

# SUPPORT SQL

Thierry GRANDADAM

# TABLE DE MATIERES

1.	PRESENTATION DE SQL .....	1
1.1.	Historique.....	1
1.2.	Mise en œuvre.....	1
1.3.	Les versions.....	1
2.	LES RESEAUX.....	2
2.1.	Les différentes familles .....	2
2.2.	Définition d'un réseau local .....	2
2.2.1.	Le concept de réseau global.....	2
2.2.2.	Les applications .....	3
2.2.3.	Les avantages .....	3
2.3.	Réseau local ou système multi-utilisateur ? .....	4
2.4.	Les types de réseaux locaux .....	4
2.4.1.	Les réseaux hiérarchique .....	4
2.4.2.	Réseau poste à poste.....	5
2.4.3.	Réseau client-serveur.....	6
2.5.	Support de transmission .....	6
2.5.1.	Principes.....	6
2.5.2.	Les différentes types.....	7
	<input type="checkbox"/> <u>Câble coaxial</u> .....	7
	<input type="checkbox"/> <u>Paire torsadée</u> .....	7
	<input type="checkbox"/> <u>Fibre optique</u> .....	8
	<input type="checkbox"/> <u>Sans fil</u> .....	8
	<input type="checkbox"/> <u>Tableau récapitulatif</u> .....	8
2.6.	Topologie .....	9
2.6.1.	Le bus.....	9
2.6.2.	L'étoile .....	10
2.6.3.	Mixte.....	10
3.	NOTION FONDAMENTALE .....	12
3.1.	Principe d'une base de données.....	12
3.2.	Principe d'une clé.....	13
3.3.	Conception.....	13
3.4.	Les relations .....	14
4.	COMMANDES DE BASES.....	16
4.1.	Principes .....	16
4.2.	Création/Suppression d'une table .....	16
4.3.	Saisies des données.....	19
4.4.	Voir les données .....	19
4.5.	Recherche simple des données.....	19
4.6.	Recherche paramétrable .....	20
4.7.	Modifier les données.....	21
4.8.	Supprimer des données.....	21

4.9. Elimination des doublons .....	21
4.10. Limitation du résultat .....	22
4.11. Modifier la structure .....	22
4.12. Les transactions .....	22
5. REQUETE SUR PLUSIEURS TABLES.....	23
5.1. Jointure .....	23
5.2. Elimination des doublons .....	24
5.3. La clause IN .....	24
5.4. La clause EXISTS.....	24
5.5. Les alias.....	25
5.6. La fusion de deux tables .....	25
6. LES CALCULS .....	26
6.1. Création d'un champs calculé .....	26
6.2. Les statistiques .....	26
6.3. La clause GROUP BY.....	26
7. L'ADMINISTRATION .....	27
7.1. Les vues .....	27
7.2. La sécurité .....	27
7.2.1. Mettre des droits .....	27
7.2.2. Enlever des droits .....	28
7.3. Les index : .....	28
7.3.1. Index simple.....	28
7.3.2. Index multiple.....	29
7.3.3. Suppression.....	29
7.4. Les contraintes d'intégrité .....	29
7.5. Fusion .....	30
8. SQL AVEC EXCEL.....	31
8.1. Les tableaux .....	31
8.2. Création d'une requête .....	31

# 1. PRESENTATION DE SQL

## 1.1. Historique

La naissance des bases de données (SGBD) de différents éditeurs au début de l'année 70 dans les environnements Unix a vite montré ses limites et surtout si leur création était faite avec des différents logiciels il a fallu très vite mettre au point un langage qui permet de manipuler et d'interroger les données. Le leader du marché qui est IBM, a mis au point un langage nommé SEQUEL fonctionnant avec leur base de données relationnelle nommée "système R". En 1980 le langage a été renommé SQL (Structured Query Language) de façon à éviter la confusion avec un autre produit matériel.

Aujourd'hui sur le marché il existe un autre SQL : MySQL. Cette version s'applique à d'autres langages de programmation comme C++, JAVA, PERL,... et de plus il est gratuit.

## 1.2. Mise en œuvre

Pour faire du SQL il vous faut :

- ❑ Une base de données (la plupart du marché accepte SQL)
- ❑ Un éditeur de texte comme le bloc note car SQL accepte le format libre sans aucune mise en forme
- ❑ Eventuellement un réseau pour faire des essais dans cette architecture.

## 1.3. Les versions

Normalement il devrait avoir qu'une seule version mais beaucoup de sociétés d'édition de bases de données comme Oracle, Informix, Microsoft ont modifié ce langage pour l'adapter à leur spécificité. Attention les instructions sont toujours les mêmes mais la syntaxe peut être différentes. Comme pour l'apprentissage d'un langage étrangère, le plus dur est d'apprendre la grammaire et non les mots.

Pour ma part, je vais considérer que vous utilisez un PC avec le pack office d'installé donc Access.

Pourquoi avoir choisi Access ?

Pas par conviction car je n'aime pas les gens qui s'amuse à modifier un standard mais cela représente plus de 36% du marché des SGBD avec notamment SQL Server et en deuxième position on trouve Oracle avec 24% et ensuite IBM avec Informix dans le monde du PC, car dans le monde des mini-ordinateurs Oracle est très largement leader et même devant IBM l'inventeur.

Avant de commencer à apprendre le langage SQL vous devez avoir plusieurs pré-requis : les réseaux et les notions fondamentales sur les bases de données. Pourquoi les réseaux, car la plupart des grosses bases de données fonctionnent dans un environnement multi-utilisateurs.

Je vais commencer par les réseaux.

## 2. LES RESEAUX

### 2.1. Les différentes familles

Les réseaux en informatique sont classés suivant leur taille ou plus exactement suivant leur longueur. Nous trouvons trois familles différentes :

- **L.A.N.** (Local Area Network) : longueur n'excédant pas 6 kms.
- **M.A.N.** (Metropolitan Area Network) : longueur n'excédant pas 80 kms.
- **W.A.N.** (Wide Area Network) : longueur n'ayant pas de limites.

### 2.2. Définition d'un réseau local

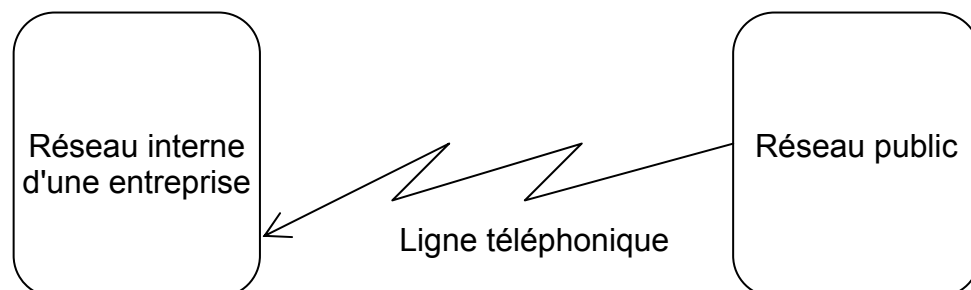
Donner une définition à la fois pertinente et détaillée d'un réseau local est une tâche difficile. Un réseau comportant des terminaux à distance connectés par modems peut-il toujours être qualifié de local ? Et que penser d'un réseau local dépourvu de tout support de transmission physique ? Existe-t-il une différence nette entre les réseaux locaux LAN, les réseaux urbains MAN et les réseaux longue distance WAN ?.

Il est utile mais pas essentiel de donner une définition des réseaux locaux. La description des caractéristiques générales d'un éléphant permet de s'en faire une représentation plus claire qu'une longue définition scientifique.

De même, avec les réseaux locaux, une simple description générale peut se révéler plus utile qu'une définition longue et précise. Voici trois perspectives différentes pour définir les réseaux locaux.

#### 2.2.1. Le concept de réseau global

La disponibilité de produits susceptibles de relier des systèmes de réseaux locaux séparés a transformé ces derniers en modules destinés à former des réseaux étendus de données. La figure ci-dessous illustre une possibilité de groupement de réseaux locaux destiné à la construction d'un réseau étendu.



Le réseau interne de l'entreprise est relié à un réseau public. S'agit-il toujours d'un réseau local ou un MAN.

### 2.2.2. Les applications

Les réseaux locaux sont désormais en mesure de gérer une vaste gamme d'applications sur mini-ordinateur et même quelques applications sur mainframe. Ils supportent également les applications tournant sur PC et stations de travail. Toutefois, il est intéressant de noter qu'aucune application n'a été créée spécialement pour les réseaux locaux. Ceux-ci présentent surtout des avantages en matière de coût, de rapidité et de contrôle de l'utilisateur.

Aujourd'hui, seules les applications à durée critique et requérant une capacité de traitement ou de mémoire importante ont échappé à l'emprise des réseaux locaux. Dans la pratique, les réseaux sont bien plus simples. Les réseaux locaux de la première génération n'avaient que cinq nœuds en moyenne.

