer dans les paramètres de l'action.

L'action "BoîteMsg" a les paramètres suivants :

- ?? Message : texte à afficher
- ?? Bip : Emet un bip sonore lors de l'affichage du texte
- ?? Type : Icône à afficher à côté du texte (point d'exclamation, d'interrogation, etc...)
- ?? Titre : titre de la fenêtre où va s'afficher le texte

Condition	Action	Commentaire
[date de la commande] Est Null	BoîteMsg	

Arguments de l'action		
Message	<input type="text" value="Vous avez oublié de saisir la date !"/>	<p>Entrez le texte à afficher dans la barre de titre de la zone de message. Par exemple, 'Validation du code client'. Pour obtenir de l'aide, appuyez sur F1.</p>
Bip	<input type="text" value="Oui"/>	
Type	<input type="text" value="Point d'exclamation"/>	
Titre	<input type="text" value="Erreur de saisie"/>	

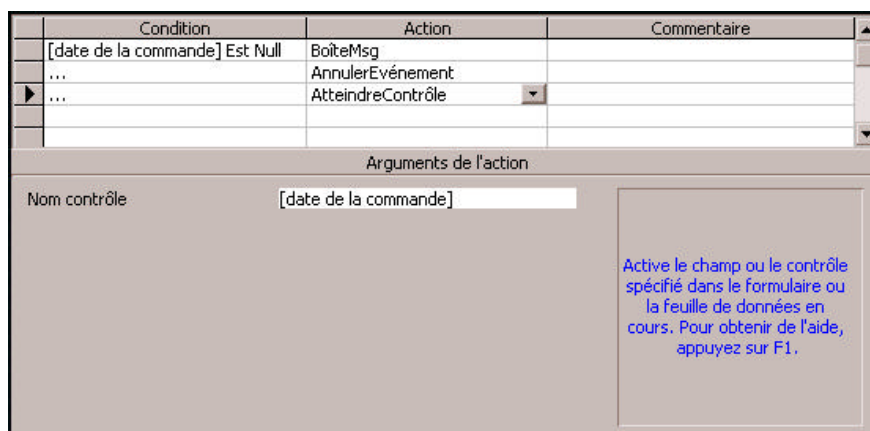
8. Déplacer le curseur

Une fois le message d'erreur affiché, il va falloir positionner le curseur automatiquement sur le contrôle Date, pour que l'utilisateur puisse le resaisir.

Il existe plusieurs actions pour déplacer le curseur :

- ?? AtteindreContrôle : Déplace le curseur sur un contrôle spécifié
- ?? AtteindrePage : Déplace le curseur sur le premier contrôle de la page spécifiée
- ?? AtteindreEnregistrement : Déplace le curseur sur l'enregistrement spécifié ou sur un nouvel enregistrement.

Nous devons déplacer le curseur dans le formulaire sur le contrôle du champ "Date", nous utiliserons donc l'action "AtteindreContrôle" :



Quoi de neuf dans cette fenêtre ?

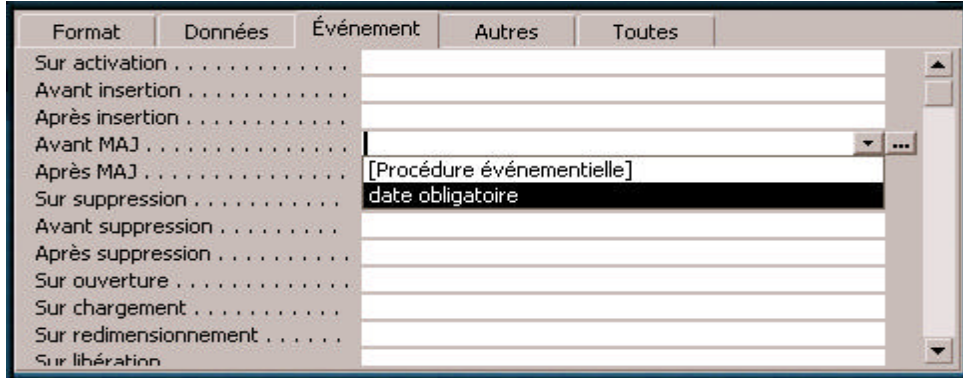
Sur la première ligne on a la condition : si la date de la commande est vide, on affiche le message d'erreur, mais après avoir affiché l'erreur que fait-on ? on va faire plusieurs choses, et pour indiquer que les autres actions que l'on va faire vont se dérouler dans le cas où l'erreur s'est produite, on va mettre comme condition "...", autrement dit, à chaque fois qu'Access va rencontrer comme condition "...", il va reprendre la condition spécifiée explicitement plus haut.