L'application de la première génération consistait à partager des fichiers et des imprimantes.

Les applications de la deuxième génération comportent : le partage d'applications, des communications, des bases de données, le contrôle en temps réel et la distribution logicielle.

### 2.2.3. Les avantages

S'il est vrai que les réseaux locaux se contentent de dupliquer des applications habituellement lancées sur mainframes et mini-ordinateurs, pourquoi a-t-on assisté à un tel engouement ?

L'un des principaux facteurs a été la réduction drastique des coûts de mise en œuvre des réseaux locaux. Mais cela ne représente qu'un aspect de l'affaire : les réseaux locaux possèdent, en effet, de nombreux avantages intrinsèques qui permettent de les recommander :

- Facilité d'installation
- Rapidité
- Indépendance vis-à-vis des constructeurs
- Intégration rapide des nouveaux produits
- Facilité de modification et de réorganisation
- Contrôle logiciel
- Coûts d'un réseau local

Il est intéressant d'étudier la répartition des coûts d'installation d'un réseau local élaboré.

Le gestionnaire et le matériel (terminaux, serveurs, cartes d'interface, etc.) représentent respectivement 10% à 30% des coûts.

Les services tels que l'installation, la gestion, la formation, la maintenance représentent 60 % du coût total.

Comme pour les mainframes, les services constituent une part considérable des coûts et un choix prudent peut avoir un impact important sur le coût total d'un réseau local.

### 2.3. Réseau local ou système multi-utilisateur ?

D'une manière générale, il est vrai de dire que les réseaux locaux et les systèmes multi-utilisateur continueront à cohabiter pendant de longues années.

Les applications sur mainframes de grandes sociétés représentent d'énormes investissements qui sont lentement supplantés par ceux destinés aux réseaux locaux. Il est évident que le passage à l'Euro ou l'an 2000 va donner une raison pour changer de stratégie concernant le matériel.

Il est bon de rappeler que le terminal le plus largement utilisé est toujours l'IBM 3270 et non pas le PC. Dans ce cas, il est plus pratique d'utiliser le réseau local comme un point d'accès au mainframe, via une passerelle, ou mieux encore d'intégrer le réseau local dans le mainframe pour des raisons de fiabilité.

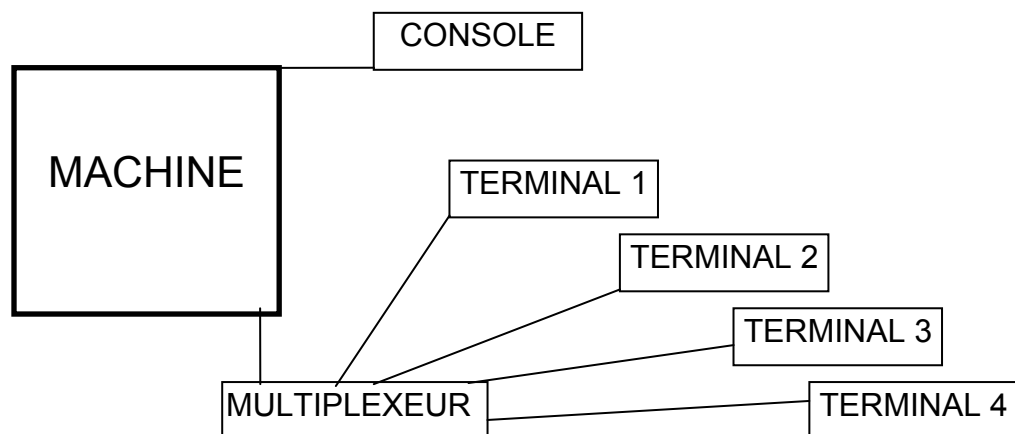
Le véritable problème du choix se pose au niveau du département. Pour les entreprises ayant des besoins limités en informatique et disposant d'un environnement stable, un système clé en main fourni par un constructeur unique peut être la solution aux nombreux problèmes de gestion. Un système multi-utilisateur complet tournant sur un mini-ordinateur privé peut être très économique et c'est la solution que proposent certains grands fournisseurs.

Toutefois, même dans ce cas, les réseaux locaux offrent désormais un support permettant de relier des terminaux à l'unité centrale. La seule différence est ici philosophique : un système central ou un système réseau local.

### 2.4. Les types de réseaux locaux

#### 2.4.1. Les réseaux hiérarchique

On peut les représenter sous forme suivante :

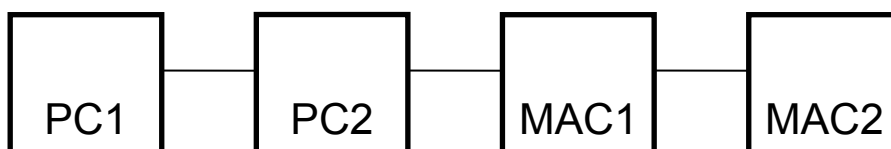


Sur ce réseau vous avez une machine et non un serveur, sur celle-ci est monté un écran et un clavier (console) et des terminaux via un multiplexeur ou une carte interface réseau. Ces terminaux sont passifs et n'effectuent aucun traitements, donc les performances de ce type de réseau sont liés directement à la puissance de la machine. Au départ le choix de cette machine se fait en fonction du nombre d'utilisateurs et des applications à monter. Plus il y aura d'utilisateurs et plus les performances vont diminuer.

En général le système d'exploitation est Unix qui à des avantages et des inconvénients.

Une variante existe en plaçant à la place des terminaux des stations qui sont en général des Pc avec une carte émulant le terminal. L'intérêt est que dans ce cas la station peut effectuer des traitements donc décharger la machine de ces tâches. Attention il faut que l'application soit prévu dans ce sens.

#### 2.4.2. Réseau poste à poste



Cette fois-ci vous n'avez pas de machine centrale mais chaque ordinateur se comporte comme une machine centrale. Chacun peut se connecter sur l'ordinateur de l'autre. Le mot magique est "partage", c'est-à-dire que l'utilisateur d'un ordinateur peut décider de partager un répertoire contenant des données ou un logiciel. Si c'est un logiciel, les autres personnes peuvent utiliser ce logiciel qui sera télécharger dans la RAM mais au prix d'une grande lenteur de téléchargement. Ici chaque ordinateur effectue son propre traitement en logiciel et communication par opposition au réseau hiérarchique. De plus on ne peut connecter que des machines homogènes.

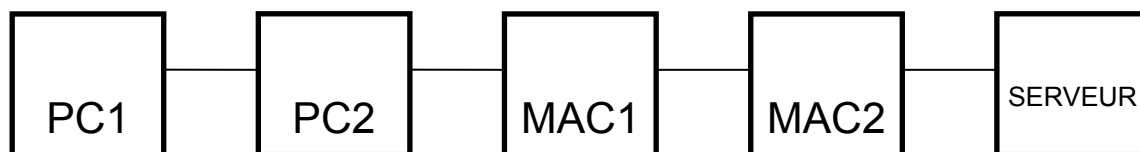
Ce type de réseau à en général une sécurité assez faible et surtout destiné à effectuer du transfert de données et du partage d'imprimante. L'intérêt est essentiellement son coût très faible puisque se limite en général au prix du matériel de câblage. Les systèmes d'exploitation qui fonctionnent dans ce type de réseau est :

- WINDOWS 3.11, 95, 98, Millénium, XP
- WINDOWS NT WORKSTATION, WINDOWS 2000
- POWER LAN
- NOVELL LITE
- MAC LAN

A noter que Windows NT Workstation et 2000 permet d'avoir une sécurité très importante.

On peut aussi trouver une machine qui va jouer le rôle de pseudo-serveur. En général les utilisateurs stockent leurs données sur ce serveur car dessus est monter un streamer qui sauvegarde les données des utilisateurs. Attention en général les logiciels se trouvent en local. A ne pas confondre avec le réseau de type client-serveur.

### 2.4.3. Réseau client-serveur



Ici vous trouvez une machine centrale que l'on appelle **serveur**. Les autres machines sont des clients de ce serveur. Le serveur va centraliser les ressources, c'est-à-dire les logiciels, les données et les périphériques. Sur ce type de réseau vous avez un système de sécurité très évolué et les temps de téléchargement des logiciels dans les clients sont très rapides. De plus on peut interconnecter à l'heure actuelle toutes les machines du marché : PC, MAC, Mainframe, vous trouvez également des utilitaires de messagerie, de sauvegarde, ...

L'inconvénient majeur est le prix, exige un administrateur du réseau et du matériel très fiable surtout en serveur. L'investissement est assez important.

Les principaux systèmes d'exploitations sont :

- NOVELL NT 63% du marché en 1997.
- WINDOWS NT Server, 2000 Server

Ce type de réseau est le point final une informatisation pour une entreprise

Maintenant nous allons étudier la composition d'un réseau local.

## 2.5. Support de transmission

La seconde moitié des années 80 vit s'enflammer les débats concernant les méthodes de transmission ; on opposait principalement le réseau Ethernet au réseau Token Ring.

Aujourd'hui, la question est devenue moins stratégique que tactique et l'intérêt se porte sur l'interconnexion et la gestion de réseaux.

### 2.5.1. Principes

Les fils de cuivre et les fibres optiques constituent les supports de transmission. Par commodité, la station de travail ou serveur et sa connexion physique au support sont appelées "nœud".

La façon dont les nœuds sont connectés entre eux fait appel au concept de topologie plutôt qu'à celui d'architecture.

L'action d'émettre ou recevoir des données sur le support est simplement désignés par le terme "accès".

Enfin, les données sont transmises sous forme de paquets, appelé des **frames**.

### 2.5.2. Les différentes types

Les supports de transmission n'ont jamais été le point fort des réseaux locaux mais les exigences en matière de capacité de gestion, de rapidité de traitement et de flexibilité modifient peu à peu cette image

Quatre types de Supports sont disponibles

- Câble coaxial
- La Paire torsadée (fils de cuivre)
- Fibre optique
- Sans fil

L'agencement des supports au sein des réseaux est décrit dans le chapitre suivant.

#### □ Câble coaxial

Le câble coaxial (coax) a été le premier support utilisé pour les réseaux locaux, en raison de son débit élevé et de sa faible sensibilité aux perturbations électromagnétique. Il est désormais supplanté par la paire torsadée, plus commode et par la fibre, qui est plus rapide et qui permet des transmissions sur une plus longue distance.

Quatre types principaux de coax sont disponibles :

- Câble Ethernet épais 50 Ohms
- Câble fin (Cheapnet) 50 Ohms
- Câble 75 Ohms pour CATV large bande
- Câble orange 90 Ohms pour Token Ring

Le réseau Ethernet a été conçu sur la base du câble dit "épais". Mais l'installation de ce type de câble s'est révélée compliquée et coûteuse; le "câble fin", plus économique et plus souple, a donc fait son apparition. Les câbles coaxiaux large bande sont principalement utilisés pour les réseaux ARCnet.

#### □ Paire torsadée

Deux types de câblage en paire torsadée sont disponibles la paire torsadée blindée (STP) et la paire torsadée non blindée (UTP).

La longueur opérationnelle d'un câble à paire torsadée dépend du débit de transmission, de l'atténuation et de la diaphonie entre les deux conducteur. Plus les conducteurs sont torsadés uniformément, plus la diaphonie est faible. Afin de réduire davantage la diaphonie, chaque conducteur peut être blindé sous forme de paires et de paires multiples.

Les câbles STP sont plus rigides et plus coûteux que les câbles UTP et sont destinés plus particulièrement aux réseaux Token Ring.

Les câbles UTP sont actuellement utilisés pour les réseaux Ethernet et Token Ring atteignant sur des distances : 100 m. L'avantage, par rapport au câble coaxial, est leur faible coût et leur facilité d'installation.

Les câbles UTP et STP sont constitués de plusieurs paires de fils, généralement 6 ou 8 avec une âme extérieure rond ou plat. Le câblage téléphonique existant a parfois remplacé les câbles UTP.

□ Fibre optique

Le principal avantage de la fibre optique est la vitesse du débit. Toutefois, trois autres caractéristiques (la distance, sa sensibilité et la sécurité) rendent son utilisation intéressante pour la transmission au sein d'un réseau local.

Un débit de 100 Mb/s, 1 Gb/s peut être déterminant pour fédérer sur un réseau ou une liaison inter-réseau.

Dans les réseaux locaux Ethernet, des répéteurs sont fréquemment utilisés entre les différents segments du réseau ces liaisons inter-répéteurs à fibre optique (FOIRL) admettent une distance de 4,5 km entre chaque segment de câble (mais elles n'augmentent pas pour autant la longueur maximale de la portion active d'un réseau Ethernet).

Dans les réseaux Token Ring on ajoute habituellement des nœuds de câblage, ce qui permet d'augmenter la distance jusqu'à près de 2 km.

Parmi les quatre types de fibres disponibles, la 62,5/125 µm est la plus communément utilisée pour les applications de transmission de données.

□ Sans fil

Peut-on dire que ce type de transmission constitue un support en soi ? Il ne s'intègre pas à la définition classique d'un réseau local mais il est à prévoir que les réseaux locaux sans fil seront utilisés dans certaines circonstances comme par exemple lors d'une exploitation dans des immeubles anciens ou par des utilisateurs se déplaçant fréquemment comme un réceptionniste.

La transmission se fait par onde infrarouge ou par radio.

□ Tableau récapitulatif

TYPE	COÛT	INSTALLATION	SENSIBILITE	EXTENSION	LONGUEUR
Coax épais	Elevé	Difficile	très peu	peu facile	500 m
Coax fin	faible	Facile	peu	facile	200 m
STP	moyen	Aisée	peu	peu facile	200 m
UTP	faible	Aisée	très	peu facile	100 m
Fibre optique	élevé	Difficile	pas	peu facile	2 500 m
sans fil	élevé	Facile	très	facile	dépend environnement

## 2.6. Topologie

Après les supports, la topologie est la partie la plus simple de la technologie des

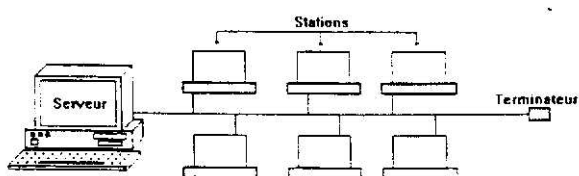
réseaux locaux. C'est un aspect logique, très visuel, clairement structuré, simple et homogène car c'est le schéma de câblage ou de disposition de votre réseau.

Sur les nombreuses topologies possibles, quatre subsistent aujourd'hui

- Bus
- Anneau
- Etoile
- Mixte

Le problème de la topologie a été au centre des débats des années 80. L'anneau ou le bus ? Ce dernier était disponible plus tôt mais le poids lourd de l'industrie IBM supportait sa propre norme, l'anneau.

### 2.6.1. Le bus



*L'architecture en bus (bus).*

Le bus doit se terminer par des terminators ou bouchons de 50 Ohms. Chaque nœud aura un T. Les câbles utilisés sont le coaxial fin ou épais.

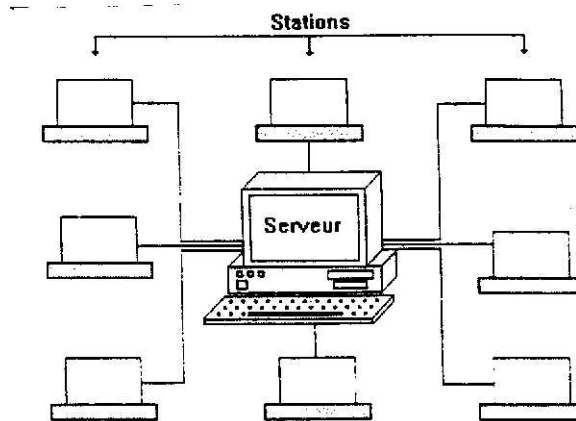
#### *Avantages :*

Extension facile  
Nécessite qu'un câble.

#### *Inconvénients :*

Dépannage difficile  
Défaillance du câble : panne totale du réseau

### 2.6.2. L'étoile



*L'architecture en étoile (star).*

Nécessite du câble paire torsadée.

#### *Avantages :*

Dépannage aisé car vous avez un HUB

Coupure du câble, pas de panne totale du réseau

#### *Inconvénients :*

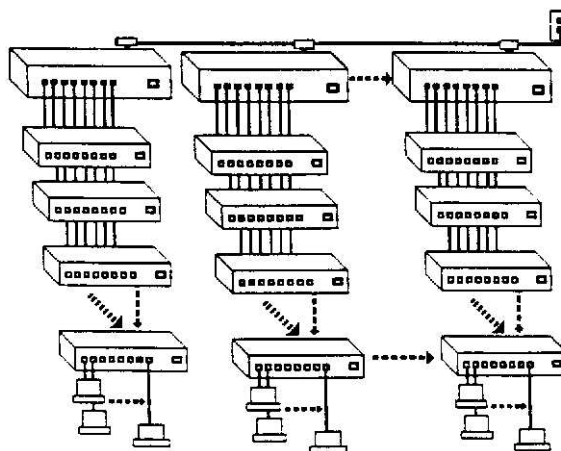
Utilise la paire torsadée dont il vaut un environnement non perturbé

nombre de câble élevé

Extension peu facile

Panne du HUB : panne totale du réseau

### 2.6.3. Mixte



*L'architecture en étoile-bus (star-bus).*

Très pratique pour de gros réseaux et réuni les avantages et les inconvénients du bus et de l'étoile. Le coût est malheureusement le plus élevé.