Donc, après avoir affiché l'erreur, on va exécuter l'action "AnnulerEvénement". A quoi sert cette action ? Normalement, après avoir modifié le formulaire, Access va mettre à jour les champs correspondants dans les tables liées à ce formulaire. Juste avant de faire cette mise à jour, Access va appeler la macro associée à l'évènement « AvantMAJ ». La macro va être exécutée, puis la mise à jour sera faite. Il faut qu'on annule cette mise à jour, sinon, le message indiquant que la date est vide va être affiché, puis Access va sauvegarder les modifications et sauver dans la table commandes la date vide. L'action « AnnulerEvénement » annule la mise à jour qui allait être faite, ainsi, la date vide ne sera pas sauvée.

Enfin, on va déplacer le curseur sur le contrôle "Date de la commande" grâce à l'action "AtteindreContrôle", Ainsi le curseur cliquera directement sur le contrôle où l'utilisateur devra saisir la date.

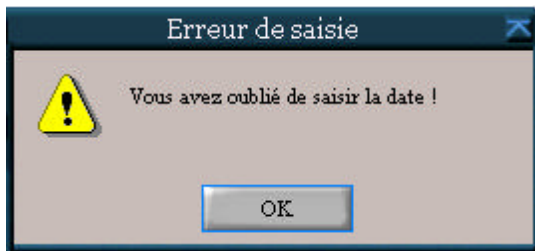
9. Affectation de la macro à l'événement

Maintenant que la macro est créée, on va l'affecter à la macro à l'événement "AvantMAJ" du formulaire "Commandes". Pour afficher les propriétés d'un formulaire, cliquez avec le bouton droit sur le carré noir situé en haut à gauche du formulaire, puis dans le menu, cliquez sur "Propriétés"



Choisissez l'événement auquel vous voulez associer la macro, dans le menu déroulant s'affichent toutes les macros existantes, on choisit "Date Obligatoire".

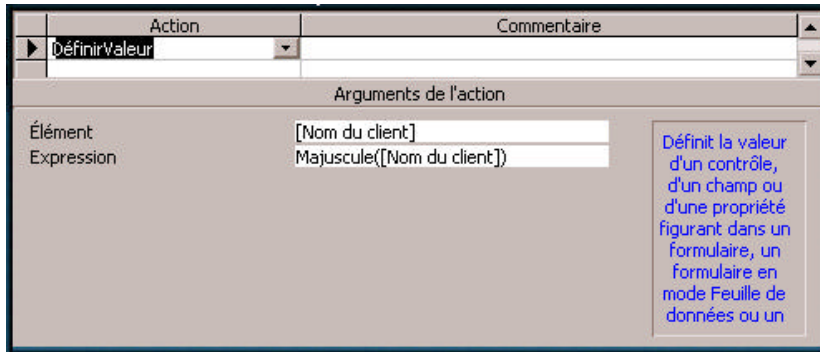
Une fois la macro associée à l'événement, il suffit d'entrer une commande dans laquelle on omet de saisir la date pour vérifier que la macro va s'exécuter (message d'erreur et repositionnement sur le contrôle) :



10. Définir la valeur d'un champ dans une macro

On peut, grâce à une macro modifier le contenu d'un champ d'une table. Imaginons que nous voulions que le nom de chaque client saisi soit en majuscule, on pourrait bien sur le faire en modifiant la propriété "Masque de saisie" du contrôle "nom du client", mais nous allons voir comment le faire à partir d'une macro. Pouvoir modifier le contenu d'un champ lors d'un événement peut s'avérer très utile.

Nous allons créer une nouvelle macro et utiliser l'action "DéfinirValeur"



Quels sont les paramètres de l'action "DéfinirValeur" ?

- ?? Élément indique qu'est-ce qui est à modifier, ce peut être comme ici le nom d'un champ, mais ce peut être aussi une propriété d'un contrôle par exemple.
- ?? Expression : Indique la nouvelle valeur de l'élément indiqué précédemment, ce peut être une valeur indiquée explicitement, le contenu d'un autre champ, le résultat d'un calcul, ou, comme ici le résultat d'une fonction. La fonction "Majuscule()" renvoie la chaîne de caractère passé en paramètre en majuscule.

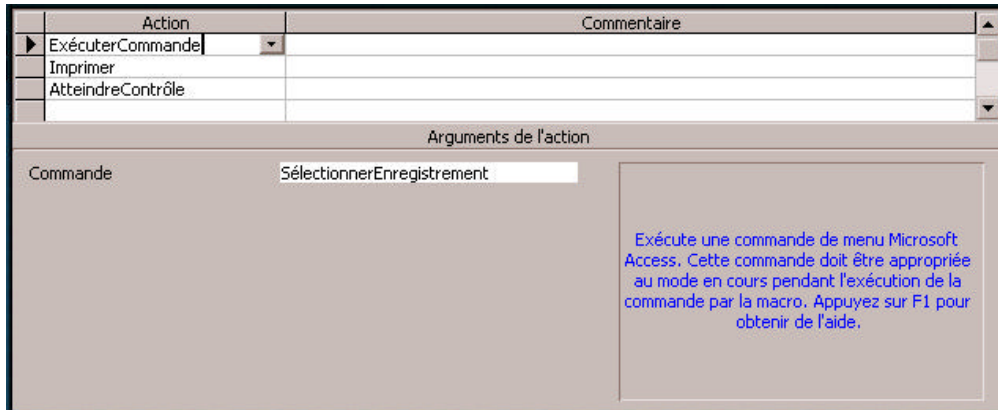
Comme précédemment, on va affecter cette macro à l'événement "AvantMAJ" su formulaire.