*Avantages :*

Dépannage aisé car vous avez un HUB

Très bien adapté pour des gros réseaux avec une grande longueur

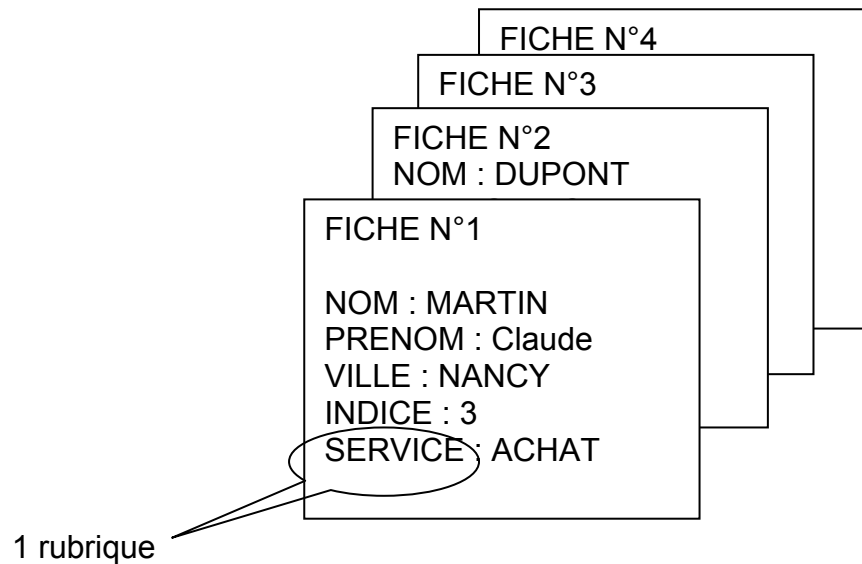
*Inconvénients :*

Coût élevé et installation assez complexe.

### 3. NOTION FONDAMENTALE

#### 3.1. Principe d'une base de données

Une base de données est un ensemble de fiches cartonnées sur lesquelles on trouve des rubriques. Voici un ensemble papier d'une base de données décrivant le personnel d'une entreprise :



Aujourd'hui vous n'êtes pas limité par le nombre de fiches ou le nombre de rubriques mais si vous voulez des temps de réponse rapide lors de vos interrogations vous avez intérêt à vous limiter à un moment à un certain nombre de fiches et de rubriques surtout, en fonction de la puissance de votre ordinateur.

Par contre en informatique, ce type de représentation sous forme de fiche et de rubrique est assez rare et on utilise plus souvent une représentation sous forme de tableau. Cela se transforme assez facilement où les fiches deviennent des lignes et les rubriques de colonnes appelé aussi des champs :

NOM	PRENOM	VILLE	INDICE	SERVICE
MARTIN	Claude	NANCY	3	ACHAT
DUPONT	Marc	NANCY	4	SAV
MARTIN	Henri	METZ	4	SAV
BIDOCHON	Robert	METZ	5	ACHAT
ENFAILLITE	Mélusine	METZ	3	ACHAT
MARTINI	Paul	NANCY	3	PRODUCTION

Maintenant on parle de *table* plutôt que de tableau.

Pour conclure une base de données est composée d'une ou de plusieurs tables et dans une table vous allez trouver des champs (colonnes ou rubriques) et des enregistrements (lignes ou fiches).

### 3.2. Principe d'une clé

Attention l'exemple de mon fichier du personnel d'une entreprise est simple et notamment il me manque un champ capital : le champ que l'on nomme clé.

En effet dans la liste des gens vous avez pu remarquer qu'il avait deux personnes avec le même nom : MARTIN. Pour éviter de faire des confusions qui pourrait être dramatique, il faut **toujours identifier de manière unique un enregistrement** grâce un champ que l'on appelle clé.

Pour faire vous avez deux possibilités :

- ❑ La **clé simple** qui est en général un entier (il est unique)
- ❑ La **clé composé** de plusieurs informations comme le numéro de sécurité social par exemple.

Si la clé composé paraît sur le papier plus séduisante elle est, en réalité plus complexe à mettre en œuvre et surtout il risque d'avoir des doublons mais si il y a beaucoup d'information. L'exemple le flagrant est le fichier des banques ou des impôts qui composé sur le nom + prénom + date de naissance. Malgré cette clé il y a des doublons.

Je vous conseille d'utiliser des clés simples dans un premier temps. Maintenant notre table personnel devient :

IDPERSONNE	NOM	PRENOM	VILLE	INDICE	SERVICE
1	MARTIN	Claude	NANCY	3	ACHAT
2	DUPONT	Marc	NANCY	4	SAV
3	MARTIN	Henri	METZ	4	SAV
4	BIDOCHON	Robert	METZ	5	ACHAT
5	ENFAILLITE	Mélusine	METZ	3	ACHAT
6	MARTINI	Paul	NANCY	3	PRODUCTION

Attention une personne qui porte un numéro, 3 par exemple, ce numéro ne doit surtout pas changer même si on supprime une personne ayant le numéro 2 par exemple.

### 3.3. Conception

Le plus dur quand vous devez réaliser une base de données est l'analyse et ensuite la conception de votre base. Je ne vais pas ici vous faire un cours sur le sujet, ni sur Merise, mais tout simplement une méthodologie.

Vous devez commencer par repérer les sujets. Par exemple si vous devez informatiser un bon de commande, dans un premier temps les différents sujets qui se détache sont les clients du bon de commande, le bon de commande en lui-même, les produits. Tous ces sujets vont devenir des **tables** ou des g et il va falloir les mettre en **relation** entre-elle.

En manière général on dit que chaque entité possède des attributs, par exemple la table client possède des attributs comme le nom, le prénom, la ville du client autrement dit des champs.

### 3.4. Les relations

Nous venons de voir dans le paragraphe précédent qu'il faut très souvent mettre en relation des tables. Pour cela il faut avoir un champ commun entre elle, et bien souvent, on va se servir de la clé.

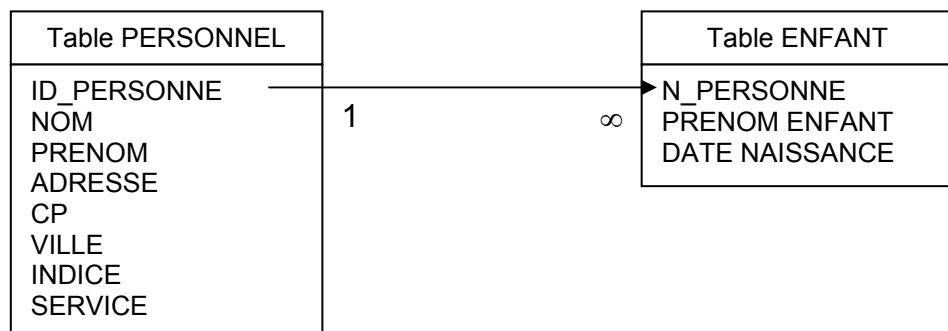
Par exemple si on reprends notre table du fichier du personnel et que vous devez gérer les cadeaux donnés aux enfants du personnel au moment de Noël, comment allez vous y prendre ?.

La première solution est de ne pas faire de deuxième table, mais d'ajouter des champs à votre table personnel. Vous obtenez la structure suivante :

ID PERSONNE	NOM	PRENOM	VILLE	PRENOM ENFANT1	PRENOM ENFANT2
1	MARTIN	Claude	NANCY	JULIE	PAUL
2	DUPONT	Marc	NANCY		
3	MARTIN	Henri	METZ	AGATHE	
4	BIDOCHON	Robert	METZ		
5	ENFAILLITE	Mélusine	METZ	BORRIS	JEAN
6	MARTINI	Paul	NANCY		

On voit tout de suite qu'une telle structure à des limites. En effet nous avons considéré dans cet exemple qu'une personne ne pouvait avoir plus de deux enfants. Bien sûr dans le cas où un employé à 7 enfants par exemple, vous pouvez ajouter d'autres colonnes mais dans ce cas, vous n'avez pas respecter le cahier des charges énoncer au paragraphe 3.1 où je vous disiez de limiter le nombre de champs. De plus vous aurez beaucoup de champs vides.


La deuxième solution est de faire une autre table appelé enfants. Dans ce cas il faut mettre en relation les deux tables. Vous obtenez le schéma suivant :



On dit qu'un employé peut avoir plusieurs enfants et on utilise surtout que trois champs. On parle de relation de un à plusieurs mais vous pouvez avoir des relations de un à un ce qui reste très rare. Attention Access ne gère pas ce type de relation un à un mais on peut ruser en passant par une relation de un à plusieurs puis par une relation de plusieurs à un et qui fait globalement une relation de un à un.

Pour vous entraîner dans la conception des bases de données je vous demande

de me donner le schéma des différentes tables et de leur relation pour informatiser le bon de commande suivant :



**PARTITION MAGIQUE**  
Rue de la Sol  
54000 NANCY  
France  
Téléphone: 04.50.40.45.45 Télécopie: 04.50.40.45.46

Commande N°: 5643535

---

BON DE COMMANDE

**Vendeur**

Nom **BORRIS**

Adresse **Rue de la Sol**

Code postal **54000** Ville **NANCY**

Pays \_\_\_\_\_ Téléphone \_\_\_\_\_

**Correspondant**

Nom **Mr BIDOCHON Robert**

Adresse **34 Rue des 6 Eglises**

Code postal **54000** Ville **NANCY**

Pays \_\_\_\_\_ Téléphone \_\_\_\_\_

Quantité	Unité	Description	Code T.V.A.	P.U.	Montant
2		Partition Symphonie n°45 de Beethoven	2	34,00 E	68,00 E
					Total H.T. 68,00 E
					TVA 13,33 E
					<b>TOTAL 81,33 E</b>

**Paiement**

Chèque

Comptant

N° de compte \_\_\_\_\_

Carte de crédit

Nom \_\_\_\_\_

N° \_\_\_\_\_

Date d'expiration \_\_\_\_\_

**Date de livraison**

23/03/01

## 4. COMMANDES DE BASES

### 4.1. Principes

Le langage SQL permet de faire les tâches suivantes :

- Un langage d'interrogation de base (instruction SELECT)
- Un langage de manipulation des données (UPDATE, INSERT, ...)
- Un langage de définition de données (CREATE, ALTER, DROP)
- Un langage de contrôle de l'accès aux données (GRANT, REVOKE).

Comme tous les SGBD, le résultat d'une interrogation sera présenté à l'écran grâce un utilitaire. Sous Oracle, il s'agit de SQL-FORMS, tandis que sous Access se sera un tableau ou un formulaire. Dans ce support nous n'étudierons pas ces utilitaires, mais je vous conseille de consulter Internet (par exemple [www.commentcamarche.net](http://www.commentcamarche.net)) ou les annexes de ce support pour des informations complémentaires.

Dans SQL il existe des instructions comme SELECT, des arguments à ces instructions et des clauses comme WHERE avec des expressions. Cela nous donne la syntaxe générale suivante :

Instruction arguments clause expressions

Dans la plupart du cas les bases de données sont en générale importante il faut savoir que l'avantage de SQL et que cela demande peu de ressources systèmes comparer à une interface graphique.

De plus, et le plus souvent, les bases de données se trouvent sur un serveur Unix et vous travaillez avec un terminal qui est rarement un terminal graphique et vous êtes obligés de travailler un mode texte.

### 4.2. Création/Suppression d'une table

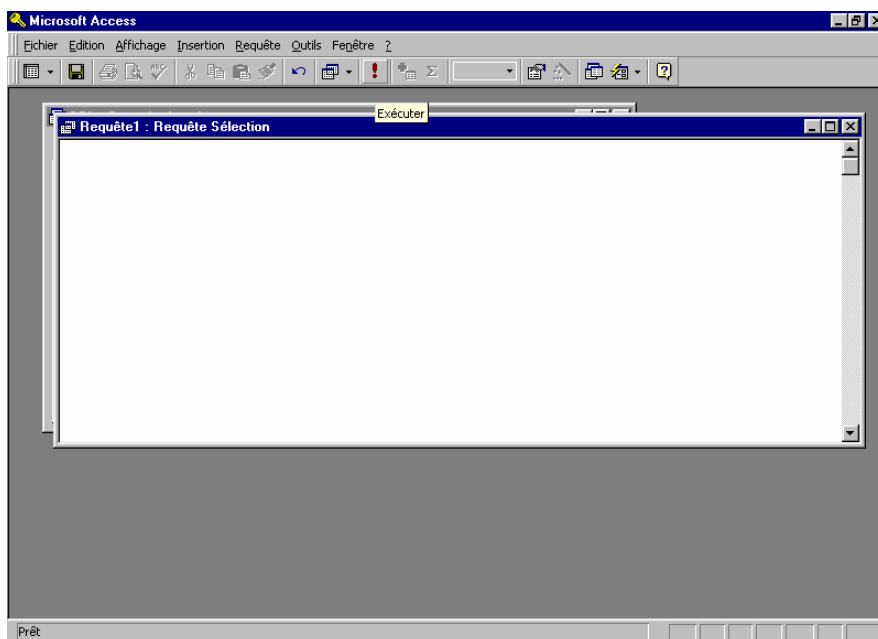
Nous allons créer notre base de données gérant les employés de l'entreprise GUDUBULLE et en partant de l'exemple de la table Employés dont voici le schéma :

Table PERSONNEL
ID_PERSONNE
NOM
PRENOM
ADRESSE
CP
VILLE
INDICE
SERVICE

Pour cela vous devez lancer Access et créer une nouvelle base de données en lui donnant comme nom GUDUBULLE.

Maintenant pour taper des instructions SQL, le plus simple est de passer par une requête. Dans ce cas vous devez :

- ❑ Cliquez sur l'onglet **Requêtes**
- ❑ Cliquez sur le bouton **Nouveau** et choisir le **mode Création**
- ❑ Cliquez sur **Fermer** car vous devez choisir une table qui n'existe pas pour l'instant.
- ❑ Dans le menu **Affichage**, cliquez sur **Mode SQL**
- ❑ Tapez une instruction SQL (vous pouvez effacer SELECT) et pour l'exécuter vous devez cliquer sur le bouton **Exécuter** :



*Dans le langage SQL il n'y a pas de différence entre la majuscule et la minuscule mais pour des raisons de simplicité de lecture, je vous conseille de les taper en majuscule.*

*Normalement pour terminer une instruction SQL vous devez utiliser le point-virgule mais il n'est pas obligatoire.*

Maintenant que vous connaissez le principe pour taper une instruction SQL, nous allons voir comment créer une table.

### **Syntaxe de la création :**

```
CREATE TABLE nomtable (nomchamp1 type (taille) NOT NULL  
CONSTRAINT nomindex PRIMARY KEY, nomchamp2 ....)
```

L'instruction CREATE TABLE se compose des éléments suivants :

<i>Élément</i>	<i>Description</i>
Nomtable	Nom de la table à créer.
Nomchamp	Nom du ou des champs à créer dans la nouvelle table. Vous devez créer au moins un champ.
Type	Type de données du champ dans la nouvelle table (voir les types plus loin).
Taille	Taille du champ en caractères (pour les données de type Texte ou Binaire uniquement).
NOT NULL	Une saisie sera obligatoire pour l'enregistrement concernant le champ.
CONSTRAINT	Instruction permettant de créer un index de type Clé primaire. Vous pouvez aussi faire un index multiple (abordé plus loin dans le support).

*N'oubliez pas de mettre la taille entre parenthèses.*

*Les types de champs sont les suivants : TEXT, BINARY, CURRENCY (monétaire), DATETIME, DOUBLE, INTEGER, MEMO, NUMERIC (Réel double), SINGLE (Réel simple), COUNTER (numéro auto).*

*Il existe l'instruction DESCRIBE qui permet de voir la structure d'une table mais n'existe pas chez Microsoft.*

**Exemple :**

Création d'une table client avec deux champs :

```
CREATE TABLE Clients (Nom TEXT (30), Datecreation DATETIME)
```

Création d'une table client avec un index primaire :

```
CREATE TABLE Clients (Nom TEXT (30), Idclient INTEGER
CONSTRAINT Idclientindex PRIMARY KEY)
```

**Syntaxe de la suppression :**

```
DROP TABLE nomtable
```

*Attention la table doit être fermée*

*L'instruction Drop permet de supprimer également un index.*

**4.3. Saisies des données**

Après la création de la table vous devez la remplir par des données grâce à

l'instruction INSERT INTO. Dans un premier temps nous allons voir comment ajouter un enregistrement.

**Syntaxe :**

```
INSERT INTO nomtable (champ1, champ2, ...) VALUES (valeur1, valeur2, ..)
```

*Attention pour les champs de type texte, vous devez encadrer les valeurs par des apostrophes ("").*

*Vous n'avez pas besoin d'insérer des informations pour les champs de type counter.*

**Exemple :**

```
INSERT INTO CLIENTS (NOM, PRENOM, CHIFFRE) VALUES ("Martin", "Jean", 30)
```

#### 4.4. Voir les données

Après avoir saisi les données, il est normal de voir le résultat. Pour cela vous devez utiliser l'instruction **SELECT**.