11. Mettre un bouton de commande dans un formulaire

Nous avons vu rapidement dans les formulaires que, parmi les contrôles existants, il y avait un contrôle nommé "Bouton de Commande". Ce bouton, placé sur un formulaire, permet d'exécuter soit n'importe quelle action d'Access, soit, et c'est ce qui nous intéresse ici, une macro.

Dans notre formulaire "Bon de commande +", qui contient la commande ainsi que son détail, nous allons ajouter un bouton qui, lorsqu'on cliquera dessus, déclenchera l'impression du bon de commande.

Nous allons créer une macro qui imprimera le formulaire :



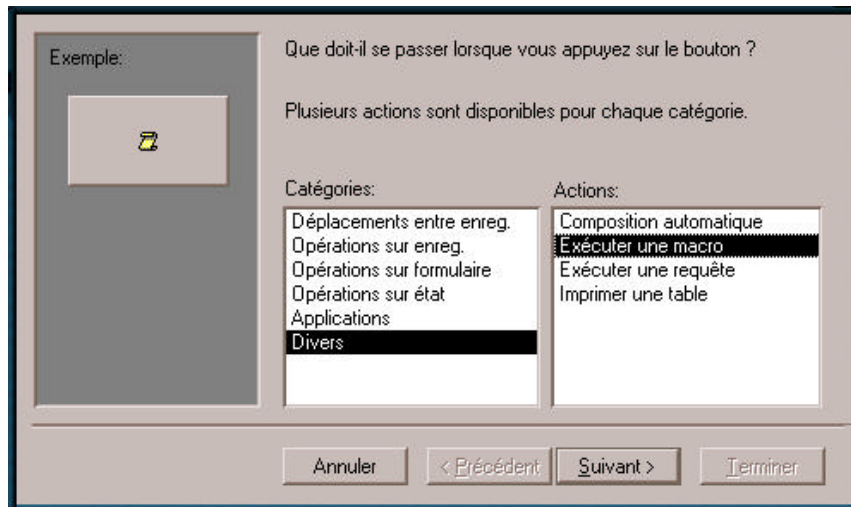
Un petit peu compliqué pour imprimer un bon de commande ?

La macro comporte trois actions, pourquoi ?

Si on n'utilise que l'action "Imprimer", Access va imprimer tous les bons de commande, il faut donc limiter l'impression au bon de commande courant, pour cela, on procède en trois étapes :

- ?? D'abord, on appelle l'action "ExecuterCommande", "ExecuterCommande" permet d'exécuter toutes les opérations possibles dans Access, on va donc choisir parmi la liste proposée, l'action "SélectionnerEnregistrement", cette action va sélectionner l'enregistrement courant. Une fois cet enregistrement sélectionné, on va pouvoir l'imprimer.
- ?? Ensuite, on appelle l'action "Imprimer", parmi les paramètres de cette commande, il y a la définition de ce qui a à imprimer : toutes les pages, un groupe, une sélection, ... on choisit la sélection.
- ?? Enfin, on appelle l'action "AtteindreContrôle", cette action va positionner le curseur sur le contrôle passé en paramètre, ce qui a pour effet de supprimer la sélection.

On insère ensuite un bouton de commande dans le formulaire, la fenêtre suivante s'affiche :



On peut choisir toutes les actions possibles, parmi celles-ci, dans la catégorie "Divers", il y a "Exécuter une macro", après avoir cliqué sur suivant, Access nous demande le nom de cette macro, on choisit celle que l'on vient de créer, puis le texte ou l'icône du bouton.

Vous pouvez essayer, ça marche !