**Syntaxe :**

```
SELECT nomchamp1, nomchamp2 FROM nomtable
```

*Si vous voulez voir tous les champs, utilisez le symbole \*.*

*L'ordre des champs à de l'importance pour l'affichage.*

*Vous pouvez aussi utiliser la syntaxe nomtable.nomchamp mais je trouve cela un peu lourd.*

**Exemple :**

```
SELECT clients.Nom, clients.prenom FROM clients
```

Vous pouvez aussi préciser un tri par ORDER BY nom champ. Par défaut l'ordre sera croissant mais pour avoir l'ordre décroissant précisé DESC.

```
SELECT nom, prenom FROM clients ORDER BY nom DESC
```

#### 4.5. Recherche simple des données

Vous pouvez rechercher des données grâce à l'instruction **WHERE** qu'il faut combiner avec l'instruction SELECT.

**Syntaxe :**

```
WHERE nomchamp opérateur expression
```

Pour les opérateurs vous avez le choix entre : = > >= < <= <> Between

Pour réaliser des recherches sur plusieurs champs en même temps utiliser les opérateurs logiques AND ou OR.

Attention n'oubliez pas les guillemets pour le texte et surtout l'instruction LIKE si vous utilisez un caractère générique comme \* ou ?.

Pour les dates il faut utiliser le symbole #.

**Exemple :**

```
SELECT Nom, prenom, chiffre FROM clients WHERE Nom="martin"
ou
SELECT Nom, prenom FROM clients WHERE Nom Like "martin*" AND
chiffre>50
ou
SELECT Nom, datenaissance FROM clients WHERE datenaissance >=
#1/1/89#
```

#### **4.6. Recherche paramétrable**

Vous avez la possibilité de faire des requêtes paramétrable, c'est à dire au moment de l'exécution de celle-ci, il faut que l'utilisateur entre une valeur. Pour cela vous, devez placer entre crochet, votre message qui s'affichera dans l'instruction WHERE mais sans oublier les parenthèses.

Voici un exemple qui recherche le nom d'un client qui sera saisi par l'utilisateur :

```
SELECT nom, prenom FROM clients WHERE (Nom=[Entrez le nom du client])
```

Vous pouvez aussi utiliser l'instruction PARAMETERS qui est une spécificité de Microsoft.

**Syntaxe :**

```
PARAMETERS [Le message] type ;
```

**Exemple :**

```
PARAMETERS [Entrez un nom:] Text ;
SELECT * FROM clients WHERE Nom = [Entrez un nom:] ;
```

#### **4.7. Modifier les données**

Pour modifier les données il faut utiliser l'instruction UPDATE. Attention vous ne pouvez pas annuler une opération de mise à jour et je vous conseille de faire

une instruction SELECT avant afin de vérifier.

**Syntaxe :**

UPDATE nomtable SET nomchamp=valeur WHERE recherche

**Exemple :**

UPDATE clients SET nom = "DUPONT" WHERE idclient=1

Ici vous changez le nom du client n°1 en DUPONT

*Si vous ne spécifiez pas de clause WHERE vous permettez à jour toute la table d'un seul coup. Ceci peut être très utile.*

Par exemple vous voulez augmenter la colonne chiffre de 10% pour tous vos clients. Cela donne l'instruction SQL :

UPDATE clients SET chiffre = chiffre \*1.1

#### **4.8. Supprimer des données**

Pour supprimer des données utilisez la fonction DELETE suivi d'une clause WHERE.

**Syntaxe :**

DELETE FROM nomtable WHERE recherche

**Exemple :**

DELETE FROM clients WHERE chiffre = 0

Ici vous supprimez les clients qui ont un chiffre égal à 0.

*Si vous ne spécifiez pas de clause WHERE vous supprimez toute la table d'un seul coup. Ceci peut être fatale.*

#### **4.9. Elimination des doublons**

Dans une instruction SELECT vous pouvez éliminer les doublons (ligne complète identique) grâce à la clause DISTINCT à taper juste après SELECT.

SELECT DISTINCT nom, prenom FROM clients

#### **4.10. Limitation du résultat**

Vous avez la possibilité de voir les x résultats grâce à la clause TOP ou les x pour % grâce à la clause PERCENT. Attention la clause TOP prend arbitrairement les enregistrements sauf si vous précisez un ordre de tri.

Par exemple vous voulez voir les 10 premiers clients de votre table "clients" :

```
SELECT TOP 10 nom, prenom FROM clients ORDER BY nom
```

*Attention la clause TOP n'effectue pas de choix entre des valeurs égales. Dans l'exemple précédent, si, parmi les 10 premiers clients obtenus, le huitième et le neuvième ont le même nom, la requête renvoie 11 enregistrements.*

Par exemple vous voulez voir 10% de vos clients :

```
SELECT TOP 10 PERCENT nom, prenom FROM clients
```

#### **4.11. Modifier la structure**

Pour modifier la structure d'une table, vous devez utiliser l'instruction ALTER TABLE.

**Syntaxe :**

```
ALTER TABLE nomtable ADD nomchamp type (taille)
```

**Exemple :**

```
ALTER TABLE clients ADD remarques TEXT(25)
```

Vous ajouter un nouveau champ remarques dans la table clients.

#### **4.12. Les transactions**

Une transaction est un ensemble de modifications de la base qui forment un tout indivisible. Un utilisateur peut à tout moment valider ou fermer la transaction en cours par la commande COMMIT.

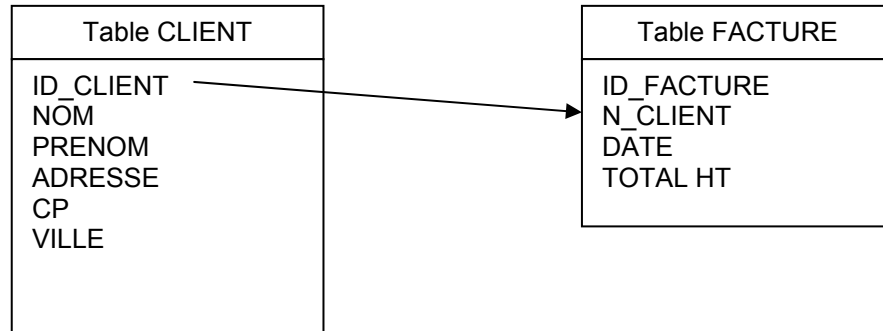
L'utilisateur peut aussi annuler cette transaction par la commande ROLLBACK.

*Cette notion de transaction n'existe pas sous Access car vous pouvez ou non enregistrer votre requête mais si vous voulez l'enregistrer, il faut que la syntaxe soit juste.*

## 5. REQUETE SUR PLUSIEURS TABLES

### 5.1. Jointure

Par exemple vous avez une table "clients" et une table "facture" de ces clients. Voici le modèle relationnel :



A partir de cet exemple vous voulez rechercher toutes les factures d'un client x.

Pour cela vous devez faire une jointure, c'est à dire rechercher les lignes des deux tables qui ont des valeurs identiques au niveau de colonnes mise en correspondance (ici le champ ID\_CLIENT et N\_CLIENT).

Attention il ne faut pas confondre une jointure et une relation.

Bien sûr nous allons utiliser la commande SELECT pour préciser les champs que l'on veut prendre (attention à ne pas oublier le nom des tables cette fois-ci) et l'instruction WHERE pour expliciter la correspondance.

Par exemple nous voulons sortir la liste de tous les clients qui ont une facture. Cela donne la commande SQL devient alors :

```
SELECT Clients.Nom, Clients.prenom, facture.MONTANT FROM Clients, facture
WHERE clients.idclient = facture.n_client
```

Pour compliquer un peu nous voulons la même chose mais en plus uniquement les clients dont le montant de facturation est supérieur à 2000 F. Cela nous donne :

```
SELECT Clients.Nom, Clients.prenom, facture.MONTANT FROM Clients, facture
WHERE clients.idclient = facture.n_client
AND facture.MONTANT >= 2000
```

## 5.2. **Elimination des doublons**

Vous voulez voir les noms des clients qui ont une facture. Cela donne la ligne SQL suivante :

```
SELECT clients.nom FROM clients, facture WHERE clients.idclient =  
facture.n_client
```

Si par exemple un même client à plusieurs factures il va apparaître plusieurs fois dans la liste. Pour éviter cela il faut ajouter la clause **DISTINCTROW**. La ligne devient alors :

```
SELECT DISTINCTROW clients.nom FROM clients, facture  
WHERE clients.idclient = facture.n_client
```

## 5.3. **La clause IN**

Vous pouvez aussi retrouver dans une requête des données provenant d'une autre table grâce à la clause **IN** qui introduit une sous-requête. Pour cela il suffit de taper **IN** suivi de votre seconde requête entre parenthèse.

Par exemple vous voulez voir le client dont le numéro de facture est le numéro 1

```
SELECT Nom, prenom FROM Clients WHERE IDCLIENT  
IN (SELECT N_CLIENT FROM facture WHERE ID_FACTURE=1)
```

*Vous pouvez dans une clause **IN** faire de nouveau une autre clause **IN**. Dans ce cas vous utilisez une sous requête dans une autre sous requête. Ceci est semblable à une calculatrice utilisant des parenthèses.*

## 5.4. **La clause EXISTS**

La clause **EXISTS** est suivie d'une sous-interrogation entre parenthèses et prend la valeur vrai s'il existe au moins une ligne satisfaisant les conditions de la sous-interrogation.

```
SELECT Nom, prenom FROM Clients WHERE EXISTS (SELECT NULL FROM  
facture WHERE ID_FACTURE=1)
```

## 5.5. **La clause INNER JOIN**

Vous permet de faire une jointure comme sur SQL ORACLE

```
SELECT champs1, champs2 FROM table1 INNER JOIN table2 ON table1.  
champs1 = table2. Champs2
```

## 5.6. Les alias

Pour vous simplifier la vie et surtout pour éviter d'avoir une ligne trop longue à taper, vous avez la possibilité d'utiliser les alias. Un alias permet de renommer une table ou une colonne.

Pour créer un alias d'une table, il suffit de taper le nom de la table suivi d'un espace et du nom de l'alias.

Par contre pour créer un alias d'une colonne il faut utiliser l'instruction AS suivi du nom de l'alias de la colonne.

Cela nous donne comme exemple :

```
SELECT c.nom As Name, c.prenom, c.chiffre FROM clients c
```

## 5.7. La fusion de deux tables

Vous pouvez réunir deux tables, c'est à dire faire une fusion des enregistrements à condition qu'il ait le même nombre de champs dans les deux tables mais pas forcément du même type. Cette réunion se fait grâce l'instruction UNION.

### **Syntaxe :**

```
TABLE nomtable1 UNION TABLE nomtable2
```

### **Exemple :**

```
TABLE clients UNION TABLE fournisseurs
```

Vous pouvez aussi précisez que certains champs et/ou certaines conditions.

Voici un exemple qui réunit deux tables avec seulement deux colonnes et suivant une condition sur la première table :

```
SELECT nom, prenom FROM clients WHERE chiffre > 60
      UNION
      SELECT montant, n_client FROM facture;
```

*Par défaut, l'opération UNION ne renvoie aucun enregistrement en double mais vous pouvez ajouter ALL devant UNION pour obtenir de façon certaine que tous les enregistrements soient renvoyés. La requête s'exécutera plus rapidement par ailleurs.*

*Attention il faut impérativement le même nombre de colonnes.*

## 6. LES CALCULS

### 6.1. Création d'un champs calculé

Vous pouvez créer un champ calculé en utilisant le principe de l'alias mais en ajoutant une formule de calcul.

Par exemple vous voulez avoir une colonne Dépt qui rendra les deux premiers caractères du champs code postal. Cela nous donne la ligne SQL suivante :

```
SELECT nom, prenom, Left([code postal],2) AS Dépt FROM clients
```

| *Il faut connaître bien sûr par cœur le nom des fonctions d'Access.*

### 6.2. Les statistiques

Vous pouvez des statistiques sur un champs grâce à des fonctions de calcul comme count(), avg(), sum(), max, min,... Pour cela on utilise l'instruction SELECT suivi de la fonction de calcul.

Par exemple vous voulez connaître le nombre total de vos clients :

```
SELECT Count(nom) AS Nbeclients FROM clients
```

Maintenant le nombre de vos clients qui ont un chiffre > 50 :

```
SELECT count(nom) AS Nbeclients FROM clients WHERE chiffre > 50
```

| *Il faut connaître bien sûr par cœur le nom des fonctions d'Access.*

| *Vous ne pouvez afficher qu'une seule colonne dans l'instruction SELECT.*

### 6.3. La clause GROUP BY

Utile pour faire des calculs, vous pouvez créer des groupes dans une table par la clause GROUP BY nomchamp pour éliminer les doublons par exemple.

Par exemple :

```
SELECT nom FROM client GROUP BY nom
```

Ou encore pour afficher le nombre de client :

```
SELECT nom, Count(prenom) AS Nbeclients FROM client  
GROUP BY nom
```

## 7. L'ADMINISTRATION

### 7.1. Les vues

La plupart des systèmes SQL sont capables de donner à chaque utilisateur sa propre image des données. On parle de vue. Sur Access une requête est une vue donc par conséquent l'instruction pour créer une vue n'existe pas, mais comme le cours se nomme SQL je dois vous en parler quand même, par exemple si vous travaillez sur Oracle.

#### **Syntaxe :**

```
CREATE VIEW nomvue AS SELECT nomchamp1, nomchamps2
FROM nomtable WHERE recherche
```

Vous voyez que d'après la syntaxe SQL une vue est bien une requête que vous avez enregistré avec une commande *SELECT*.

### 7.2. La sécurité

La sécurité consiste à donner des autorisations d'accès, s'est à dire certains utilisateurs peuvent seulement consulter, d'autres consulter et modifier, etc... L'instruction qui permet de donner des droits est *GRANT* et pour les enlever est *REVOKE*. Sur Access cette instruction n'existe pas, mais voici quand même la syntaxe :

#### 7.2.1. Mettre des droits

#### **Syntaxe :**

```
GRANT droit ON nomtable (champ1, champ2) TO user1, user2
```

Pour les droits :

Consulter	SELECT
Ajouter	INSERT
Modifier	UPDATE
Supprimer	DELETE
Créer un index	INDEX
Modifier la structure	ALTER
Tous les droits	ALL

#### **Exemple :**

```
GRANT update ON clients (nom, prenom) TO paul
```

Ici l'utilisateur paul pourra modifier les nom et prénom de la table clients.

```
GRANT all ON clients TO henri
```

Ici l'utilisateur henri pourra tout faire sur la table clients.

### 7.2.2. Enlever des droits

**Syntaxe :**

```
REVOKE droit ON table FROM user1
```

**Exemple :**

```
REVOKE update ON clients TO paul
```

Ici vous ôtez les droits de modifier pour l'utilisateur paul sur la table clients.

### 7.3. Les index :

Essayez de chercher dans une liste, un nom, sachant que cette liste n'est pas triée sur les noms. Vous allez perdre un temps fou à chercher alors qu'il aurait fallu avoir tout simplement cette liste rangée dans l'ordre alphabétique pour aller beaucoup plus vite. En informatique cela est la même chose. Pour faire des recherches rapides, il faut créer des "index" sur un champ ou un ensemble de champs. Une autre analogie pour mieux comprendre l'utilité des index et l'index d'un livre qui permet d'accéder très rapidement à une page. Un index est un fichier à part qui contient une ou plusieurs colonnes mais ne servant que pour les recherches ( à ne pas confondre avec la clause ORDER BY qui fait un tri sans générer un fichier à part).

L'inconvénient des index est qu'ils occupent de la place et surtout il doit être mis à jour et cela prends du temps.

Je vous conseille de faire des index uniquement sur des recherches qui se font journalièrement.

#### 7.3.1. Index simple

Un index simple est un index sur un champ.

**Syntaxe :**

```
CREATE UNIQUE INDEX nomindex ON nomtable (nomchamp) ordre
```

Vous pouvez taper la clause UNIQUE pour éliminer les doublons.

L'ordre est soit DESC ou ASC pour décroissant et croissant

*Vous ne pouvez pas créer deux fois un index portant le même nom.*

*Un index ne se voit pas à l'écran. Le seul moyen pour vérifier si il existe est, d'aller dans la structure de la table puis dans le menu Affichage/Index.*

**Exemple :**

```
CREATE INDEX nomclient ON clients (nom) DESC
```

Vous créez un index de la table clients basé sur le champ nom dans l'ordre décroissant mais avec les doublons.

### 7.3.2. Index multiple

Un index multiple est un index sur plusieurs champs.

**Syntaxe :**

```
CREATE UNIQUE INDEX nomindex ON table (champ1 ordre, champ2 ordre)
```

**Exemple :**

```
CREATE INDEX nompreclient ON clients (nom DESC, prenom ASC)
```

Vous créez un index de la table clients basé sur le champ nom et prénom dans l'ordre décroissant pour les noms et croissant pour les prénoms.

### 7.3.3. Suppression

Pour supprimer un index il faut utiliser l'instruction DROP.

**Syntaxe :**

```
DROP INDEX nomindex ON nomtable
```

| *Attention la table doit être fermée*

**Exemple :**

```
DROP INDEX nomclient ON clients
```

Vous supprimez l'index nomclient de la table clients.

## 7.4. Les contraintes d'intégrité

Une contrainte d'intégrité est une règle que les données doivent respecter. Par exemple un champ ne peut avoir que trois valeurs par exemple.

Pour réaliser ces contraintes d'intégrité il faut utiliser l'instruction CHECK.

Attention cette instruction est non disponible sous Access.

**Syntaxe :**

```
CHECK (nomchamp IN (valeur1, valeur2, valeur3))
```

**Exemple :**

```
CHECK (categorie IN ('AP', 'EM', 'SP'))
```

| *Sous Access il existe une contrainte qui est NOT NULL. Cette contrainte permet de vérifier que les données que vous entrez réponds bien aux contraintes.*

## 7.5. Fusion

Vous pouvez faire une fusion entre deux tables suivant un champ commun entre les deux tables qui contient des valeurs identiques. C'est la clause INNER JOIN.