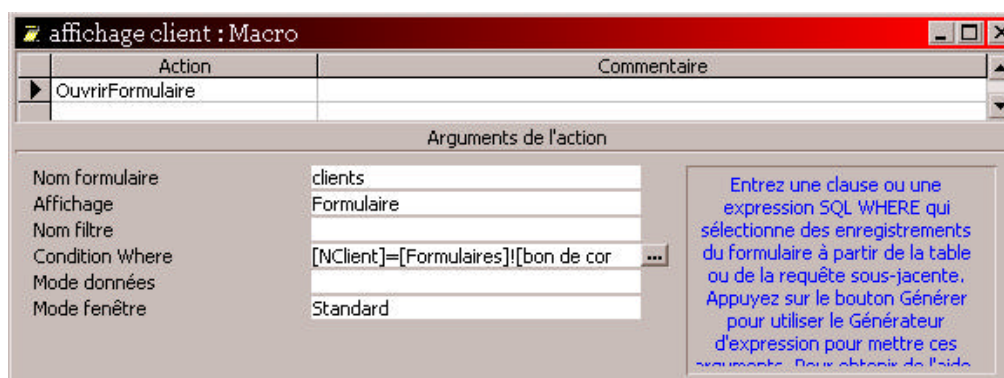
Si par la suite, vous allez regarder les propriétés du bouton qui vient d'être créé, vous verrez que la macro a été affectée à l'événement "Sur Clic", ce qui signifie, qu'à chaque fois qu'on cliquera sur le bouton, la macro se déclenchera, vous pouvez la placer ailleurs, par exemple sur l'événement "Sur Double Clic", il faudra alors double-cliquer sur le bouton pour imprimer.

12. Ouvrir un formulaire et afficher un enregistrement

Notez que cette manipulation n'est donnée qu'à titre d'exemple, on peut faire la même beaucoup plus simplement en utilisant l'assistant du bouton de contrôle.

Nous allons ajouter un bouton sur le bon de commande, ce bouton affichera les informations complètes sur le client concerné par le bon de commande :

D'abord, on crée une macro qui va ouvrir le formulaire :



On indique qu'on ouvre le formulaire "Clients", et ici, on utilise la condition Where qu'on avait vu plus tôt, on va indiquer que dans le formulaire qui va être ouvert, le champ [Nclient] (numéro de client) va être égal à [Formulaires]![bon de commande +]![Nom Client]. Ce qui signifie que le champ [Nclient] dans le formulaire "Clients" va être égal au contenu du contrôle "Nom Client" dans le formulaire "Bon de commande+".

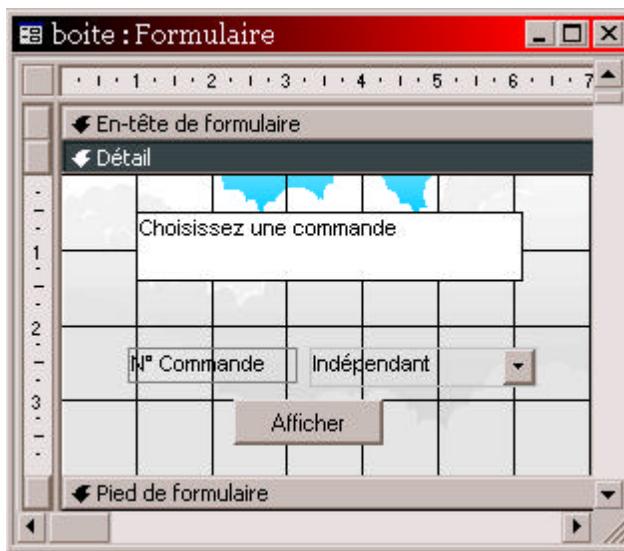
13. Créer une boîte de dialogue

Il peut être pratique de créer une boîte de dialogue pour répondre à une question par exemple, nous allons créer une boîte de dialogue dans laquelle nous allons entrer un numéro de commande, et qui affichera le formulaire correspondant à cette commande.

1. Création de la boîte de dialogue

Une boîte de dialogue est un formulaire standard qui a des propriétés particulières : Dans la fenêtre propriétés du formulaire, cliquez sur l'onglet "Toutes", et mettez la propriété "Fen Indépendante" à oui, "Fen Modale" à oui, "Auto Center" à oui, "Boite Contrôle" à non et "Affich par défaut" à "Mode simple".

Ce qui aura pour effet de créer un formulaire qui sera centré à l'écran, qui n'affichera pas le menu système de windows en haut à gauche et qui n'affichera pas les icônes pour se déplacer sur les enregistrements.



La liste de choix contient le résultat d'une requête allant chercher les numéros de toutes les commandes.

2. On crée ensuite la macro

La macro est du même type que la précédente, la clause Where ici est :

```
[N° Commande]=[Formulaires]![boîte]![liste commandes]
```

où [N° Commande] est le nom du champ contenant le numéro de commande dans le formulaire "Bond de Commande" et [liste commandes] est le nom de la liste contenant la liste des commandes existantes.

Voilà, nous avons fait le tour rapidement des fonctionnalités offertes par l'utilisation des macros, on peut, bien sur, pousser plus loin et faire beaucoup d'autres choses avec, à vous d'expérimenter et de trouver d'autres utilisations aux macros.