### **Syntaxe :**

```
FROM table1 INNER JOIN table2 ON table1.champ1 oprcomp table2.champ2
```

Oprcomp est un opérateur de comparaison tel que : = <> > >= < <=

| *L'instruction INNER JOIN est toujours utilisé avec l'instruction SELECT.*

### **Exemple :**

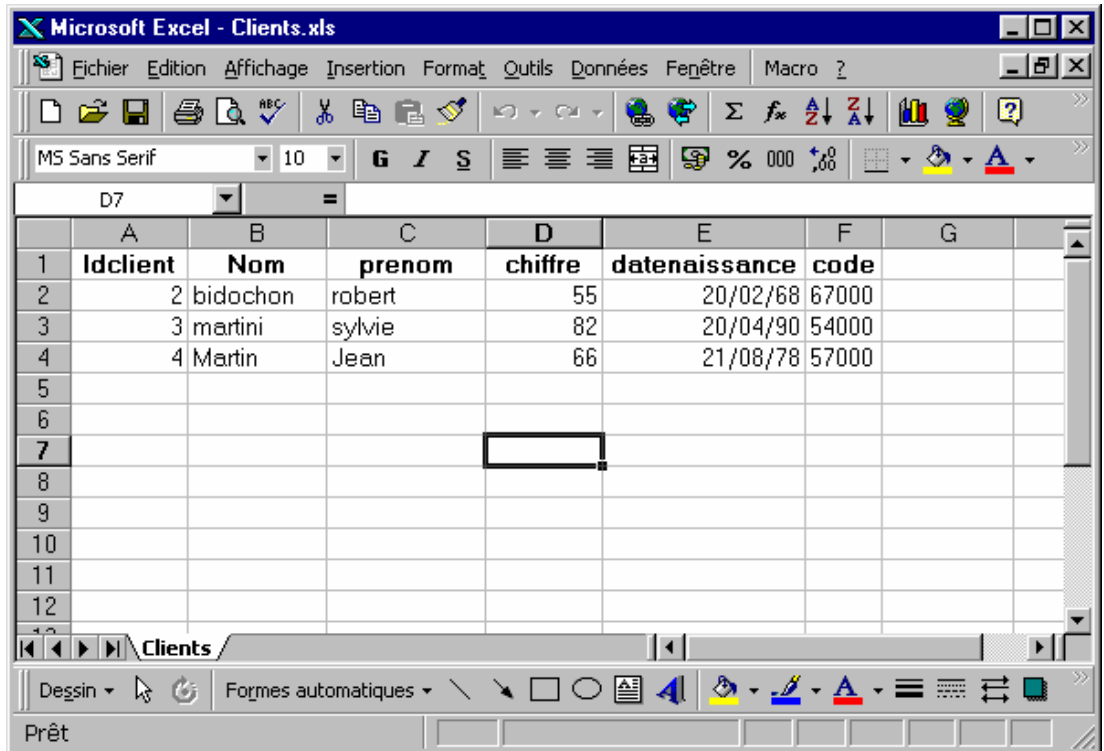
```
SELECT nom, prenom FROM clients INNER JOIN facture  
ON clients.idclient= facture.n_client
```

Vous sélectionnez le nom et le prénom des clients qui ont une facture.

## 8. SQL AVEC EXCEL

### 8.1. Les tableaux

Avant de voir comment faire une interrogation SQL avec Excel, il faut d'abord avoir une base de données sous Excel. Le plus simple est de faire un tableau organisé comme une table sous Access. Voici l'exemple de notre table Clients d'Access sous Excel (vous pouvez aussi faire une exportation à partir d'Access vers Excel).



The screenshot shows a Microsoft Excel window titled "Microsoft Excel - Clients.xls". The spreadsheet contains a table with the following data:

	A	B	C	D	E	F	G
1	Idclient	Nom	prenom	chiffre	datenaissance	code	
2	2	bidochon	robert	55	20/02/68	67000	
3	3	martini	sylvie	82	20/04/90	54000	
4	4	Martin	Jean	66	21/08/78	57000	
5							
6							
7							
8							
9							
10							
11							
12							

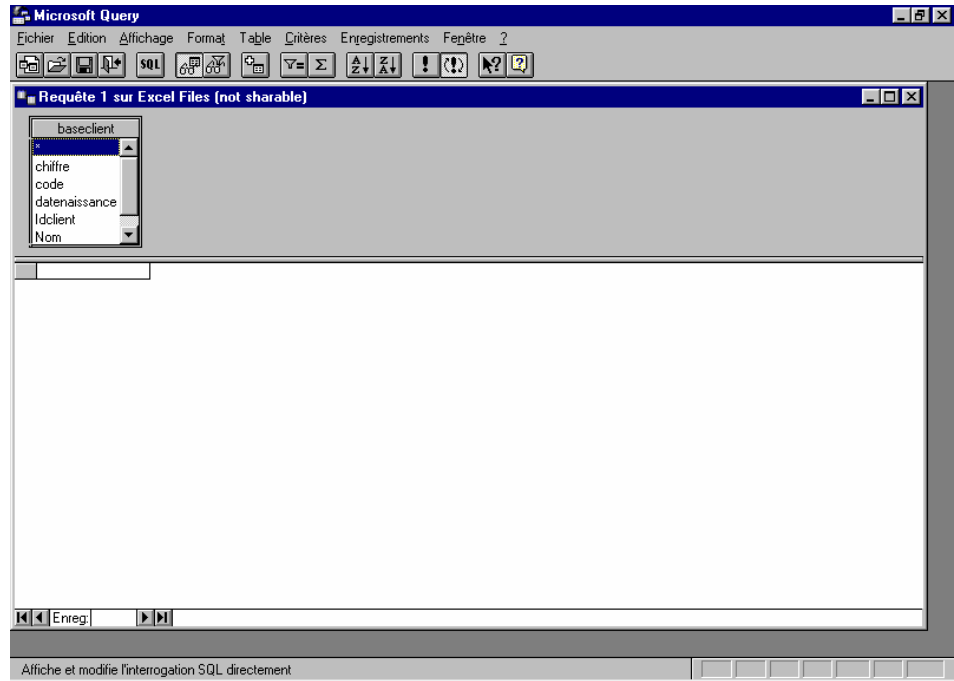
Je vous conseille de nommer votre base. Pour cela sélectionner votre base et donnez un nom dans la zone des noms en haut en gauche.

### 8.2. Création d'une requête

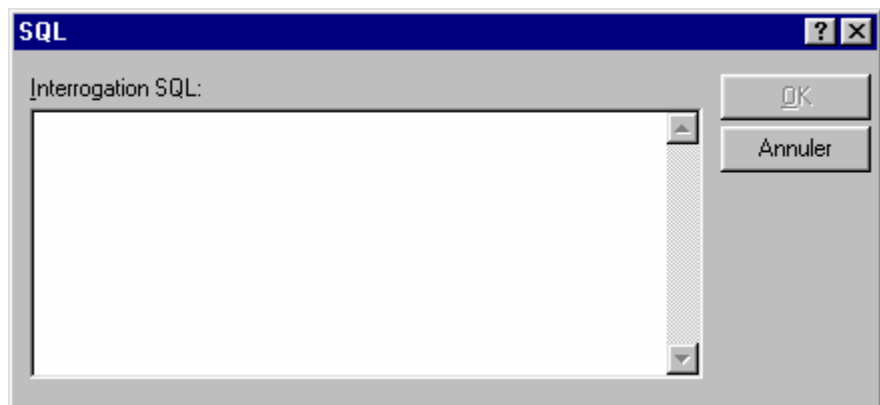
Pour créer une requête SQL dans Excel, nous allons utiliser un utilitaire qui s'appelle "Microsoft Query".

Pour cela vous devez aller :

- Dans le menu **Données**, puis **Données Externes** et **Créer une requête**.
- Choisissez "**Excel Files**" puis sélectionnez votre fichier Excel. Maintenant vous vous trouvez dans Microsoft Query.
- Sélectionnez une table qui sont les plages de cellules nommées.



- Maintenant vous pouvez passer directement en mode SQL en cliquant sur l'icône SQL et vous obtenez l'écran suivant :



- Tapez une instruction SQL et cliquez sur OK pour voir le résultat.

*Vous pouvez enregistrer votre requête (.dqy) que vous pouvez facilement ouvrir par le menu Données/Données externe/créer une requête/Requête/Parcourir .*

*Le gros plus avec SQL et que vous pouvez faire des jointures entre deux tableaux ce qui impossible à faire avec Excel seul.*

*Dans la cause FROM vous devez spécifier le chemin complet pour accéder à votre fichier par `chemin\nom`nomtable.*